Loop-Closure Detection with a Multi-Resolution Point Cloud Histogram Mode in Lidar Odometry and Mapping for Intelligent Vehicles

Qingyu Meng, Hongyan Guo *Member, IEEE,* Xiaoming Zhao, Dongpu Cao *Member, IEEE,* and Hong Chen, *Senior Member, IEEE*

***Abstract*—Precise positioning is the basic condition for intelligent vehicles to complete perception, decision-making and control tasks. In response to this challenge, in this paper, lidar simultaneous localisation and mapping (SLAM) is taken as the research object, and a SLAM system is designed that integrates motion compensation and ground information removal functions, and can construct a real-time environment map and determine its own position on the map while the vehicle is driving. A loop-closure detection method with a multi-resolution point cloud histogram mode is proposed, which can effectively detect whether the vehicle passes through the same position and perform optimisation to obtain globally consistent pose and map information in the urban conditions with more driving loops. We conduct experiments on the well-known KITTI dataset and compare the results with those of state-of-the-art systems. The experiments confirm that the lidar SLAM system designed in this paper can provide accurate and effective positioning information for intelligent vehicles. The proposed loop-closure detection algorithm has excellent real-time performance and accuracy, which can guarantee the long-term driving operation of these vehicles.**

***Index Terms*—Lidar SLAM, Loop-Closure Detection, Pose Estimation, Multi-Resolution Histogram, KITTI Dataset.**

I. Introduction

An intelligent vehicle integrates a very complex system structure, and scholars usually divide this structure into several sub-modules for more specific research, generally including advanced perception, decision-making, and control algorithms [1]–[3]. A major prerequisite for the above tasks is that the intelligent vehicle has accurate positioning capabilities; only after the location of the intelligent vehicle on the map is determined can the obstacle information transmitted by the sensing module have credibility [4]–[6]. The main reason why positioning errors are difficult to eliminate is that the sensor output used to estimate the track of an intelligent vehicle is affected by noise. The existing positioning sensors on intelligent vehicles mainly include an inertial measurement unit (IMU) and a global positioning system (GPS) [7]. The former measures the acceleration and angular velocity of the carrier and then calculates the pose; the latter uses real-time differential positioning technology to provide accurate pose estimation for the carrier. However, the measurement data of the IMU are affected by a variety of factors, including random noise and temperature, which cause errors in the estimation of the pose and transmission at each moment. The GPS system works at low frequencies and the signal is easily blocked. It cannot continuously and stably output pose information in urban areas with tall buildings, or on roads blocked by trees or viaducts, causing the positioning system to fail in a short time. An effective solution to eliminate this error is a subfunction of simultaneous localisation and mapping (SLAM), called loop-closure detection [8], [9]. It uses the observation of surrounding scenes during the last pass to correct the state of the intelligent vehicle at the current time when the intelligent vehicle has travelled to a previously passed scene. Furthermore, the positioning error accumulated during driving can be eliminated, and a globally consistent pose estimation can be obtained.

According to the different sensor input signals, loop-closure detection algorithms are mainly divided into two types: vision-based and point-cloud-based [10], [11]. The vision-based loop-closure detection method has been greatly developed. For example, the well-known ORB-SLAM system uses a bag-of-words model to construct the features of keyframes and establish the corresponding database. It compares the similarity between current features and historical features to determine whether the driving trajectory forms a closed loop [12]. Some loop-closure detection algorithms add inertial constraints, and iteratively update the pose of the intelligent vehicle through filtering or graph-based optimisation methods and detect whether a loop occurs in real time, thereby increasing the accuracy and robustness of loop-closure detection [13]–[15]. However, the vision-based loop-closure detection algorithm often needs to deal with high-dimensional graphics state variables, using K-means clustering and other methods to classify the features, which necessitates substantial calculations and required storage space [16]. Additionally, the image information is highly susceptible to light, and questions about the safety of prolonged use of this novel method for intelligent vehicles have been raised.

In recent years, the method of using point cloud data to construct loop-closure detection models in lidar SLAM systems has begun to attract more attention. LOAM subtly divides the pose estimation problem into two algorithms and relies on the edge and planar features of the point cloud to implement a

Q. Meng and H. Guo and X. Zhao are with the State Key Laboratory of Automotive Simulation and Control and the Department of Control Science and Engineering, Jilin University (Campus NanLing), Changchun 130025, PR China (corresponding author email: guohy11@jlu.edu.cn).

D. Cao is with the Department of Mechanical and Mechatronics Engineering, University of Waterloo, Canada (e-mail: dongpu@uwaterloo.ca).

H. Chen is with the Clean Energy Automotive Engineering Center, Tongji University, Shanghai, 201804, China (e-mail: chenhong2019@tongji.edu.cn).

real-time online lidar SLAM system; however, the loop-closure detection module in LOAM is mainly dedicated to maintaining an accurate map instead of optimising the pose [17]. Another straightforward approach is to violently identify whether the current scene is revisited through point cloud registration, and to use a coarse-to-fine strategy to perform registration step by step to correct the pose of the intelligent vehicle [18]. LEGO-LOAM is a lightweight lidar SLAM system that estimates the 6 degrees-of-freedom position and attitude of the carrier in two steps and has shown a closed-loop detection ability in a variety of scenarios [19]. The histogram-based method can quickly construct the spatial geometric features of 3D point clouds, express them in the form of histograms, and determine whether a loop is formed by comparing the similarity of histograms for different scenes [20]. The typical lidar loop-closure detection performance is shown in Fig. 1.

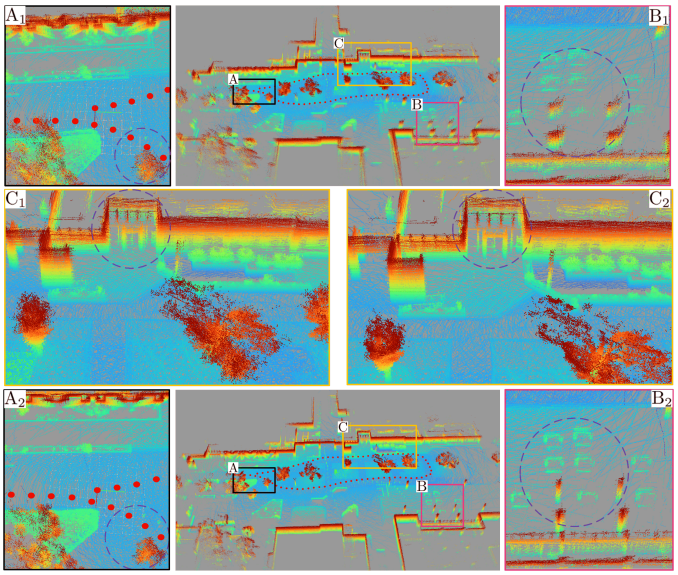


Fig. 1. Lidar loop-closure detection effect diagram. The middle part of the first and third rows in the figure are the point cloud maps before and after loop-closure detection. We use the three boxes A, B and C to circle the local map, and the lower right index 1 and 2 are the maps before and after correction respectively. It can be seen that with loop-closure detection, the trajectory is adjusted and the map is clearer.

In addition to the feature-based lidar loop-closure detection method described above, the segment-based method has also become a research hotspot. A data-driven description method based on the 3D segment point cloud called SegMap was proposed, which classifies the point cloud segments through a customised machine learning algorithm and verifies a potential loop closure identified by the matching procedure with the RANSAC geometric test [21]. Surfel-based SLAM defines a point cloud map representation called a surfel map; when a new point cloud is input, the current frame is projected onto the surfel map. If map overlap is found for a location far from the current moment, then a loop is considered to be detected, and a pose graph is used to optimise the map and trajectory [22]. However, these methods have their shortcomings: (a) some of these methods cannot be run in real time, and the input of the system needs to be reduced in frequency to run [19]; (b) some excessively rely on a certain feature, resulting in insufficient algorithm generalisation ability, and cannot accurately identify insignificant loops [23]; (c) the data-driven learning classification method is less interpretable than other methods [24]; and (d) some methods focus on correcting the map rather than on optimising the driving trajectory [25]. In summary, providing a lidar loop-closure detection algorithm for intelligent vehicles is still a difficult challenge to address.

Based on the above discussion, we take lidar point cloud data as input and discuss methods that may affect the accuracy of the point cloud and eliminate redundant information. Through an efficient feature extraction and matching algorithm, we provide reliable initial conditions for pose estimation, and the corresponding relationship between points is used to construct the analytical form of the residual equation of the optimisation problem. The frame-to-frame pose transformation is iteratively estimated by the Levenberg-Marquardt algorithm [26]. In particular, for the driving characteristics of intelligent vehicles that may pass through repeated scenes, a loop-closure detection algorithm based on multi-resolution feature histograms is proposed to correct driving trajectories and maps. The main contributions of this paper are as follows:

1) Considering the influence of point cloud distortion caused by motion, ground information and other factors, a complete set of lidar SLAM systems with good performance is designed to complete pose estimation and environment reconstruction.
2) A loop-closure detection algorithm is proposed to construct point cloud feature histograms of different types and resolutions. Meanwhile, a similarity measurement is used to evaluate the differences between the histograms to identify whether the current vehicle location represents a previously passed scene.
3) We integrate a factor graph optimisation method in our lidar SLAM system, which treats the detected loop information as a constraint condition for graph optimisation, then optimise the driving trajectory of the intelligent vehicle and obtain a globally consistent point cloud map.

The outline of the remainder of the paper is as follows. In Section II, we discuss methods including motion compensation and ground information estimation, and provide a strategy for feature extraction and matching. In Section III, we comprehensively introduce the proposed loop-closure detection algorithm and realise global optimisation based on loop information. In Section IV, we give the main experimental results and discussion of the paper. Section V provides the conclusion of the paper.

II. Preprocessing of the Point Cloud

The raw data collected by lidar are often very extensive in latitude and redundant in information, which makes directly solving the problem of point cloud registration and state estimation difficult. In view of the above problems, this section mainly studies point cloud data preprocessing. We perform motion distortion compensation, ground segmentation, and feature extraction on the input original point cloud data, which provides a basis for the design of the subsequent system. In addition, we define some common symbols in the paper.

A. System Anatomy

Denote the world coordinate frame of the lidar system by W, which is referred to as the absolute reference. Denote the lidar coordinate frame by L, where its origin is located at the geometric centre of the lidar. $\mathcal{P}_k$ indicates the point cloud perceived during sweep k, and the coordinates of a point i, $i \in \mathcal{P}_k$, in L_k are represented by $p^L_{(k,i)}$.

Fig. 2 shows the overview of our lidar SLAM system. The point cloud data are input into two modules at the same time, namely, the pose estimation module and the loop-closure detection module. The pose estimation module first preprocesses the original data, then realises the frame-to-frame pose estimation and backprojects the point cloud to the 3D space to construct the problem map. The loop-closure detection module constructs keyframes for the voxel dimensionality reduction point cloud data and draws a histogram according to the cell type. When the histogram similarity of the keyframe meets the threshold, loop detection is performed, the system state is optimised, and finally, the pose, trajectory and map information are output.

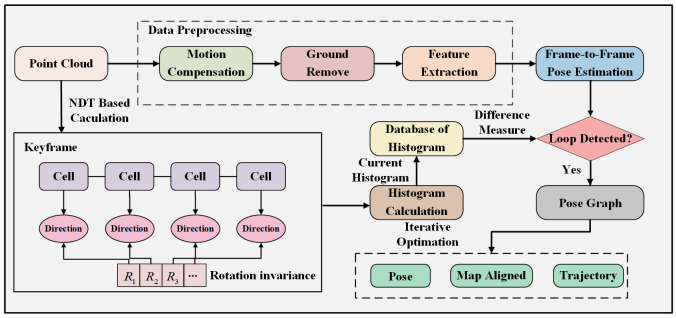
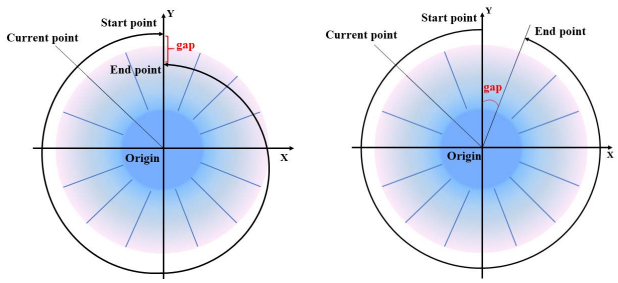


Fig. 2. Workflow of the designed system.

B. Motion Compensation and Ground Estimation

The movement of the vehicle causes distortion of the point cloud data constructed by the lidar, as shown in Fig. 3. When an intelligent vehicle is in a scene surrounded by tall buildings, the point cloud should truly restore the structure and texture of these buildings under normal circumstances. However, when the vehicle is moving rapidly, such as in the forward direction, the position of the lidar changes with the vehicle body during a sweep, resulting in a gap in the front and rear directions of the final point cloud, as shown in Fig. $3(a)$. Similarly, this gap also occurs when the vehicle is turning, which is reflected in Fig. $3(b)$.

To solve the above problem, we need to find a coordinate transformation relationship to project all points within a sweep period $[t_k, t_{k+1}]$ into the lidar coordinate system L_{t_k}, where t_k and t_{k+1} are the starting times of sweeps k and $k+1$, respectively. Let $T^L_{k,k+1}$ be the lidar pose transform for $[t_k, t_{k+1}]$, which contains 6 degrees of freedom, expressed as $T^L_{k,k+1} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z]^T$, where θ_x,θ_y and θ_z are rotation angles about the X-axis, Y-axis and Z-axis of L, respectively, and t_x, t_y and t_z represent the translations along the three axis. Fig. 4 presents an overview of our motion compensation scheme. Since the lidar has a very high



(a) Point cloud distortion caused by translation. (b) Point cloud distortion caused by rotation.

Fig. 3. Lidar point cloud distortion caused by intelligent vehicle movement.

operating frequency, we regard the motion modelling between two sweeps as a uniform velocity model [28]. On this basis, for the sake of smoothness, consider the point data in two consecutive sweeps; then, for a point $i, i \in \mathcal{P}_k$, let t_i be the timestamp. $T^L_{k,i}$ can be obtained by linear interpolation:

$$T^L_{k,i} = T^L_k \frac{t_i - t_k}{t_{k+1} - t_{k-1}} \tag{1}$$

where T^L_k can be obtained from lidar odometry as described in Section III. Fig. 4 shows the overview of coordinate transformation in motion compensation. L_{t_k} represents the lidar coordinate system at the beginning of the k-th sweep, L_{t_i} represents the coordinate system of the point scanned in the k-th sweep, and (1) obtains the relative pose between L_{t_i} and L_{t_k} through the linear interpolation between t_{k-1} and t_{k+1} to backproject L_{t_i} into L_{t_k}.

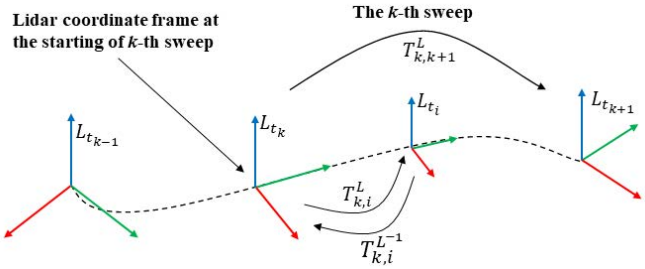


Fig. 4. The position of the point cloud in the lidar coordinate system at different times and the schematic diagram of distortion compensation.

In addition, we estimate the ground information because when lidar is used for intelligent vehicles, it receives a large number of point clouds from the ground. The discrimination between them is very low, and considerable redundant information is obtained, which negatively impacts the efficiency of the odometry, causing the loop detection module accuracy to decrease. To eliminate the influence of ground information, we project 3D point clouds into 2D planes in columns to perform local ground estimation and then connect the ground information between frames to remove ground information [29].

C. Feature Extraction and Matching

The point cloud with motion compensation and ground information removal is the desired system input, and we next perform feature extraction on each frame of the point cloud to realise lidar odometry. In general, we mainly extract feature points from two regions: edge and planar regions. For a point i in one sweep, $i \in \mathcal{P}_k$, define S as a set of consecutive points obtained by the same laser beam near point i, and use the following criteria to measure the smoothness of the local surface:

$$c = \frac{1}{|S| \cdot \|p^L_{(k,i)}\|} \| \sum_{j \in S, j \neq i} (p^L_{(k,i)} - p^L_{(k,j)}) \| \tag{2}$$

In this paper, we specify $S = 10$; that is, we comprehensively evaluate the 5 points on each side of each point to calculate the smoothness of the point. We traverse all the points of a sweep to obtain their corresponding values of c and then reorder these points according to the values of c; specifically, larger values of c correspond to edge features, and smaller values correspond to planar features. To extract point features uniformly, $\mathcal{P}_k$ is divided into 4 sub-regions according to the timestamps of each scan point. The edge and planar features selected in each sub-region are generally limited to at most 2 edge points and 4 planar points during each laser beam scan in one week; that is, the first four feature points determined in descending order of c and the last two feature points determined in ascending order of c returned by a beam are considered target features. Note that those points with c values within the threshold range are also recognised as edge points and planar points but are not the objects that need to be extracted. $\mathcal{F}^e_k$ and $\mathcal{F}^p_k$ denote all the edge points and planar points that meet the threshold range in the k-th sweep, and $\mathbb{F}^e_k$ and $\mathbb{F}^p_k$ are the target edge points and planar points in the k-th sweep.

The feature points in the point clouds of two adjacent frames can be registered by three-dimensional transformation to restore the motion information between frames. This step is usually implemented by the iterative closest point (ICP) algorithm, which finds the correspondence between the two sets of point clouds and calculates the optimal transformation until the convergence accuracy requirements for correct registration are met [30]. For such an optimisation problem, a good initial value is beneficial to reduce iterations and avoid falling into a local minimum, and this initial value is often given by feature matching. For edge point matching, let i be a point in $\mathbb{F}^e_{k+1}$, $i \in \mathbb{F}^e_{k+1}$. Since an edge line is determined by at least two points, we first determine a point j that is closest to i in $\mathcal{F}^e_k$ and then determine a point l that is the closest neighbour of i in the two sweeps consecutive to the sweep of j, where (j, l) constitutes the matching for i.

Similarly, for planar point matching, we have $i \in \mathbb{F}^p_{k+1}$. First, we still determine a point j that is closest to i in $\mathcal{F}^p_k$. Since a plane requires at least three non-collinear points, we still need to determine two points, where l is taken from the same laser beam sweep as j, m is the closest neighbour of i in the two sweeps consecutive to the sweep of j, and (j, l, m) constitutes the matching of i. We search for potential matching for i in $\mathcal{F}^e_k$ and $\mathcal{F}^p_k$ instead of $\mathcal{P}_k$ and utilise a 3D KD tree in the storage form, which greatly improves the efficiency of the algorithm [31].

III. Lidar Odometry and Loop-Closure Detection

In this section, we design a complete lidar SLAM system. In general, the system is discussed with respect to two parts: the front-end and the back-end. The front-end utilises the preprocessed data mentioned above to estimate the frame-to-frame pose transformation, called lidar odometry; the back-end performs lidar mapping and pose optimisation. We focus on the lidar loop-closure detection method based on the normal distribution transform (NDT) histogram to eliminate accumulated location errors, which is not available in many lidar SLAM systems.

A. Frame-to-Frame Pose Estimation

According to the method in Section II-C, we can obtain the features extracted from two adjacent frames of the point cloud and the matching correspondence between them. Unlike visual SLAM, which directly acquires image data, $\mathcal{P}_k$ is empty at the beginning of a sweep and is constantly being filled in. Therefore, the pose estimation process is recursive. For the i-th feature point in the k-th sweep, this point is reprojected to the beginning time t_k of the sweep according to (1). Let the reprojected edge and plane feature sets be $\widetilde{\mathcal{F}}^e_k$ and $\widetilde{\mathcal{F}}^p_k$, respectively. The goal of the optimization problem is to minimize the distance between the current point cloud and the corresponding point in the target point cloud. The cost function is defined in the form of the point-to-plane distance [32]:

$$C\left(T^L_k\right) = \sum_{i \in \widetilde{\mathcal{F}}_k, j \in \widetilde{\mathcal{F}}_{k-1}} (\vec{n}^T_{p_{(k-1,j)}} (T^L_k p^L_{(k,i)} - p^L_{(k-1,j)}))^2 \tag{3}$$

where $\widetilde{\mathcal{F}}_k$ and $\widetilde{\mathcal{F}}_{k-1}$ represent all projected feature points in the $k-th$ and $(k-1)th$ sweeps, respectively; that is, $\widetilde{\mathcal{F}}_k = [\widetilde{\mathcal{F}}^e_k \cup \widetilde{\mathcal{F}}^p_k]$ and $\widetilde{\mathcal{F}}_{k-1} = [\widetilde{\mathcal{F}}^e_{k-1} \cup \widetilde{\mathcal{F}}^p_{k-1}]$. In addition, $\vec{n}$ is the unit normal vector at j. Consider T^L_k to impose a disturbance $\exp(\xi)$ on the prior pose estimate $\hat{T}^L_k$, where ξ is the Lie algebraic representation of the pose. Then, formula (3) can be rewritten as:

$$C\left(\xi\right) = \sum_{i \in \widetilde{\mathcal{F}}_k, j \in \widetilde{\mathcal{F}}_{k-1}} (\vec{n}^T_{p_{(k-1,j)}} (\exp(\xi) \hat{T}^L_k p^L_{(k,i)} - p^L_{(k-1,j)}))^2 \tag{4}$$

We define the residual at each point as:

$$e_{i,j}\left(\xi\right) = \vec{n}^T_{p_{(k-1,j)}} (\exp(\xi) \hat{T}^L_k p^L_{(k,i)} - p^L_{(k-1,j)}) \tag{5}$$

To minimise the residual $e_{i,j}\left(\xi\right)$, we need to calculate the Jacobian matrix J of $e_{i,j}\left(\xi\right)$ relative to ξ. Due to space limitations, we directly provide the following conclusion:

$$J = \frac{\partial e_{i,j}\left(\xi\right)}{\partial \xi} = \begin{pmatrix} \vec{n}^T_{p_{(k-1,j)}} & \left(p^L_{(k-1,j)} \times n_{p_{(k-1,j)}}\right)^T \end{pmatrix} \tag{6}$$

Then, the optimal solution of T^L_k can be solved iteratively by the Levenberg-Marquardt (L-M) method:

$$T^L_k - (J^T J + \lambda \mathrm{diag}(\mathrm{J^T J}))^{-1} \mathrm{J^T e_{i,j}}\left(0\right) \Rightarrow \mathrm{T^L_k} \tag{7}$$

where λ is the Lagrange multiplier. As Equation (7) iterates to convergence, we can estimate the pose of the lidar in the k-th sweep, but an inevitable error occurs in each estimate. With the accumulation of this error, the accuracy of the lidar odometry continues to decline or ultimately drifts. In response to this problem, we introduce the loop-closure detection module to correct the cumulative error to achieve globally consistent optimal pose estimation.

B. NDT-Histogram-Based Loop-Closure Detection

1) Feature description: The NDT algorithm maps the point cloud to a smooth surface to represent it, and uses a set of local probability density functions (PDFs) for description, where each PDF describes the shape of a part of the surface. Specifically, as each frame of the point cloud is received, it is partitioned into several cells, and each cell is represented by two parameters, which are the mean μ and covariance Σ:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_i^L \tag{8}$$

$$\Sigma = \frac{1}{N-1}\left(\sum_{i=1}^{N}(p_i^L-\mu)(p_i^L-\mu)^T\right) \tag{9}$$

where N is the number of points in a cell and p_i^L corresponds to their coordinates in the lidar coordinate system.

Note that the cell is a fixed spatial area, with new points entering continuously and old points leaving; the mean and covariance should also be continuously updated. When the number of received point cloud frames reaches a certain value, a keyframe is formed; in this paper, we specify 100 frames, and a keyframe is formed. We calculate the features of each cell in the keyframe, including the shape and direction; that is, eigenvalue decomposition of the covariance matrix of each cell is performed [33]:

$$\Sigma V = V\Lambda \tag{10}$$

where $V = (v_1, v_2, v_3)$ is the eigenvector of the covariance matrix and Λ is a diagonal matrix. The main diagonal elements are arranged in descending order of eigenvalues, denoted as λ_1, λ_2 and λ_3. We divide cells into three categories in general, as shown in Fig. 5.

1) Line shape: if the eigenvalues satisfy $\lambda_1 \gg \lambda_2, \lambda_3$, then we identify the cell as a line, and the direction is $C_d = v_1$.
2) Plane shape: if the eigenvalues satisfy $\lambda_3 \ll \lambda_1, \lambda_2$, then we identify the cell as a plane, and the direction is $C_d = v_3$.
3) If a cell does not meet one of the two type definitions above, then it is judged as a non-characteristic cell in this paper.

According to the feature direction, we further divide the plane cells into sub-classes [34]. Assuming that a set of straight lines that pass through the origin and are evenly distributed exists, we classify the plane cells as the line closest to their feature direction v_3, and the index for the sub-classes is:

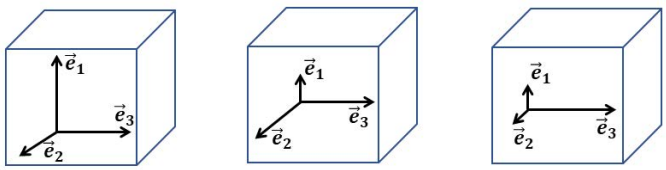


Fig. 5. Three different types of cells: spherical (left), plane (middle), and line (right).

$$\kappa = \arg\min_i \delta(v_3, l_i) \tag{11}$$

where l_i is a line that pass through the origin and $\delta(v_3, l_i)$ is a function that returns the distance between point v_3 and line l_i.

2) Rotation invariance: Since our feature descriptors are defined by direction, we need to apply rotation invariance to ensure the robustness of the feature at different angles. In particular, we wish to determine two peaks. The primary peak can be used for alignment along the Z-axis, and the secondary peak can be used for alignment along the Y-axis. Both peaks should be found in the plane direction. The rotation matrix used to achieve the desired alignment is calculated by Rodrigues' formula [35]:

$$R_z = \cos\theta_z I + (1-\cos\theta_z)\vec{n}_z\vec{n}_z^T + \sin\theta_z\vec{n}_z^\wedge \tag{12}$$

where $\vec{n}_z = \vec{l_i} \times [0,0,1]^T$, $\theta_z = -\arccos(\vec{l_i}\cdot[0,0,1]^T)$, $\vec{l_i}$ the unit vector corresponds to l_i, I is identity matrix and the symbol $\wedge$ represents the transformation of a vector to an antisymmetric matrix. R_z realises the alignment of the primary peak along the z-axis; similarly, for the secondary peak, R_y is designed to rotate the secondary peak to the yz plane:

$$R_y = \cos\theta_y I + (1-\cos\theta_y)\vec{n}_y\vec{n}_y^T + \sin\theta_y\vec{n}_y^\wedge \tag{13}$$

where $\vec{n}_y = [0,0,1]^T$ and $\theta_y = -\arccos(R_z\vec{l_i}\cdot[0,1,0]^T)$.

For all plane feature directions v_3 and line feature directions v_1, we apply the computed rotation matrix R_yR_z to unify their directions to make feature directions invariant to arbitrary rotation of the keyframe. We take the positive direction of the X-axis as the reference direction and let the feature direction $C_d = [C_{d_x}, C_{d_y}, C_{d_z}]$, so the directions of the two types of cell features are transformed as follows:

$$C_d = \text{sign}(C_{d_x})\cdot C_d \tag{14}$$

where $\text{sign}(\cdot)$ is a function that satisfies $\text{sign(x)} = -[\text{x}<0] + [\text{x}>0]$. The pitch angle α and yaw angle γ of the feature direction in the lidar coordinate system L are computed by:

$$\alpha = \sin^{-1}(C_{d_z}) \tag{15}$$

$$\gamma = \tan^{-1}\left(\frac{C_{d_y}}{C_{d_x}}\right) \tag{16}$$

We construct a 2D histogram matrix for plane and line features with resolutions of 3°, 4°, and 5°. Each element in the matrix represents the number of sub-intervals of this type of

cell divided by the corresponding angle in the $0-180^\circ$ interval. Specifically, for the plane feature, the i-th row, j-th column of the 2D histogram matrix with a 3° resolution is the number of cells satisfying the following Euler angle relationships:

$$\begin{aligned} j \times 3^\circ \leq \alpha < (j+1) \times 3^\circ \\ i \times 3^\circ \leq \gamma < (i+1) \times 3^\circ \end{aligned} \tag{17}$$

Similarly, the 2D histograms with resolutions of 4° and 5° correspond to (17), where the step length is changed to 4 and 5. By comparing the difference in the 2D histograms between the current and candidate keyframes, whether a loop is detected can be determined, that is, whether the vehicle has reached a scene that it passed before. The advantage of this multi-resolution mode is that we can first compare low-resolution histograms with less calculation and then perform calculations under the high-resolution conditions when the similarity of the low-resolution histograms meets the threshold, which improves the efficiency while ensuring the accuracy and robustness of loop-closure detection.

3) Difference measure: As the keyframe forms, the calculated histogram matrix is implicitly saved in the loop-closure detection thread. Compared with the frame-to-frame pose estimation thread, the frequency of loop-closure detection is lower, generally 2 Hz, which guarantees that the system can run online. For a newly added keyframe, we calculate the difference between it and all historical keyframes [36]:

$$\sigma(H_c, H_h) = \frac{\sum_{i,j}(H_c(i,j) - \bar{H}_c)(H_h(i,j) - \bar{H}_h)}{\sqrt{\sum_{i,j}(H_c(i,j) - \bar{H}_c)^2 (H_h(i,j) - \bar{H}_h)^2}} \tag{18}$$

where $\sigma(H_c, H_h)$ is the normalized cross-correlation indicator that measures the difference between the two histograms H_c and H_h, where the former is the histogram of the current keyframe, the latter is the histogram of historical keyframes in the database compared with H_c, and $\bar{H}_c$ and $\bar{H}_h$ are the means of H_c and H_h respectively. Once the value of $\sigma(H_c, H_h)$ exceeds the threshold, the algorithm determines that a loop has been detected, thereby correcting the pose and map.

4) Graph-based correction: Once a loop is detected, the system performs global map and pose optimisation. The specific optimisation method is graph-based, the nodes in the graph are labelled as states (generally poses) to be optimised at each sampling time, and the edges between the nodes are labelled as the constraints between the states. The optimisation process maximises the joint probability distribution in terms of the graph model by adjusting the value of each node to reach the global optimal state, as shown in Fig. 6:

$$\{\mathbf{x}, \mathbf{l}\}^* = \operatorname{argmax}(\mathbf{x}_0) \underbrace{\prod P(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k)}_{=1} \prod \mathrm{P}(\mathbf{z}_\mathrm{k} | \mathbf{x}_\mathrm{i}, \mathbf{l}_\mathrm{j}) \tag{19}$$

where $\mathbf{x}$ is the system state, corresponding to the pose in this paper, and $\mathbf{l}$ is the landmark, corresponding to the point returned by lidar. $\{\mathbf{x}, \mathbf{l}\}^*$ is the global optimal state, and $P(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k)$ is the conditional probability that the state changes from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ when the control input is $\mathbf{u}_k$; since we have not considered the problem of control input, the term is equal to 1. $P(\mathbf{z}_k | \mathbf{x}_i, \mathbf{l}_j)$ is the probability of observing the landmark $\mathbf{l}_j$ as $\mathbf{z}_k$ in the $\mathbf{x}_i$ state.

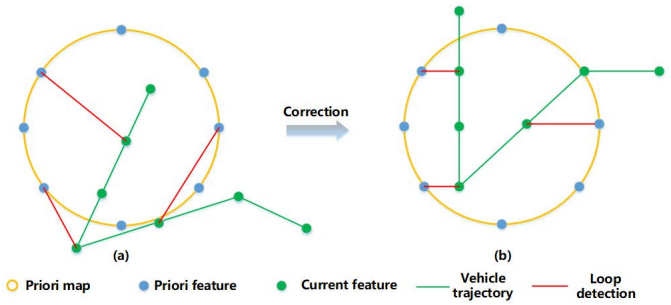


Fig. 6. Global optimisation of the pose and map; (a) and (b) are the map and vehicle trajectory before and after correction, respectively. In (a), the system uses the method described in this chapter to detect a loop (red line) and at the same time adds constraints to the factor graph; in (b), the factor graph performs global optimisation, which can be regarded as performing registration between the current features and the previous features, thereby correcting the map and optimising the pose information related to the two keyframes.

As an important module of the SLAM system, loop-closure detection is integrated into many mature optimisation libraries, including GTSAM [37], G2O [38], Ceres [39], etc. We use GTSAM in this paper, which implements incremental optimisation effects based on the factor graph and only re-optimises the state related to the current moment without optimising the entire factor graph when a new node enters the graph. In this way, this architecture reduces unnecessary calculations and guarantees the real-time performance of loop-closure detection.

C. Mapping

Compared with the odometry module, the mapping module has lower requirements for real-time performance, so it only runs once in each lidar sweep cycle. After the $k-th$ scan is complete, we can obtain the current point cloud map and the corresponding pose, denoted $\{\mathcal{Q}_k, T_k^L\}$. With the start of the sweep of frame $k+1$, the pose between t_k and t_{k+1} is calculated recursively; meanwhile, the point cloud released by the odometry module is restored to the three-dimensional space to form a new point cloud map according to the backprojection mean. Since a large number of identical scan points exist between two adjacent frames of point cloud data, the mapping module also needs to perform feature extraction and matching to ensure map consistency. The specific method is to perform principal component analysis (PCA) [40] on the point cloud clusters around the feature point to determine the corresponding edge or plane; once the type of feature point is determined, the subsequent algorithm is consistent with that in II-C.

IV. Results and Discussion

To verify the positioning accuracy and loop-closure detection performance of the designed system for intelligent vehicles, we conducted simulation tests using the KITTI dataset

TABLE I
PARAMETERS

Description	Symbol	Value
Consecutive points for feature extraction	S	10
Features in current sweep	$\mathcal{F}_k^e, \mathcal{F}_k^p$	2,4
Features in last sweep	$\mathbb{F}_{k-1}^e, \mathbb{F}_{k-1}^p$	50,100
Number of cells	N_{cell}	$100 \times 100 \times 100$
Threshold of the line cell	λ_2/λ_1	≤ 0.1
Threshold of the plane cell	λ_1/λ_2	≤ 0.1
Number of lines dividing space	l_i	9
Line histogram similarity threshold $(3°, 4°, 5°)$	σ_{line}	$0.5, 0.6, 0.7$
Plane histogram similarity threshold $(3°, 4°, 5°)$	σ_{plane}	$0.7, 0.8, 0.9$
Frequency of pose estimation		10Hz
Frequency of mapping		2Hz

[41]. The simulation platform was a Nuvo-6180GC industrial computer (E3-1275 3.60 GHz×8 CPU, 32 RAM) with an Ubuntu 16.04 environment. The algorithm was implemented in C++, and ROS [42] was used to realise communication between nodes. The remaining parameters in our system are listed in Table *I*.

A. KITTI Dataset

The public dataset KITTI provides a variety of automatic driving sensor data (such as images, lidar point clouds and GPS/IMU information) invoking a vehicle equipped with four colour video cameras, two greyscale cameras, a Velodyne HDL-64E lidar and a combined GPS/IMU inertial navigation system in 'Road', 'City', 'Residential', 'Campus' and 'Person' scenes with 22 sequences named "00-21". Additionally, KITTI contains the standard trajectories with reliable navigation parameters, called the GroundTruth, available to users, which includes the complete pose information of the vehicle in 3D space, providing a basis for us to verify and test the accuracy of the system. Note also that the GroundTruth provided by the KITTI dataset does not contain timestamp information, which means that we cannot achieve frame-to-frame alignment under system time. We use the evo toolbox to compare the estimated trajectory with the real trajectory and compare the absolute error on each axis through a customised script [43].

B. Positioning Performance Without Loop-Closure Detection

We first test the system we designed on the 00 sequence of the KITTI dataset, which is a representative working condition. The vehicle travelled more than 3,724 meters, passed through multiple intersections (with a large number of vehicles and pedestrians on the road), and passed through repeated locations many times before returning to the vicinity of the starting point.

The SLAM system usually estimates the 6-DOF pose, which is the motion in 3D space. However, we rarely care about the vertical motion of a vehicle while driving on a road. Moreover, for the sake of intuition, the performances of SLAM systems are diagrammatically given in the form of 3D to 2D projections, which are characterised by absolute positioning errors (APEs), as shown in Fig. 7. In Fig. 7, the dotted lines represent the GroundTruth trajectory, the blue full line is the trajectory estimated by the lidar SLAM system in this paper, and the yellow full line is the trajectory estimated by ICP SLAM [44]. "GT" in the legend denotes GroundTruth, "Ours_noloop" means our system when loop-closure detection is disabled, and "ICP_noloop" means ICP SLAM when loop-closure detection is disabled. To show the detailed relationship between the trajectories, a green dashed line is used to circle the partially enlarged trajectory.

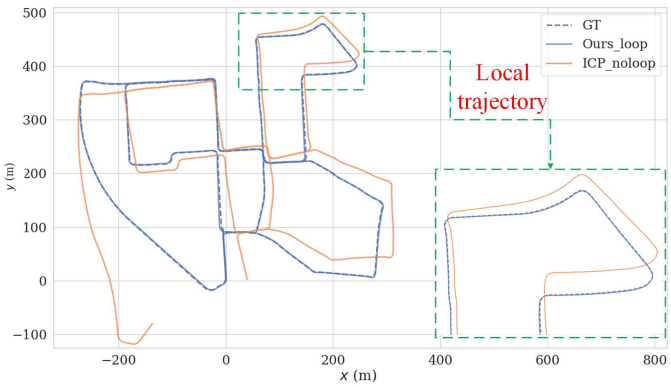


Fig. 7. Comparison of vehicle trajectories without the loop-closure detection module on 00 sequences of the KITTI dataset.

From a global perspective, the orange full line is obviously farther from the dotted line than the blue full line, which means that the lidar SLAM system we designed offers higher positioning accuracy than other methods. The position in the lower left corner of the figure corresponds to the vehicle returning to the starting point. ICP SLAM has deviated far from the GroundTruth, which is attributed to the low positioning accuracy of its odometry module and lack of correction in loop-closure detection. From a partial point of view, our SLAM system result is also closer to the GroundTruth in the enlarged trajectory, indicating that good accuracy can still be achieved without the loop-closure detection module.

Although the estimated trajectory is consistent with the GroundTruth trend, for intelligent driving tasks, such accuracy is not sufficient to support the normal operations of planning, decision making and other functions. Fig. 8 further depicts the positioning error in Fig. 7, where the error of ICP SLAM is large and therefore not shown. As shown in Fig. 8, the overall error on the three axes shows a divergent trend, which is attributed to the accumulated error characteristic of the lidar odometry module, and no loop-closure detection module is

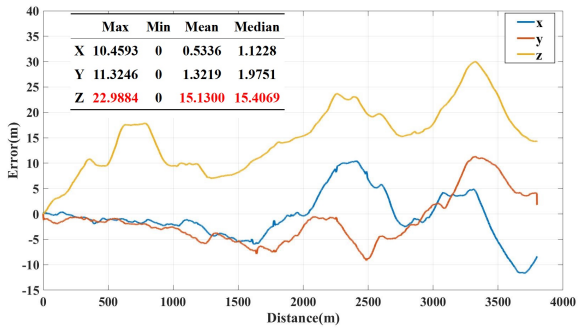


Fig. 8. Tri-axis absolute positioning error of our system without the loop-closure detection module and some of the statistics on the 00 sequences of the KITTI dataset.

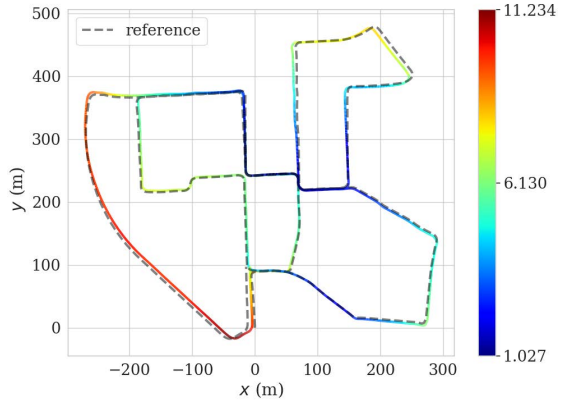
employed to provide timely corrections. Among these errors, the average value of the X-axis error is 0.5 meters, and that of the Y-axis error is 1.3 meters, whereas the Z-axis error has reached 15 meters, far exceeding the average error. Considering that we focus on the lateral and longitudinal positioning accuracy of vehicles when driving on a road, the vertical direction is often of reference value in specific situations such as overpasses. Therefore, in the following experiments, we ignore the influence of the Z-axis error on the overall accuracy and evaluate the performance of the system otherwise. Based on this, we provide a trajectory comparison ignoring the influence of the Z-axis error in Fig. 9.

In Fig. $9(a)$, the dotted lines represent the GroundTruth, and the coloured lines represent the trajectories estimated by our lidar SLAM; the closer the colour of the lines is to red, the greater the APE. The colour of the line on the right side of the figure is closer to blue; that is, the error is relatively small. When the colour of the line on the left approaches red, the vehicle is driving back to the original point, which is due to the accumulation of lidar odometry errors leading to the larger positioning error at this time. In Fig. $9(b)$, we display the distribution of APEs and list some statistics, including the mean, root mean squared error (RMSE), median and standard deviation (STD). The average positioning error is approximately 5.5 meters, or approximately 6 meters, as shown in Fig. 8.

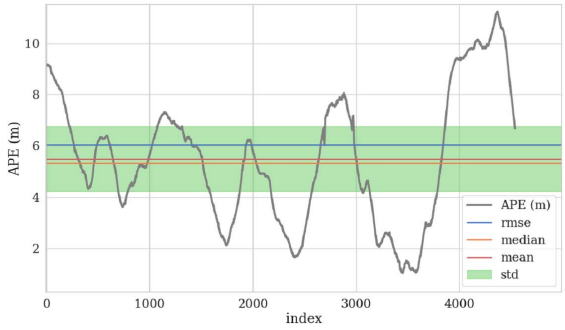
C. Positioning Performance With Loop-Closure Detection

The left side of Fig. 10 shows the APE of our lidar SLAM system with a loop-closure detection module, and four obvious loops are marked in the figure, that is, the vehicle repeatedly passes a position. The four columns of pictures on the right side of Fig. 10 correspond to the visual scene (top) and point cloud (below) of the four labelled points A, B, C, and D. Due to the influence of the loop detection module, the APE of the vehicle's trajectory has been significantly reduced.

According to the image of the scene, many vehicles are parked on the side of the city street, and the intelligent vehicle encounters traffic vehicles and shuttle pedestrians, which will exert some negative effects on the positioning and loop-closure detection module. Except for some large corners where the error is slightly abrupt, the errors in the straight sections and



(a) The error map of our system when ignoring the influence of the Z-axis



(b) The APE of our system when ignoring the influence of the Z-axis

Fig. 9. The error map and APE of our system when ignoring the influence of the Z-axis.

the part where the loop is detected are essentially on the order of tens of centimetres, which proves that our lidar SLAM offers excellent robustness. This is mainly due to the feature extraction that filters out a large number of redundant points and the fact that the multi-resolution histogram can be used to perform deeper mapping of the scene.

To show the effect of the point cloud histogram more intuitively, we use part A in Fig. 10 to provide a detailed explanation. We judge whether loop-closure detection occurs according to the threshold set in Table I; that is, once the two detected point clouds meet the above similarity threshold, the system determines that a loop is detected and performs global optimisation. Fig. 11 shows the multi-resolution point cloud histogram of the plane features before and after passing through position A twice (selecting the more informative plane feature histogram as the display). The left side of the figure shows the tensor-based visual information of the histogram, while the right side corresponds to the $3°$, $4°$, and $5°$ resolution histogram matrix from bottom to top, and the pixel colours represent the corresponding values. The data distribution in the two sets of histograms is similar enough to meet the threshold we set, and a loop is identified.

Furthermore, we show some results on the 05 sequence of the KITTI dataset in Fig. 12, where ICP SLAM is still used for comparison with the designed SLAM system. In particular, "ICP_loop" and "Ours_loop" correspond to ICP SLAM and

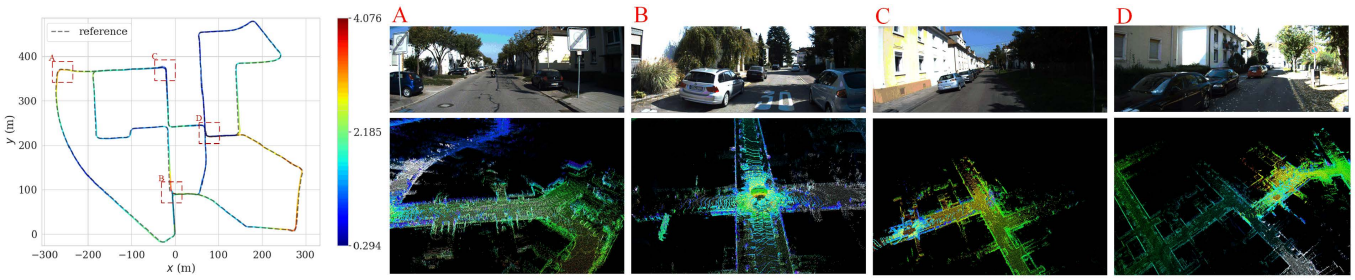
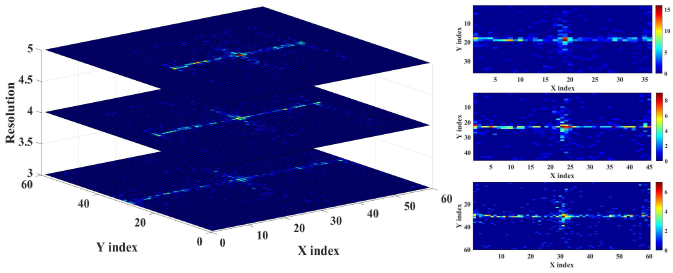
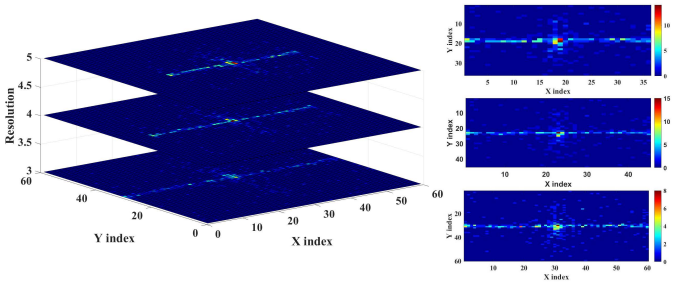


Fig. 10. Positioning performance with the loop-closure detection module and view of images and the point cloud.



(a) Point cloud histogram when the vehicle passes point A for the first time



(b) Point cloud histogram when the vehicle passes point A for the second time

Fig. 11. Our multi-resolution point cloud histogram.

TABLE II
% ERROR PER $100m$ OF SOME STATE-OF-THE-ART SYSTEMS ON THE 01 TO 05 SEQUENCES OF THE KITTI DATASET

Sequence	ICP SLAM[42]	LOAM[16]	LEGO-LOAM[18]	Our
01(Highway)	2.14	1.43	1.08	0.92
02(Urban+country)	1.49	0.92	0.81	0.52
03(Country)	2.01	0.86	0.99	0.81
04(Country)	1.78	0.71	0.69	0.45
05(Urban)	2.52	0.57	0.68	0.40

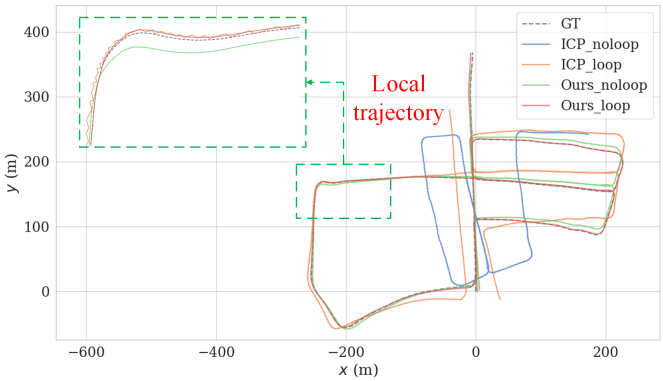


Fig. 12. The performance of different systems on the 05 sequence in the KITTI dataset.

our system when loop detection is enabled, respectively. The blue trajectory and GroundTruth exhibit a severe deviation in scale, which is attributed to the excessive positioning error that makes the iterative calculation accuracy diverge. Therefore, the trajectory of ICP SLAM without the loop-closure detection module is interrupted, and it fails to complete the global tracking. In the local trajectory, our system with loop-closure detection is close to the GroundTruth, whereas the trajectory of ICP SLAM with loop-closure detection is jagged and gradually diverges from the GroundTruth as the vehicle moves (along the negative direction of the Y-axis). This occurs because ICP SLAM uses a gradient descent method without considering the influence of the optimisation step. In this paper, we solve this problem through the L-M method to make the trajectory more accurate and smooth. Finally, we show the performance of our SLAM system and other advanced SLAM systems on multiple dataset sequences in Table II to illustrate the superiority of our method.

D. Mapping performance

Fig. 13 shows the mapping result for the 00 sequence. Our lidar SLAM can provide precise positioning while building a clear outline point cloud map, which can be used for navigation and obstacle avoidance. The loop-closure detection effect with map correction is shown in Fig. 14. Fig. $14(a)$ and Fig. $14(c)$ correspond to the point cloud maps at A and B in Fig. 13 before loop-closure detection correction, respectively, and Fig. $14(b)$ and Fig. $14(d)$ correspond to the point cloud maps after loop-closure detection correction. Intuitively, the corrected maps have better consistency; that is, no gaps or ghosts occur in the maps, which reflects the superior performance of the loop-closure detection system in this paper.

TABLE III
COMPUTING TIME OF EACH MODULE

	Feature extraction	Pose estimation	Histogram construct	Difference measure	Mapping
Computing time	$17.3ms$	$20.7ms$	$0.97ms$	$9\mu s$	$436ms$

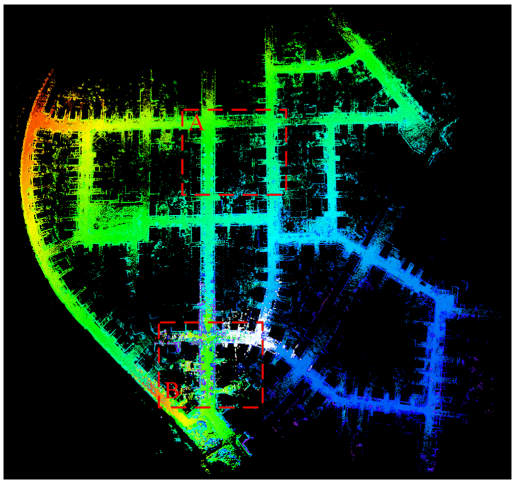
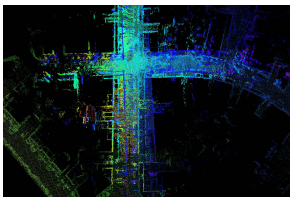
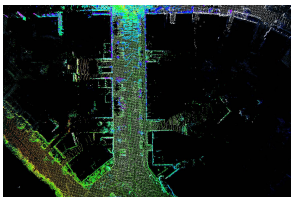


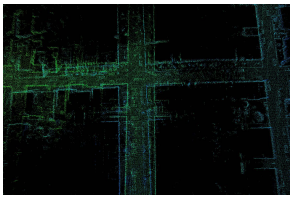
Fig. 13. Mapping results of our SLAM system on 00 sequence of KITTI dataset.



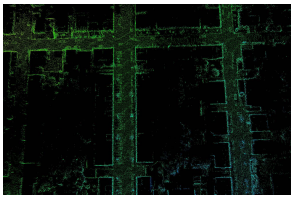
(a) Point A before loop-closure detection



(b) Point A after loop-closure detection



(c) Point B before loop-closure detection



(d) Point B after loop-closure detection

Fig. 14. Map continuity performance before and after correction by loop-closure detection module.

E. Running time

We evaluated the calculation time of the designed lidar SLAM system in blocks, including feature extraction, frame-to-frame pose estimation, computation of the point cloud histogram, similarity evaluation and mapping. The computing time of each module is given in Table III, where we can see that the total time consumption of the pose estimation module is approximately 0.038 seconds, which fully meets the frequency requirement of 10 Hz; the total time consumption of the loop-closure detection and mapping module is approximately 0.437 seconds, which is also sufficient to meet the frequency requirement of 2 Hz. In summary, we declare that our system is sufficient for running online.

V. Conclusion

In this paper, a fully automatic lidar SLAM system that integrates motion compensation and ground information removal was designed and implemented. We developed a loop-closure detection algorithm based on point cloud histograms in multi-resolution mode, which can accurately detect the closed loop of the vehicle during driving and obtain accurate pose information and map texture through global optimisation. We conducted a comprehensive test on the public dataset KITTI, and the results show that our SLAM system supported by the loop-closure detection module can provide reliable positioning information for intelligent vehicles and that the real-time map can be used for planning, obstacle avoidance and other functions.

Since the mechanical lidar obtains the point cloud by rotating the laser beam in the lateral direction, vertical features exhibit a low correlation, which causes our system to poorly estimate the vertical pose transformation. The next step is to add an inertial measurement unit to achieve tightly coupled information fusion, which can improve the positioning accuracy and robustness of the system.

Acknowledgements

This work is supported by the National Nature Science Foundation of China (U19A2069, 61790563), the Project of the Science and Technology Department Department of Jilin Province (20200401088GX, 20200501011GX), and the Project of the National Development and Reform Commission of Jilin Province (2019C036-5).

References

[1] B. Soualmi, C. Sentouh, J. Popieul, and S. Debernard, "Automation-driver cooperative driving in presence of undetected obstacles," *engineering practice*, vol. 24, pp. 106–119, 2014.

[2] Y. Huang, H. Wang, A. Khajepour, H. He, and J. Ji, "Model predictive control power management strategies for hevs: A review," *Journal of Power Sources*, vol. 341, pp. 91–106, 2017.

[3] C.-F. Lin, J.-C. Juang, and K.-R. Li, "Active collision avoidance system for steering control of autonomous vehicles," *IET Intelligent Transport Systems*, vol. 8, no. 6, pp. 550–557, 2014.

[4] W. Wen, G. Zhang, and L.-T. Hsu, "Object-detection-aided gnss and its integration with lidar in highly urbanized areas," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 3, pp. 53–69, 2020.

[5] Y. Chen, S. Huang, and R. Fitch, "Active slam for mobile robots with area coverage and obstacle avoidance," *IEEE/ASME Transactions on Mechatronics*, 2020.

[6] D. Tang, Q. Fang, L. Shen, and T. Hu, "Onboard detection-tracking-localization," *IEEE/ASME Transactions on Mechatronics*, 2020.

[7] J. Duan, H. Shi, D. Liu, and H. Yu, "Square root cubature kalman filter-kalman filter algorithm for intelligent vehicle position estimate," *Procedia engineering*, vol. 137, no. 11, pp. 267–276, 2016.

[8] O. Guclu and A. B. Can, "Integrating global and local image features for enhanced loop closure detection in rgb-d slam systems," *The Visual Computer*, vol. 36, no. 6, pp. 1271–1290, 2020.

[9] A. R. Memon, H. Wang, and A. Hussain, "Loop closure detection using supervised and unsupervised deep neural networks for monocular slam systems," *Robotics and Autonomous Systems*, vol. 126, p. 103470, 2020.

[10] S.-p. Li, T. Zhang, X. Gao, D. Wang, and Y. Xian, "Semi-direct monocular visual and visual-inertial slam with loop closure detection," *Robotics and Autonomous Systems*, vol. 112, pp. 201–210, 2019.

[11] H. Jo, H. M. Cho, S. Jo, and E. Kim, "Efficient grid-based rao–blackwellized particle filter slam with interparticle map sharing," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 714–724, 2018.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] T. Nguyen, G. K. Mann, A. Vardy, and R. G. Gosine, "Ckf-based visual inertial odometry for long-term trajectory operations," *Journal of Robotics*, vol. 2020, 2020.

[14] L. Jin, H. Zhang, and C. Ye, "Camera intrinsic parameters estimation by visual–inertial odometry for a mobile phone with application to assisted navigation," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1803–1811, 2020.
[15] K. Eckenhoff, P. Geneva, and G. Huang, "Closed-form preintegration methods for graph-based visual–inertial navigation," *The International Journal of Robotics Research*, vol. 38, no. 5, pp. 563–586, 2019.
[16] W. Liu, S. Wu, Z. Wu, and X. Wu, "Incremental pose map optimization for monocular vision slam based on similarity transformation,"*Sensors*, vol. 19, no. 22, p. 4945, 2019.
[17] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.
[18] M. Tomono, "Loop detection for 3d lidar slam using segment-group matching," *Advanced Robotics*, pp. 1–15, 2020.
[19] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.
[20] T. Röhling, J. Mack, and D. Schulz, "A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data," in*2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 736–741, IEEE, 2015.
[21] R. Dubé, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 339–355, 2020.
[22] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
[23] J. Lin and F. Zhang, "Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," *arXiv preprint arXiv:1909.06700*, 2019.
[24] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: 3d segment mapping using data-driven descriptors," *arXiv preprint arXiv:1804.09557*, 2018.
[25] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, 2014.
[26] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.
[27] M. D. Kim and J. Ueda, "Dynamics-based motion deblurring improves the performance of optical character recognition during fast scanning of a robotic eye," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 491–495, 2018.
[28] F. Neuhaus, T. Koß, R. Kohnen, and D. Paulus, "Mc2slam: Real-time inertial lidar odometry using two-scan motion compensation," in *German Conference on Pattern Recognition*, pp. 60–72, Springer, 2018.
[29] W. Song, Y. Yang, M. Fu, F. Qiu, and M. Wang, "Real-time obstacles detection and status classification for collision warning in a vehicle active safety system," *IEEE Transactions on intelligent transportation systems*, vol. 19, no. 3, pp. 758–773, 2017.
[30] A. Ortiz-Gonzalez, V. Kober, V. Karnaukhov, and M. Mozerov, "Algorithm for the design of a three-dimensional map of the environment with a depth camera," *Journal of Communications Technology and Electronics*, vol. 65, no. 6, pp. 690–697, 2020.
[31] F. Zhang, Y. Gao, and L. Xu, "An adaptive image feature matching method using mixed vocabulary-kd tree," *Multimedia Tools and Applications*, vol. 79, no. 23, pp. 16421–16439, 2020.
[32] A. Handa, "Simplified jacobians in 6-dof camera tracking," tech. rep., Technical report, University of Cambridge, 2014.
[33] W. Wen, L.-T. Hsu, and G. Zhang, "Performance analysis of ndt-based graph slam for autonomous vehicle in diverse typical driving scenarios of hong kong," *Sensors*, vol. 18, no. 11, p. 3928, 2018.
[34] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 892–914, 2009.
[35] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, "Real-time 6-dof multi-session visual slam over large-scale environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1144–1158, 2013.
[36] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," *arXiv preprint arXiv:1909.11811*, 2019.
[37] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," tech. rep., Georgia Institute of Technology, 2012.
[38] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pp. 9–13, 2011.
[39] S. Agarwal, K. Mierle, *et al.*, "Ceres solver—a large scale non-linear optimization library," 2019.
[40] L.-H. Chen and C.-C. Peng, "A robust 2d-slam technology with environmental variation adaptability," *IEEE Sensors Journal*, vol. 19, no. 23, pp. 11475–11491, 2019.
[41] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
[42] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.
[43] M. Grupp, "evo: Python package for the evaluation of odometry and slam.." https://github.com/MichaelGrupp/evo, 2017.
[44] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, pp. 1948–1955, April 2019.

Qingyu Meng received the M.S degree from Northeast Electric Power University, Jilin, China, in 2019. He is now a Ph.D. in the Department of Control Science and Engineering in the School of Communication Engineering, Jilin University, Changchun, China. His research interests include vision and lidar SLAM, and state prediction in intellitent vehicles.

Hongyan Guo (M'17) received the Ph.D. degree from the Jilin University, Changchun, China, in 2010. She joined Jilin University, Changchun, China, in 2011. From 2014, she is a associate professor with the Department of Control Science and Engineering, Jilin University. In 2017, she is a visiting scholar at Cranfield University, UK. Her current research interests include path tracking and stability control of autonomous vehicles and vehicle states estimation.

Xiaoming Zhao received the B. S. degree from the Jilin University , Changchun, China, in 2017. Now he is a master in Jilin University, Changchun, China. His current research interests is target trajectory prediction.

Dongpu Cao M'08) received the Ph.D. degree from Concordia University, Canada, in 2008. He is currently an Associate Professor at University of Waterloo, Canada. His research focuses on vehicle control and intelligence, automated driving and parallel driving, where he has contributed more than 130 publications and 1 US patent. He received the ASME AVTT'2010 Best Paper Award and 2012 SAE Arch T. Colwell Merit Award. Dr. Cao has been serving on the SAE International Vehicle Dynamics Standards Committee and a few ASME, SAE, IEEE technical committees.

Hong Chen (M'02, SM'12) received the Ph.D. degree from the University of Stuttgart, Stuttgart, Germany, in 1997. She joined Jilin University of Technology, Changchun, China, in 1986, where she became an Associate Professor in 1998 and has been a Professor since 1999. Her current research interests include model predictive control, optimal and robust control, and applications in process engineering and mechatronic systems.