LiDAR-IMU Tightly-Coupled SLAM Method Based on IEKF and Loop Closure Detection

Huimin Pan, Dongfeng Liu, Jingzheng Ren, Tianxiong Huang, and Huijun Yang

***Abstract*—Simultaneous Localization and Mapping (SLAM) technology based on LiDAR can achieve real-time robot positioning and establish environmental maps in unknown environments. LiDAR odometry can achieve accurate pose estimation in short distances or small-scale environments, but the accuracy will decrease with the accumulation of errors. At the same time, under scenes with insufffcient structural features, point cloud-based LiDAR SLAM will show degradation phenomena, leading to failures in localization and mapping. We propose a loop closure detection method based on fusion of Inertial Measurement Unit (IMU) and LiDAR information to reduce system cumulative errors and solve environment degradation problems. Firstly, by fusing LiDAR key feature and IMU information, an odometry calculation method is proposed based on Iterative Extended Kalman Filtering (IEKF) to improve the accuracy of FAST-LIO initial pose. Then in loop closure detection, by considering geometric and intensity information simultaneously, a new scan context global descriptor is constructed from dedistortion feature points of front-end IMU to enhance the accuracy of loop closure detection for original point cloud descriptors. Finally, GTSAM is used for global optimization and GPS constraint is introduced to reduce trajectory drifts, obtaining globally consistent trajectories and mapping. Compared with existing LiDAR SLAM on KITTI dataset and self-collected dataset, proposed method has smaller trajectory errors on KITTI sequences, which reduced by 15% than baseline FAST-LIO2, and average time consumption of loop closure detection is reduced by about 50% than SC-ALOAM, and mapping drifts is reduced, which enhanced mapping accuracy and robustness while ensuring global consistency of constructed maps.**

***Index Terms*—Inertial measurement unit (IMU), iterative extended Kalman filtering (IEKF), LiDAR simultaneous localization and mapping (SLAM), loop closure detection.**

Manuscript received 29 August 2023; revised 25 December 2023; accepted 10 January 2024. Date of publication 9 February 2024; date of current version 26 March 2024. This work was supported in part by the Foundation of Key Research and Development Program of Shanxi province under Grant 2023-YBNY-229 and Grant 2024NC-YBXM-197, and in part by Undergraduate Training Program for Innovation and entrepreneurship plan under Grant 202210712189. *(Huimin Pan and Dongfeng Liu contributed equally to this work.) (Corresponding author: Huijun Yang.)*

Huimin Pan and Jingzheng Ren are with the College of Information Engineering, Northwest A&F University, Yangling 712100, China.

Dongfeng Liu and Tianxiong Huang are with the Shenzhen Agricultural Science and Technology Promotion Center, Nanshan 518000, China (e-mail: liudfh@163.com; 842848500@qq.com).

Huijun Yang is with the College of Information Engineering, Northwest A&F University, Yangling 712100, China, also with the Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture and Rural Affairs, Northwest A&F University, Yangling 712100, China, and also with the Shanxi Engineering Research Center of Agricultural Information Intelligent Perception and Analysis, Northwest A&F University, Yangling 712100, China (e-mail: yhj740225@nwafu.edu.cn).

Digital Object Identifier 10.1109/JSTARS.2024.3357536

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is a crucial technology for unmanned autonomous operations, enabling effective environment perception and real-time localization and mapping of mobile robots. SLAM methods can be categorized as LiDAR SLAM and visual SLAM based on the sensors used [1]. Visual SLAM, relying on cameras, faces uncertainties in outdoor environments due to light and environmental changes, impacting its accuracy and robustness [2]. In contrast, LiDAR SLAM offers significant advantages in measurement accuracy, range, and resistance to environmental interference [3]. Consequently, LiDAR SLAM excels in mapping and localization, finding widespread applications indoors and outdoors.

With 360° horizontal field of view, accurate distance measurement, insensitivity to environmental lighting and optical textures, as well as effective perception in dark environments, 3-D LiDAR finds wide application in diverse scenarios [4]. However, LiDAR-based methods encounter degradation in scenarios like long corridors, tunnels, and open roads [5]. This degradation leads to significant errors in LiDAR SLAM state estimation and mapping overlapping and intersecting. At the same time, in large-scale or complex environments, there are high requirements for data processing, real-time performance, and stability of the system, while a single sensor may not provide enough information to ensure high-precision SLAM. In view of the above problems, this article studies the laser SLAM scheme based on multisensor fusion. Inertial measurement unit (IMU) data and laser mileage count are fused in a tightly coupled way to improve the accuracy of positioning map construction. The research shows that the fusion of these two sensor data can improve the accuracy and robustness of vehicle positioning and navigation. However, due to the differences between 3-D LiDAR and IMU in frequency, precision and coordinate system, it is difficult to learn from each other, calibrate, and synchronize multisensor parameters.

On the one hand, loop closure detection presents a challenging issue in LiDAR SLAM as it prevents significant deviations in state estimation over time [6], which crucial for accurate and effective back-end mapping. LiDAR-based loop closure detection methods are typically classified into local and global descriptors [3], [7]. Global descriptors, proposed to address the instability of matching local key points, calculate similarity between the current frame and others to detect loop closures [8]. Dube et al. [9] proposed SegMatch, a segment-based scene recognition method, as a loop closure detection module for global optimization of vehicle pose [10], [11]. However, due to

additional point cloud processing, it increases the computational cost of loop closure detection.

On the other hand, incorporating back-end nonlinear global optimization, such as GTSAM [12] and Levenberg–Marquardt (LM), trajectory is optimized to produce high-precision map on a larger scale. The main methods for back-end global optimization include filter-base and graph-based approaches. Due to computational complexity and limited robustness of former, graph-based optimization dominates the field and with the following four categories:

1) least squares;
2) random gradient descent;
3) relaxation iteration;
4) manifold iteration, which can conveniently construct and solve large-scale factor graph problems.

Aiming at inaccuracy of positioning in existing LiDAR SLAM [3], [10], [13], especially in large-scale environments, the data information of multiple sensors is huge and complex, requiring a large amount of computing resources and effective data processing schemes, and for long-term system operation, it may lead to sensor drift, cumulative errors and other problems, in this article, by introducing loop closure detection and back-end optimization, FAST-LIO2 [14] is improved to reduce the cumulative errors and improve the accuracy of localization and mapping. The main contributions are as follows.

1) A tightly coupled localization method is proposed based on LiDAR key features and IMU. Curvature-based feature extraction scheme is proposed to enhance the spatial constraints of planar feature points. The initial pose estimation of LiDAR-Inertial Odometry is then obtained by an IEKF to improve the accuracy of the localization.
2) A loop closure detection method based on height-intensity scan context (HISC) global descriptor is proposed. By integrating geometric and intensity information, HISC global descriptor is designed and an adaptive distance threshold is introduced to correct the accumulated errors over time in LiDAR-IMU system, improve the trajectory and mapping accuracy of large-scale environment.
3) A feature matching is proposed for calculating the pose transformation between pairs of loop-point clouds, to reduce the computational cost of pose estimation. In addition, we introduce a trajectory global optimization method based on GTSAM, which constructs a factor graph using iSAM2 and incorporates GPS factors to impose absolute position constraints, thereby reducing mapping trajectory drift.

The rest of this article is organized as follows. In Section II, we discuss current research works of LiDAR SLAM methods. Section III describes the details of the proposed method, including LiDAR inertial odometry, loop closure detection, global optimization and mapping. The experiments are presented in Section IV. And the experimental results are discussed in Section V. Finally, Section VI concludes this article.

II. Related Work

At present, the research of LiDAR SLAM has been very extensive. In this section, we focus on analyzing the research status of LiDAR SLAM, loop closure detection, and tightly coupled of LiDAR and IMU sensors.

A. LiDAR SLAM

The development of SLAM can be categorized into three stages: early (1986–2010), middle (2010–2014), and modern (2014–present). In the early stage, methods based on Kalman filter (KF) held prominence, followed by the emergence of methods based on extended Kalman filter (EKF) and particle filter, among others. In 2010, the introduction of Karto SLAM [15] marked the arrival of optimization-based SLAM, which demonstrated better performance compared to filter-based methods. In 2014, Zhang Ji [5] proposed Lidar Odometry and Mapping in Real-time (LOAM), which marked the basic maturity of LiDAR SLAM, and separated the localization and mapping into two algorithms. One performs high-frequency odometry with low accuracy (localization), while the other operates at a lower frequency to perform point cloud matching and registration (mapping and odometry correction). By combining these, a high-precision and real-time LiDAR odometry system is achieved. However, it also has certain limitations, such as reduced optimization efficiency in feature-rich environments due to the lack of a closed-loop detection function [16]. As a result, enhancing the robustness of SLAM has become a new research focus, leading to continuous improvements in LOAM's performance.

Based on Ceres Solver [17], Wang et al. [1] proposed Fast LiDAR Odometry and Mapping (F-LOAM), which improves the frame matching accuracy of LOAM by removing scan-to-scan pose estimation, retaining only scan-to-map pose optimization, and adopting a noniterative two-level distortion compensation method to reduce computational complexity and cost. According to the official evaluation criteria of KITTI dataset [18], although F-LOAM achieved the best accuracy in pose estimation, it lacks loop closure detection, leading to accumulated errors and reducing effectiveness in large-scale scenarios [3]. Shan [19] proposed Light-weight and Ground-Optimized LiDAR Odometry and Mapping (LeGO-LOAM), which incorporated loop closure detection and lightweight and ground optimization on feature extraction, by combining ICP and Euclidean distance to identify loop closure points. Compared to other methods, it achieved similar or better accuracy. However, the loop closure detection in LeGO-LOAM is unstable and may occasionally result in detection errors or missed identifications [13].

B. Optimization Methods for LiDAR SLAM

Many researchers have attempted to enhance the performance of SLAM systems by incorporating additional modules. Among them, loop closure detection is a critical module in SLAM. Over time, LiDAR odometry accumulates errors, leading to drift problems in long-term navigation and mapping. To address this issue in large-scale scenarios, integrating a loop closure detection module in the back-end can correct odometry drift [1]. Kim et al. [8] proposed a method using scan context global descriptor to reduce the dimensionality of point cloud data frames, which are stored in a 2-D matrix, the rows represent the distance of the divided ground regions (i.e., bins) from the LiDAR center,

and the columns represent the angle of these regions relative to the x-direction. By using the global descriptor to calculate the similarity between current frame and others, loop closure detection results can be obtained (e.g., [13] and [20]).

However, existing 3-D loop closure detection methods often use local or global geometric descriptors while neglecting the intensity information in point clouds. To address this limitation, a global descriptor called intensity scan context (ISC) was proposed based on F-LOAM, which incorporates both geometry and intensity information [21]. Furthermore, to enhance the efficiency of loop closure detection, a two-stage hierarchical rerecognition method was introduced. Optimized-sc-f-loam [3] is also a loop closure detection method based on F-LOAM. Unlike [21], it employs a feature-point matching method instead of using the original LiDAR point cloud to calculate the pose transformation between the current frame and the closed-loop frame, resulting in reduced computation time. In the back-end, global optimization is based on GTSAM, and an adaptive distance threshold is utilized for more precise loop closure detection.

Integrating an IMU is another approach to enhance the performance of LiDAR SLAM system. The incorporation of IMU can significantly improve the accuracy and robustness of the LiDAR odometry and compensate motion distortions in LiDAR scans. With the rapid development of robotics technology, localization and mapping techniques are being applied to increasingly complex and dynamic scenarios [22]. The combination of LiDAR and IMU offers advantages such as high accuracy, fast speed, and immunity to environmental lighting conditions, which makes it widely employed [23].

At present, LiDAR and IMU fusion can be categorized into the loosely-coupled and the tightly-coupled. The former deal with two sensor information separately to infer their motion constraints which are fused later (e.g., [5], [19], [24], and [25]). The authors in [26] proposed combining IMU measurements with attitude estimates, which obtained from a LiDAR-based Gaussian particle filter and a prebuilt map. Generally speaking, the loosely-coupled fusion method is computationally effective and has good real-time performance. However, the motion constraints between LiDAR and IMU may lead to information loss [4], making it difficult to ensure accuracy in high-speed motion or degraded scenes.

Another approach is the tightly-coupled method that directly fuses LiDAR information and IMU data by joint optimization, which can be further divided into optimization-based [27], [28] and extended KF-based approaches [4], [29]. The authors in [30] proposed a graph optimization based tightly-coupled approach, which combined prior information from the LiDAR-IMU odometry and optimization method based on rotational constraints to further refine the odometry pose, it can obtain a globally consistent and robust mapping trajectory. However, constraints and batch optimization in constructing local map windows are time consuming and have poor real-time performance. Shan et al. proposed LIO-SAM [28], which is based on incremental smoothing [31] that introduced a global factor graph consisting of LiDAR odometry factors, IMU preintegration factors, GPS factors, and loop closure factors, and it achieved high-precision global consistency in mapping and motion estimation for the robot. The authors in [29] proposed a LiDAR-IMU fusion odometry framework with high computational efficiency and good robustness, which adopted the iterative extended Kalman Filter similar to [4], and used forward propagation to predict the state as well as backward propagation to correct motion distortion in LiDAR scanning, and proposed a new Kalman gain calculation formula to reduce the computational complexity. But the system lacked back-end optimization and only worked in small environment. The authors in [14] inherited FAST-LIO and introduced a new data structure called the incremental k-d tree (ikd-tree) [32], which supports incremental update (e.g., point insertion and deletion) as well as dynamic rebalancing, significantly reducing the amount of computation. This approach improves the accuracy and robustness of odometry and map generation. However, similar to [29], this method is also limited to small-scale environments.

In summary, 3-D LiDAR faces challenges in long-range environments with sparse structural features, leading to localization and mapping failures and inadequate construction of surrounding environment maps. The tightly-coupled of LiDAR and IMU incurs high computational costs, significant time consumption, and large memory usage, which making it difficult to ensure real-time and accurate operation of the system. Therefore, improving the precise localization and mapping of 3D point cloud maps in large-scale environments, both indoors and outdoors, is of great significance for applications such as mobile robotics and autonomous driving.

III. Methodology

The proposed LiDAR SLAM method consists of three parts: 1) LiDAR-IMU odometry; 2) loop closure detection; and 3) global optimization and mapping, as in Fig. 1, to enable real-time six-DOF state estimation of the robot and establish a globally consistent map. To address the asynchronization issue between the LiDAR and IMU, the LiDAR point cloud data is first preprocessed. Considering challenges as the large quantity of raw LiDAR points, slow data processing speed, and computational complexity, local neighbor that based feature extraction is given to compute curvature and extracted feature points, including planar points and edge points. Next, the motion model provided by the IMU is employed to predict the state of feature points, which is fused and optimized using iterative Kalman Filter to estimate the robot's pose and position during motion. Simultaneously, the IMU backward propagation is employed to mitigate the motion distortion of the LiDAR, to obtain more accurate and robust localization results. Subsequently, a loop closure detection method based on the HISC is proposed to reduce the localization drift caused by cumulative errors in the LiDAR-IMU odometry. Then, an adaptive distance threshold is introduced to reduce the probability of wrong loop closure detection and the similarity between the current frame and historical frames is computed based on proposed global descriptor, which is constructed by the distortion-corrected feature points. Finally, by adding odometry factors, loop closure factors, and GPS factors to the iSAM2 factor graph, a global optimization-based GTSAM is presented to correct the accumulated drift errors and generate globally consistent trajectories and maps.

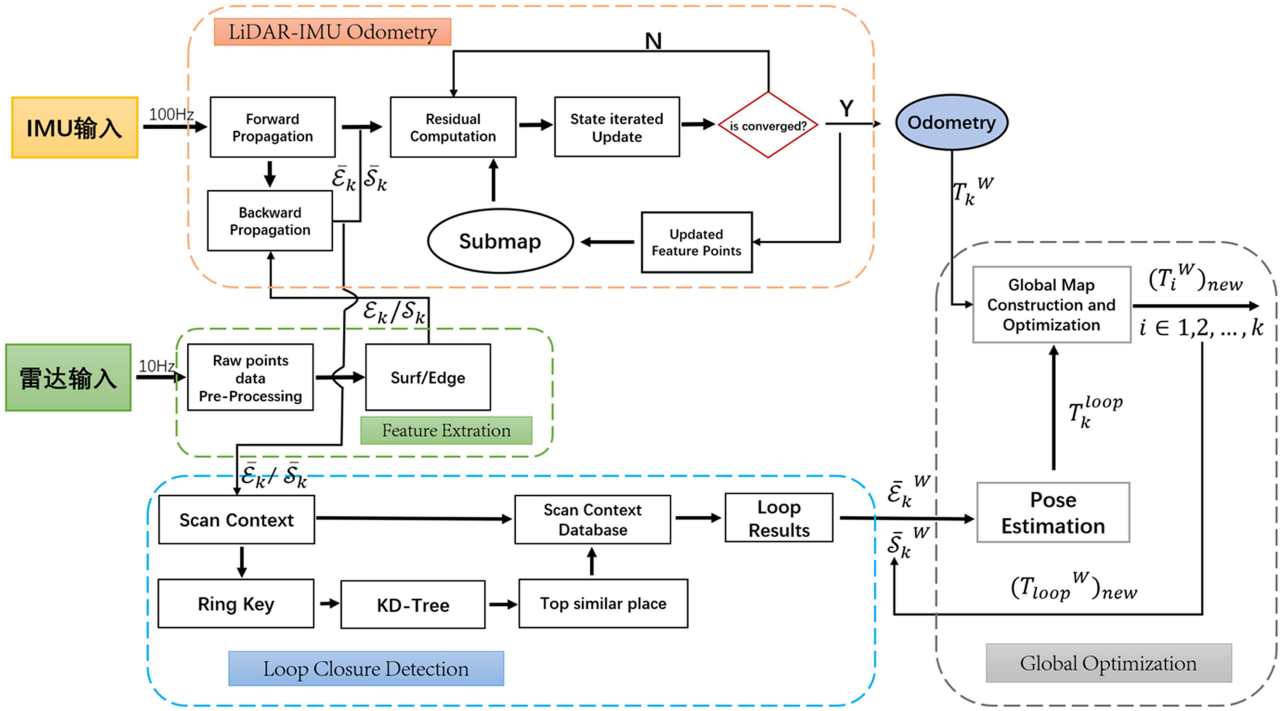


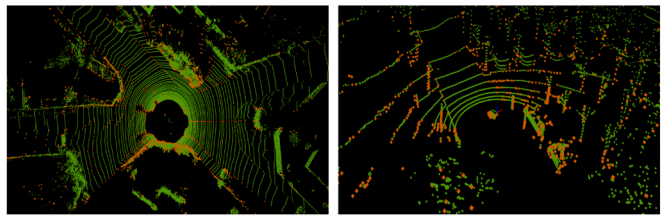
Fig. 1. System framework overview.

A. LiDAR-Inertial Odometry

1) Feature Extraction: Due to the large number of points contained in a single scan of LiDAR, traditional point cloud matching methods, such as ICP and NDT, have low computational efficiency. Therefore, it is necessary to process the raw point cloud data $\mathcal{P}_k$. In practical applications, feature-based matching methods have shown stronger robustness and efficiency compared to raw point cloud matching methods. To improve the speed, matching accuracy, and efficiency of the algorithm, this article extracts plane feature points and edge feature points [5] from the raw point cloud while discarding noisy or less significant points. As the 3-D mechanical rotating LiDAR scan produces sparser points in the vertical direction and denser point clouds in the horizontal direction, the algorithm focuses on points on the plane for each scan and calculates the local plane curvature σ as follows:

$$\sigma_k^{(m,n)} = \frac{1}{|S_k^{(m,n)}|} \sum_{\mathrm{p}_k^{(m,j)} \in S_k^{(m,n)}} ||\mathrm{p}_k^{(m,j)} - \mathrm{p}_k^{(m,n)}|| \quad (1)$$

where $k \in \mathrm{Z}^+$ is the scan number; $\mathcal{P}_k$ is the point cloud obtained from the kth scan, each point is $\mathrm{p}_k^{(m,n)}$, $m \in [1, M], n \in [1, N]$; $S_k^{(m,n)}$ is the local point cloud set formed by the neighboring points of $\mathrm{p}_k^{(m,n)}$ in the horizontal direction (along the clockwise and anticlockwise, respectively, choose five points as a local adjacent point cloud); $|S_k^{(m,n)}|$ is the number of point clouds in the local point cloud set. Compared to the local search method, $S_k^{(m,n)}$ can be obtained more quickly according to the index number n of the points, reducing the computational cost. For planar points, such as feature points on walls, they have smaller smoothness values. On the other hand, edge points have larger smoothness values. Therefore, for a given scan point cloud $\mathcal{P}_k$, point with larger σ value is edge point $(\sigma > \sigma_e)$, while point with smaller σ value is planar point $(\sigma < \sigma_s)$. $\mathcal{E}_k$ is the set of edge points and $\mathcal{S}_k$ is the set of planar points. $\mathbb{F}_k = \{\mathcal{E}_k, \mathcal{S}_k\}$ is the set of feature points. The planar and edge points extracted using this method on the KITTI dataset and self-collected dataset are shown in Fig. 2.



(a) (b)

Fig. 2. Extract feature points (yellow indicates edge points, green indicates planar points). (a) KITTI dataset. (b) Self-collected dataset.

2) Tightly-Coupled LiDAR-IMU With IEKF: To address the issue of localization and mapping failures in scenes with insufficient structural features, this article inherits the FAST-LIO2 [14] by integrating IMU and LiDAR with IEKF. This approach aims to handle degraded and weak-textured scenes and improve system stability. It mainly includes forward propagation based on IMU measurements and iterative updating based on LiDAR scanning.

1) Forward Propagation Based on IMU

State definitions as follows:

$$x = [p^T, v^T, R^T, b_\omega^T, b_a^T, g^T]^T \quad (2)$$

where p, R are the position and attitude of IMU, v is the velocity, g is the gravity vector, and b_ω and b_a are the IMU bias.

Assume the optimal state estimate after fusing the last (i.e., $k-1$th) LiDAR scan is $\bar{x}_{k-1}$ with covariance matrix $\bar{\mathrm{P}}_{k-1}$. The forward propagation is performed upon the arrival of an IMU measurement. The state and covariance are propagated as follows:

$$\hat{x}_{i+1} = \hat{x}_i \boxplus (\Delta t \mathrm{f}(\hat{x}_i, \mu_i, 0)); \hat{x}_0 = \bar{x}_{k-1} \tag{3}$$

$$\hat{\mathrm{P}}_{i+1} = \mathrm{F}_{\tilde{x}_i} \hat{\mathrm{P}}_i {\mathrm{F}_{\tilde{x}_i}}^T + \mathrm{F}_{w_i} Q_i {\mathrm{F}_{w_i}}^T; \hat{\mathrm{P}}_0 = \bar{\mathrm{P}}_{k-1} \tag{4}$$

where i represent the index of IMU data, x_i, $\hat{x}_i$, and $\bar{x}_i$ represent the ground-true, propagated, and updated value of x_i, respectively. Based on the definition of $\boxplus$ and $\boxminus$ in [29], the continuous state transition equation $x_{i+1} = x_i \boxplus (\Delta t \mathrm{f}(x_i, \mu_i, w_i))(w_i = 0)$ is discretized using the IMU sampling interval Δt to obtain (3). And in (4), the matrix $\hat{\mathrm{P}}_{i+1}$ is the covariance of the error of $\tilde{x}_{i+1} = x_{i+1} \boxminus \hat{x}_{i+1} = \mathrm{F}_{\tilde{x}_i}\tilde{x} + \mathrm{F}_{w_i} w_i$. Then, Q_i is the covariance of the noise w_i and the matrix $\mathrm{F}_{\tilde{x}_i}$ and F_{w_i} are computed as follows:

$$\mathrm{F}_{\tilde{x}_i} = \frac{\partial (x_{i+1} \boxminus \hat{x}_{i+1})}{\partial \tilde{x}_i} | \tilde{x}_i = 0, w_i = 0 \tag{5}$$

$$\mathrm{F}_{\tilde{x}_i} = \begin{bmatrix} \mathrm{I} & \mathrm{I}\Delta t & 0 & 0 & 0 & 0 \\ 0 & \mathrm{I} & -\hat{R}\lfloor a_m - b_a \rfloor_\wedge & 0 & -\hat{R}\Delta t & \mathrm{I}\Delta t \\ 0 & 0 & \mathrm{Exp}(-(\omega_m - b_\omega)\Delta t) & \text{-I}\Delta t & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} \end{bmatrix} \tag{6}$$

$$\mathrm{F}_{w_i} = \frac{\partial (x_{i+1} \boxminus \hat{x}_{i+1})}{\partial w_i} | \tilde{x}_i = 0, w_i = 0 \tag{7}$$

$$\mathrm{F}_{w_i} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathrm{I}\Delta t & 0 & 0 & 0 \\ 0 & -\mathrm{I}\Delta t & 0 & 0 \\ 0 & 0 & -\mathrm{I}\Delta t & 0 \\ 0 & 0 & 0 & -\mathrm{I}\Delta t \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{8}$$

The forward propagation continues until reaching the end time of the current kth scan where the propagated state and covariance are denoted as $\hat{x}_k$, $\hat{\mathrm{P}}_k$, respectively, serving as the predicted state for the forward propagation and then continues to the next scan.

2) Residual Computation

Assume the estimate of state x_k at the current kth iterate update is ${\hat{x}_k}^{\mathrm{k}}$, when $k = 0$, ${\hat{x}_k}^{\mathrm{k}} = \hat{x}_k$, where $\hat{x}_k$ is the predicted state from the propagation in (3). Then, using the coordinate transformation defined in (9), the scan points ${\mathrm{p}_j}^L$ of the LiDAR, after distortion correction using IMU [29], are projected onto the global coordinate system.

$${\hat{\mathrm{p}}_j^{\mathrm{k}}}^W = {\hat{\mathrm{T}}_{I_k}^{\mathrm{k}}}^W {\hat{\mathrm{T}}_{L_k}^{\mathrm{k}}}^I {\mathrm{p}_j}^L \tag{9}$$

where L, I, W represent LiDAR coordinate system, IMU coordinate system, and global coordinate system, respectively; $\hat{\mathrm{T}}_{L_k}^{\mathrm{k}}$ and $\hat{\mathrm{T}}_{I_k}^{\mathrm{k}}$ are the state transition matrix of coordinate transformation.

For each LiDAR feature point, we assume that the nearest plane or edge defined by its neighboring feature points in the map represents its true location. The residual is defined as the distance between the estimated global coordinates ${\hat{\mathrm{p}}_j^{\mathrm{k}}}^W$ of the feature point and the nearest plane (or edge) in the map. Let μ_j be the normal vector of the corresponding plane or the direction of the corresponding edge, and let ${\mathrm{q}_j}^W$ be a point on the plane or edge. Then, the residual is defined as follows:

$${\mathrm{z}_j}^{\mathrm{k}} = {\mu_j}^T ({\hat{\mathrm{p}}_j^{\mathrm{k}}}^W - {\mathrm{q}_j}^W) \tag{10}$$

$${\mathrm{z}_j}^{\mathrm{k}} = \lfloor \mu_j \rfloor_\wedge ({\hat{\mathrm{p}}_j^{\mathrm{k}}}^W - {\mathrm{q}_j}^W) \tag{11}$$

where (10) is the residual of plane feature points, and (11) is the residual of edge feature points.

Substituting (9) into (10) and (11) yields the measurement model $0 = \mathrm{h}_j(x_k, {n_j}^L) = G_j({\hat{\mathrm{T}}_{I_k}^{\mathrm{k}}}^W {\hat{\mathrm{T}}_{L_k}^{\mathrm{k}}}^I {\mathrm{p}_j}^L - {\mathrm{q}_j}^W)$ (when the point is a planer point $G_j = {\mu_j}^T$, otherwise $G_j = \lfloor \mu_j \rfloor_\wedge$. Moreover, approximating the measurement equation by its first order approximation made at ${\hat{x}_k}^{\mathrm{k}}$ leads to (12).

$$\begin{aligned} 0 &= \mathrm{h}_j\left(x_k, {n_j}^L\right) \\ &= \mathrm{h}_j\left({\hat{x}_k}^{\mathrm{k}} \boxplus {\tilde{x}_k}^{\mathrm{k}}, {n_j}^L\right) \\ &\simeq \mathrm{h}_j\left({\hat{x}_k}^{\mathrm{k}}, 0\right) + {\mathrm{H}_j}^{\mathrm{k}} {\tilde{x}_k}^{\mathrm{k}} + \mathrm{v}_j \\ &= {\mathrm{z}_j}^{\mathrm{k}} + {\mathrm{H}_j}^{\mathrm{k}} {\tilde{x}_k}^{\mathrm{k}} + \mathrm{v}_j \end{aligned} \tag{12}$$

where ${\tilde{x}_k}^{\mathrm{k}} = x_k \boxminus {\hat{x}_k}^{\mathrm{k}}$ (or equivalently, $x_k = {\hat{x}_k}^{\mathrm{k}} \boxplus {\tilde{x}_k}^{\mathrm{k}}$); ${\mathrm{z}_j}^{\mathrm{k}} = \mathrm{h}_j({\hat{x}_k}^{\mathrm{k}}, 0)$ is called the residual; ${\mathrm{H}_j}^{\mathrm{k}}$ is the Jacobin matrix of $\mathrm{h}_j({\hat{x}_k}^{\mathrm{k}} \boxplus {\tilde{x}_k}^{\mathrm{k}}, {n_j}^L)$ with respect to ${\tilde{x}_k}^{\mathrm{k}}$, evaluated at zero; $\mathrm{v}_j \in (0, \mathrm{R}_j)$ is due to the raw measurement noise.

3) State Iterative Update

The propagated state $\hat{x}_k$ and covariance $\hat{\mathrm{P}}_k$ from (1) impose a prior Gaussian distribution for the unknown state x_k. More specifically, $\hat{\mathrm{P}}_k$ represents the covariance of the error state in (13).

$$\begin{aligned} x_k \boxminus \hat{x}_k &= \left({\hat{x}_k}^{\mathrm{k}} \boxplus {\tilde{x}_k}^{\mathrm{k}}\right) \boxminus \hat{x}_k \\ &= {\hat{x}_k}^{\mathrm{k}} \boxminus \hat{x}_k + {\mathrm{J}}^{\mathrm{k}} {\tilde{x}_k}^{\mathrm{k}} \sim \mathcal{N}\left(0, \widehat{\mathrm{P}}_k\right) \end{aligned} \tag{13}$$

where ${\mathrm{J}}^{\mathrm{k}}$ is the partial differentiation of $({\hat{x}_k}^{\mathrm{k}} \boxplus {\tilde{x}_k}^{\mathrm{k}}) \boxminus \hat{x}_k$ with respect to ${\tilde{x}_k}^{\mathrm{k}}$ evaluated at zero. For the first iteration, ${\hat{x}_k}^{\mathrm{k}} = \hat{x}_k$, ${\mathrm{J}}^{\mathrm{k}} = \mathrm{I}$.

Besides the prior distribution, the state distribution of (14) is computed based on the measurement model derived from (12).

$$-\mathrm{v}_j = {\mathrm{z}_j}^{\mathrm{k}} + {\mathrm{H}_j}^{\mathrm{k}} {\tilde{x}_k}^{\mathrm{k}} \sim \mathcal{N}\left(0, \mathrm{R}_j\right). \tag{14}$$

Combining the prior distribution in (13) and the measurement model from (14) yields the posteriori distribution of the state x_k equivalently represented by ${\tilde{x}_k}^{\mathrm{k}}$ and its maximum a-posteriori estimate (MAP) in (15).

$$\min_{{\tilde{x}_k}^{\mathrm{k}}} \left(\|x_k \boxminus \hat{x}_k\|^2_{\hat{\mathrm{P}}_k} + \sum_{j=1}^{m} \left\| {\mathrm{z}_j}^{\mathrm{k}} + {\mathrm{H}_j}^{\mathrm{k}} {\tilde{x}_k}^{\mathrm{k}} \right\|^2_{\mathrm{R}_j} \right). \tag{15}$$

This MAP problem can be solved by IEKF method as follows:

$$\mathrm{K} = \left({\mathrm{H}}^T \mathrm{R}^{-1} \mathrm{H} + \mathrm{P}^{-1}\right)^{-1} {\mathrm{H}}^T \mathrm{R}^{-1} \tag{16}$$

$$\hat{x}_k{}^{k+1} = \hat{x}_k{}^{k} \boxplus (-\mathrm{K}z_k{}^{k} - (\mathrm{I} - \mathrm{KH})(\mathrm{J}^{k})^{-1}(\hat{x}_k{}^{k} \boxminus \hat{x}_k)) \quad (17)$$

where K is the Kalman gain, and $\hat{x}_k{}^{k+1}$ is the poststate estimation. $\mathrm{H} = [\mathrm{H}_1{}^{T}, \mathrm{H}_2{}^{T}, \ldots, \mathrm{H}_m{}^{T}]$, $\mathrm{R} = \mathrm{diag}[\mathrm{R}_1, \mathrm{R}_2, \ldots, \mathrm{R}_m]$, $\mathrm{P} = (\mathrm{J}^{k})^{-1}\hat{\mathrm{P}}_k(\mathrm{J}^{k})^{-T}$, $z_k{}^{k} = [z_1{}^{k^T}, z_2{}^{k^T}, \ldots, z_m{}^{k^T}]^{T}$.

The above process repeats until convergence, then, the final optimal state and covariance estimates after convergence are given as follows:

$$\bar{x}_k = \hat{x}_k{}^{k+1} \quad (18)$$

$$\bar{\mathrm{P}}_k = (\mathrm{I\text{-}KH})\mathrm{P}. \quad (19)$$

Subsequently, the optimal state estimate $\bar{x}_k$ and covariance estimate $\bar{\mathrm{P}}_k$ are used as inputs for the next scan, and the previous operations are repeated to obtain the estimated LiDAR-Inertial odometry.

B. Loop Closure Detection

To address the issue of accumulated errors in the LiDAR-IMU system during long-term operation, this article introduced a loop closure detection method to reduce positioning drift. The stability is improved by correcting the trajectory and optimizing the map. The Scan Context [8] and Intensity Scan Context [21] algorithms reduced computational complexity by dimensionality reduction, mapping the 3-D point cloud data to a 2-D image. The global descriptor, as shown in Fig. 3(b), is calculated to measure the similarity between point cloud frames and determine if a loop closure is formed. However, it utilized the original point cloud from the LiDAR for constructing the global descriptor, resulting in high computational cost and a high rate of wrong loop closures. By using feature points after front-end IMU distortion corrected, we introduced a global descriptor to improve accuracy in measuring the similarity between point clouds. To better represent the characteristics of the point cloud, a combined consideration of intensity and height information is proposed, known as the HISC global descriptor. To address the limitation of fixed distance threshold, which may result in missing loop closures or false positives, an adaptive distance threshold is proposed to replace it to reduce the probability of false loop closure detection.

1) Global Descriptor of HISC: This article introduced a novel global descriptor, which constructed by height [8] and intensity [21] information. The height information provides an effective summary of the vertical structure of the surrounding buildings, eliminating the need for complex calculations to analyze the point cloud characteristics. The maximum height indicates the visible portion of the surrounding structure. This self-centered visibility allows for the analysis of place characteristics [8], making it useful for loop closure detection and verifying if the system passes through the same location. Moreover, objects exhibit different intensity values, and intensity information serves as a representation of the reflectance of surfaces in the environment. For instance, highly reflective materials like metal plates have higher intensity values, while concrete surfaces have lower values. Therefore, intensity information can effectively serve as a feature to assist laser odometry in location identification.

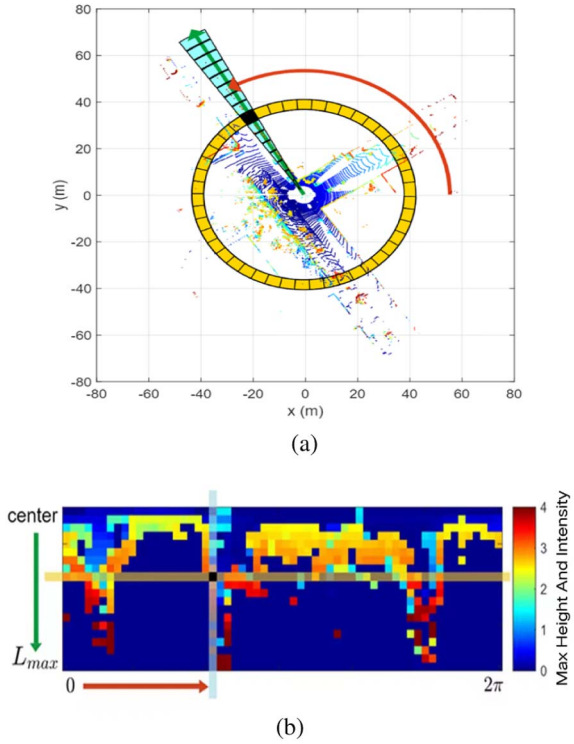


Fig. 3. Using the top view of a point cloud from a 3-D scan (a), we partition ground areas into bins, that are divided according to azimuth (from 0 to 2π within the LiDAR frame) and radial (from center to maximum sensing range) directions. The yellow area as a ring (N_r), the cyan area as a sector (N_s), and the black-filled area as a bin. Scan Context is a matrix as in (b). The ring and sector described in (a) are represented by the same-colored column and row, respectively, in (b). In this article, we use the maximum Height and Intensity of points in a bin.

Assuming the LiDAR scan contains n points, denotes as $P = \{\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_n\}$ and the intensity value is η. $[x, y, z]$ represents the spatial position coordinates of each point. And in Cartesian coordinate system, each point is represented as $\mathrm{p}_k = [x_k, y_k, z_k, \eta_k]$ (where k is the kth point in the point cloud P). According to the method in [21], we first divide the 3-D scan point cloud into the azimuth and radial bin in the sensor coordinates, in an equally spaced manner as shown in Fig. 3(a). N_s, N_r are the number of sectors and rings, respectively. That is to say, if we set the maximum measurement range of the LiDAR sensor to $L_{\max}$ the radial clearance between the rings is $L_{\max}/N_r$, and the center angle of the sector to be equal to $2\pi/N_s$, this article defines $N_s = 60$, $N_r = 20$, each frame point cloud is divided into the subspace $S_{i,j} (i \in \{1, 2, \ldots, N_s\}, j \in \{1, 2, \ldots, N_r\})$ of $N_r \times N_s$. For the same object, the intensity values are consistent, and the number of points in each subdivided subspace is much smaller than the total number of points in a frame. Therefore, it is assumed that the intensity values of LiDAR points are the same within each subspace, combined with the height information of point clouds in each subspace, the sum of the maximum height value and intensity value of all points in each subspace is employed to calculate the value of this subspace, is defined as a new descriptor, as follows:

$$\Omega(i, j) = \max_{\mathrm{p}_k \in S_{i,j}} (\eta_k + z_k). \quad (20)$$

According to the above process, HISC global descriptor is finally expressed as the matrix I of $N_r \times N_s$ in (21).

$$I = (a_{ij}) \in \mathbb{R}^{N_r \times N_s}, a_{ij} = \Omega(i, j). \tag{21}$$

The matrix contains geometric information and intensity information of the environment. The Similarity Score between Scan Contexts and Two-phase Search Algorithm method [8] is employed to calculate the similarity between point cloud pairs according to I, so as to determine whether there is a closed loop.

2) Adaptive Distance Threshold: The original Scan Context method uses a fixed distance threshold to detect and exclude point cloud pairs that have high similarity but are far apart. Setting a small fixed distance threshold may result in undetected loops, while setting a large threshold may detect more false loops. To address this problem, an adaptive distance threshold [3] is introduced to replace it, which effectively detects closed loops and reduces the occurrence of false loop closures.

When a loop closure was detected, the matrix in (21) is computed to obtain a pair of similarity loop-closing points, denoted as ${\mathrm{p}_k}^L$ and ${\mathrm{p}_{\mathrm{loop}}}^W$. Using coordinate transformations, the relative pose transformations of the two-point cloud frames with respect to the world coordinate system, ${T_k}^W$ and ${T_{\mathrm{loop}}}^W$, are calculated. Then, the relative pose transformation between the two-point clouds, $\tilde{T}_k^{\mathrm{loop}}$, is computed as follows:

$$\tilde{T}_k^{\mathrm{loop}} = {T_{\mathrm{loop}}}^{W^{-1}} {T_k}^W. \tag{22}$$

The distance d between the two points is calculated based on the obtained transformations in (23).

$$d = \sqrt{\tilde{T}_k^{\mathrm{loop}}.x^2 + \tilde{T}_k^{\mathrm{loop}}.y^2 + \tilde{T}_k^{\mathrm{loop}}.z^2} \tag{23}$$

where pose ${T_k}^W$ and ${T_{\mathrm{loop}}}^W$ are obtained from the front-end LiDAR-IMU odometry. $\tilde{T}_k^{\mathrm{loop}}$ is the relative pose transformation calculated based on the current and the loop-closed frame.

Then, a linear function is constructed using the number of keyframes k from the front-end odometry to define the threshold d_{thre}, as shown in (24).

$$d_{\mathrm{thre}} = 60 + k/100. \tag{24}$$

If d is greater than the threshold d_{thre}, it indicates that no loop closure has been detected. This method helps to reduce the error rate in loop closure detection, improve the results of global optimization, and enhance the efficiency of the overall LiDAR SLAM algorithm.

C. Global Optimization and Mapping

In order to address the problem of system error accumulation over time and loop closure detection, we proposed back-end optimization to correct system errors. While local optimization only considered the relationship between consecutive frames, it is prone to noise and motion deviations. Therefore, global optimization is introduced, which utilizes global observation data and pose to correct errors. Scan-to-map constraints are constructed for all observed key frames using the feature point matching method [5], forming a large-scale nonlinear least squares problem. The LM algorithm is employed to minimize the error functions, eliminate errors, and correct trajectory drift. In addition, the GTSAM library based on factor graph is employed for global optimization. The iSAM2 algorithm [31] is utilized to incorporate odometry factors, loop closure detection factors, and GPS factors. By using joint optimization, errors are effectively eliminated, enabling the determination of an optimal trajectory and pose for each key frame. This leads to more accurate map and positioning results, enhancing the precision of map construction.

1) Pose Estimation Based on Feature Matching: The feature points ${\mathrm{p}_k}^L$ of the current frame, which are in the LiDAR coordinate system, are transformed to the global coordinate system as ${\mathrm{p}_k}^W$ using the coordinate transformation matrix $({T_{\mathrm{loop}}}^W)_{\mathrm{new}}$ optimized in the previous frame. A submap is then constructed in the global coordinate system, consisting of the loop closure point cloud ${\mathrm{p}_{\mathrm{loop}}}^W$ and the neighboring point clouds. First, (25) is used to project the feature points from the LiDAR coordinate system to the global coordinate system.

$${\mathrm{p}_k}^W = R_{\mathrm{loop}}^W * \left(R_L^{\mathrm{loop}} * {\mathrm{p}_k}^L + t_L^{\mathrm{loop}}\right) + t_{\mathrm{loop}}^W \tag{25}$$

where ${\mathrm{p}_k}^L$ is the feature point in the LiDAR coordinate system. $(R_L^{\mathrm{loop}}, t_L^{\mathrm{loop}})$ represents the external parameter between the LiDAR coordinate system and the carrier coordinate system. $(R_{\mathrm{loop}}^W, t_{\mathrm{loop}}^W)$ represents the transformation of the carrier coordinate system to the global coordinate system.

Then, a feature point matching-based method is used to match the planar points and edge points in the current point cloud frame with the submap. Point-to-line and point-to-plane positional constraints are then constructed as (26) and (27), respectively.

$$d_{\varepsilon_k} = \frac{\left|\left({\mathrm{p}_k}^W - {\mathrm{p}_l}^W\right) \times \left({\mathrm{p}_k}^W - {\mathrm{p}_m}^W\right)\right|}{\left|{\mathrm{p}_l}^W - {\mathrm{p}_m}^W\right|} \tag{26}$$

where l and m represent the two nearest points to point k that are not on the same laser ring. These two points form a line, and d_{ε_k} is the distance from point k to the line lm.

$$d_{S_k} = \frac{\left|\begin{array}{c}\left({\mathrm{p}_k}^W - {\mathrm{p}_l}^W\right) \\ \left({\mathrm{p}_l}^W - {\mathrm{p}_m}^W\right) \times \left({\mathrm{p}_l}^W - {\mathrm{p}_j}^W\right)\end{array}\right|}{\left|\left({\mathrm{p}_l}^W - {\mathrm{p}_m}^W\right) \times \left({\mathrm{p}_l}^W - {\mathrm{p}_j}^W\right)\right|} \tag{27}$$

where l, m, and j are the three points closest to k and not collinear, forming a plane lmj, d_{S_k} is the distance from point k to the plane lmj.

Then, using (26) and (27), nonlinear equations are constructed, and the L-M nonlinear optimization method is used to iteratively solve for the pose transformation matrix ${T_k}^{\mathrm{loop}}$ between the current point cloud frame and the submap, as follows:

$$\min\left\{\sum_{i=1}^{\mathrm{m}} d_\varepsilon\left(\mathrm{p}_i(X)\right) + \sum_{j=1}^{\mathrm{n}} d_{\mathcal{S}}\left(\mathrm{p}_j(X)\right)\right\} \tag{28}$$

where X is the vector composed of the estimated state variables, i.e., $(\mathrm{roll}, \mathrm{pitch}, \mathrm{yaw}, \mathrm{x}, \mathrm{y}, \mathrm{z})^T$. m is the total number of successfully matched edge feature points, while n is the total number of successfully matched plane feature points. $d_\varepsilon(\mathrm{p}_i(X))$ represents the distance from the ith corner point to the line calculated

from the map matching, i.e., the point-to-line distance. Similarly, $d_{\mathcal{S}}(\mathrm{p}_j(X))$ represents the distance from the point to the plane.

Finally, the pose transformation matrix ${T_k}^{\text{loop}}$ is passed to the map construction and optimization module for building the global map.

2) Pose Graph Construction: This module takes the transformation matrix ${T_k}^{\text{loop}}$ obtained from the pose estimation module in (28), as well as the ${T_k}^{W}$ obtained from the LiDAR-IMU odometry module. We introduced GTSAM to construct a factor graph for global optimization. The received odometry poses ${T_k}^{W}$ are added to the corresponding factor graph nodes, and ${T_k}^{\text{loop}}$ is added to the edges connecting the nodes. In addition, we introduced GPS factors to construct absolute position constraints and reduce trajectory drift.

The constructed factor graph is then subjected to global optimization, resulting in optimized poses $({T_i}^{W})_{\text{new}}$, yielding a globally consistent trajectory. Simultaneously, the obtained poses are used to integrate newly acquired map information into the existing map, continuously improving the overall environment map. Finally, a complete 3-D point cloud map is obtained.

IV. Experiment

To evaluate the performance of the proposed algorithm, we conducted tests on both public dataset and self-collected dataset. The KITTI dataset sequences 05, 06, 07, and 09 are used to validate the localization accuracy of the proposed algorithm, and compared with A-LOAM [5] and FAST-LIO2 [14]. In order to verify the time efficiency of the proposed loop closure detection algorithm, that compared with SC-A-LOAM on KITTI sequences. In order to demonstrate the performance of the proposed global descriptor, the proposed method, SC [8] and ISC [21] methods were tested by comparing the RMSE of the generated trajectory on KITTI dataset. For further evaluation, self-collected dataset by VLP-16 LiDAR and N100 IMU are named Park dataset and Street dataset. A comparison of trajectory results was performed against SC-A-LOAM [20], Optimized-sc-f-loam [3], and FAST-LIO2 [14]. Then, comparisons were made with SC-A-LOAM, Optimized-sc-f-loam, LIO-SAM,FAST-LIO2 on the KITTI sequence 07, the outdoor Park dataset, and indoor dataset to validate the effectiveness of the proposed algorithm in mitigating mapping drift. Finally, the CPU usage and memory usage of five different algorithms running three different data sets are analyzed. The experimental data in this article is the best result of multiple measurements.

A. Hardware Platform

The experimental hardware platform in this article consisted of a laptop with an Intel i5-12500H, 16 GB RAM, and an NVIDIA GeForce RTX 3050 graphics card. The operating system used was Ubuntu 20.04. The software framework employed was Robot Operating System, which version is Noetic, and the primary programming language used was C++. The sensors utilized in this study included the VLP-16 mechanical LiDAR and the WHEELTEC N100 IMU.

For the experimental evaluation, the publicly available KITTI dataset, which contains data from outdoor autonomous driving scenarios, was utilized. The dataset includes 22 different outdoor environment sequences, and for 11 of these sequences (00-10), ground-truth trajectories obtained from GPS are provided. Each data sequence consists of grayscale images, color images, and LiDAR point cloud data. In addition, a campus environment data acquisition platform was used. Fig. 4(a) illustrates the configuration of this data collection platform, and Fig. 4(b) shows the environment.

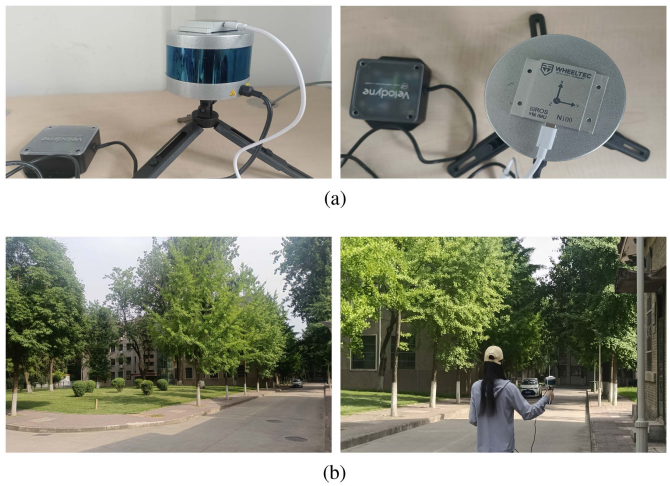


Fig. 4. Self-data collection in the campus environment. (a) Shows the self-data collection platform consisting of LiDAR and IMU. (b) Shows the self-data collection environment and the experimental operating environment.

B. Localization Accuracy

1) KITTI Dataset: In this article, we use the EVO evaluation tool to assess the localization accuracy of the proposed algorithm. Specifically, we compared the performance of the proposed algorithm with the ground-truth on KITTI dataset sequences 05, 06, 07, and 09. These sequences contain loop closures, making them suitable for evaluating the effectiveness of the improved loop closure detection algorithm proposed in this article. The evaluation was based on trajectory visualization, Absolute Trajectory Error (ATE), and Relative Pose Error (RPE).

Fig. 5 illustrates the visual comparison results of the proposed algorithm, A-LOAM, and FAST-LIO2 running on KITTI dataset sequences 05, 06, 07, and 09, along with the ground truth trajectories.

Table I presents the ATE statistics comparing the estimated trajectories obtained by the three algorithms on the sequences of the KITTI dataset with the ground truth trajectories. From the comparison of the statistical values in the table, it can be observed that the proposed algorithm has smaller error values in sequences 05 and 07. The proposed algorithm also achieves the lowest average error (mean) and root mean square error (RMSE) across all four sequences. Compared to the A-LOAM algorithm, the RMSE of the proposed algorithm are reduced by 65.1%, 73.6%, 78.3%, and 61.3% in the four sequences, respectively. Similarly, compared to the FAST-LIO2 algorithm, the RMSE is reduced by 59.4%, 1.16%, 36.4%, and 12.0%.

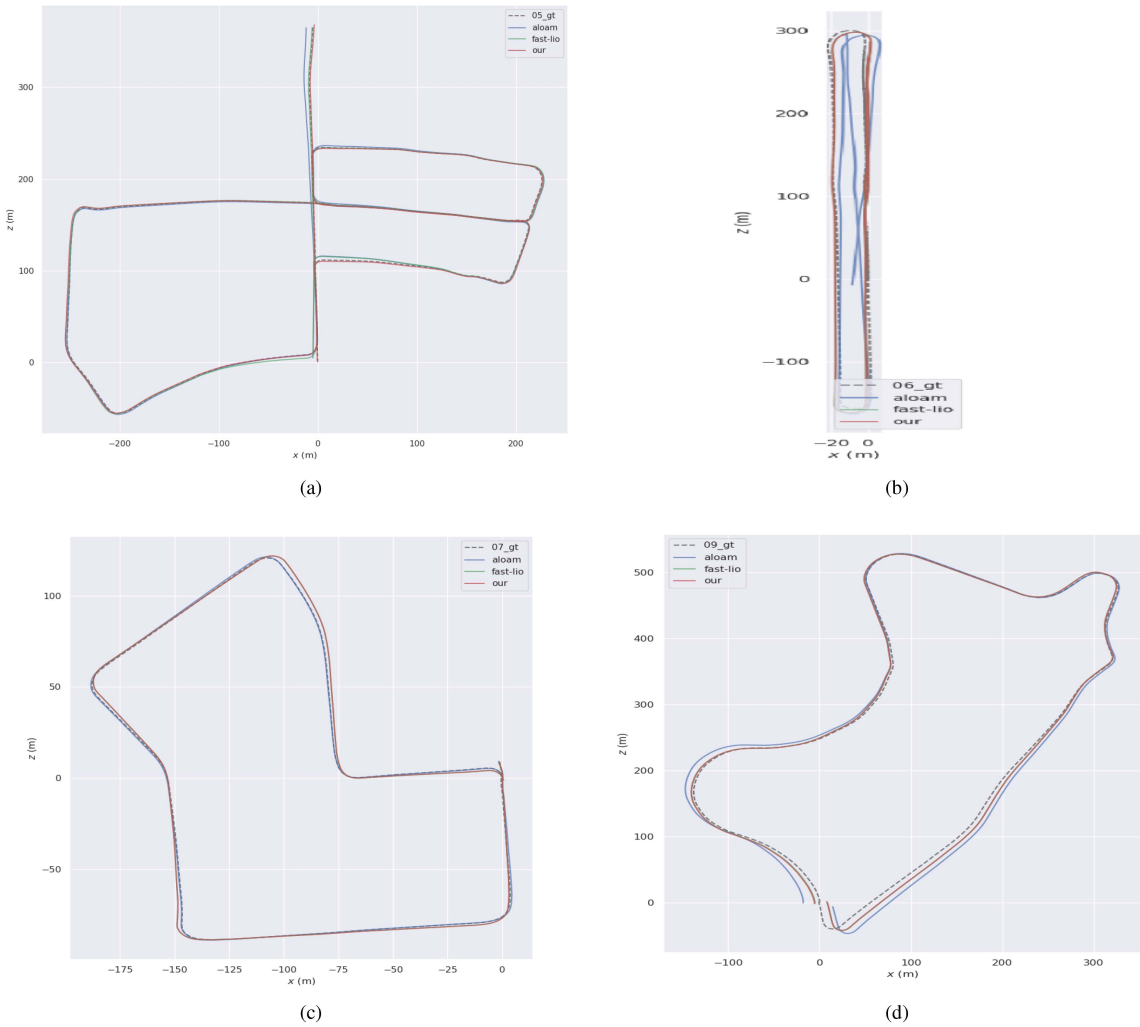


Fig. 5. Visualization of trajectory comparison between A-LOAM, FAST-LIO, and our method on KITTI dataset sequences. (a), (b), (c), and (d) correspond to sequences 05, 06, 07, 09, respectively. (a) 05 sequence. (b) 06 sequence. (c) 07 sequence. (d) 09 sequence.

These results demonstrate that the proposed algorithm achieves smaller trajectory estimation errors compared to the ground truth trajectories in the KITTI sequences, indicating higher accuracy of the algorithm.

The RPE is used to compare pose increments and describe the accuracy of the pose difference between two frames relative to the real pose under a fixed time difference. The error values of the three algorithms on the four sequences are shown in Table II. It can be observed that the algorithm proposed in this article does not achieve the minimum error in each case. However, for sequences 05, 06, and 09, the RMSE statistics of the proposed algorithm are the lowest among the three algorithms. In addition, the proposed algorithm also exhibits the majority of the lowest errors in other statistical values. Overall, the algorithm proposed in this article demonstrates superior performance.

2) Self-Collected Dataset: The experimental platform shown in Fig. 4 was used to collect dataset within the campus of Northwest A&F University, including the Park dataset and Street dataset, which contain LiDAR and IMU information. The satellite map is shown in Fig. 6, and it was used for evaluating the algorithm. The algorithm proposed was compared with three other algorithms: SC-A-LOAM [20] and Optimized-sc-f-loam [3], these only use LiDAR data, and FAST-LIO2 [14] which utilize both LiDAR and IMU data. The trajectories obtained by running different algorithms are shown in Fig. 7, which include long-distance loop closures. Fig. 8 displays the trajectories in the x, y, and z directions. Due to different numbers of keyframes extracted, the length of the displayed trajectories are different.

(a) (b)

Fig. 6. Satellite map of campus dataset. (a) The dataset consists of a long-distance navigation around the campus garden. (b) The dataset depicts walking around the teaching building. (a) Park dataset. (b) Street dataset.

TABLE I
ATE EVALUATION FOR DIFFERENT SEQUENCES AND ALGORITHMS IN KITTI DATASET

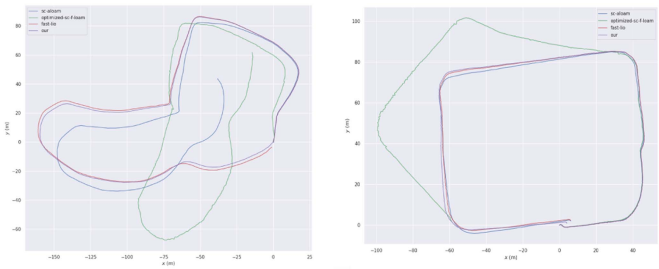
Seq	Algorithm	max	mean	median	min	rmse	std
05	A-LOAM	10.798814	2.736601	2.225152	0.802594	3.226805	1.709762
	FAST_LIO2	5.037337	2.646398	2.614963	0.440284	2.774483	0.833268
	Our	**2.747648**	**1.048157**	**1.041072**	**0.194009**	**1.124527**	**0.407343**
06	A-LOAM	41.068987	16.808181	12.257649	2.984678	20.805651	12.262142
	FAST_LIO2	9.123730	5.137147	5.117446	**1.172659**	5.547861	2.094870
	Our	**8.898170**	**5.085799**	**5.012891**	1.363148	**5.483459**	**2.050114**
07	A-LOAM	4.190354	2.413983	2.490165	0.193651	2.637720	1.063133
	FAST_LIO2	1.454183	0.862997	0.906948	0.225024	0.899657	0.254202
	Our	**1.203614**	**0.528793**	**0.499551**	**0.072487**	**0.572007**	**0.218106**
09	A-LOAM	19.509797	7.701764	4.796878	1.152445	9.723217	5.934962
	FAST_LIO2	**8.122218**	3.918664	3.780629	0.879314	4.277362	1.714613
	Our	8.947627	**3.390457**	**3.241449**	**0.642775**	**3.762784**	**1.631976**

The table shows the error in the translation section of the ATE in meters.

TABLE II
RPE EVALUATION FOR DIFFERENT SEQUENCES AND ALGORITHMS IN KITTI DATASET

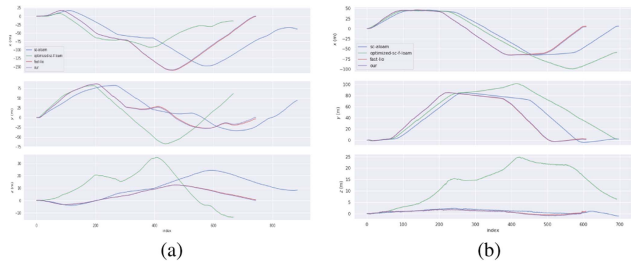
Seq	Algorithm	Max	Mean	Medium	Min	Rmse	std
05	A-LOAM	**1.705681**	1.124697	1.218758	0.000723	1.197308	**0.410614**
	FAST_LIO2	2.192275	1.122483	1.195423	**0.000652**	1.198430	0.419840
	Our	2.187867	**1.121042**	**1.193968**	0.000766	**1.196875**	0.419255
06	A-LOAM	3.286003	1.610941	1.684661	0.725726	1.670571	**0.442352**
	FAST_LIO2	2.920592	1.601048	1.635046	0.530281	1.669592	0.473479
	Our	**2.916118**	**1.599459**	**1.633482**	**0.529994**	**1.667864**	0.472756
07	A-LOAM	**1.698377**	**0.895032**	0.993363	**0.001463**	**1.000292**	**0.446658**
	FAST_LIO2	2.182669	0.901509	0.971551	0.001638	1.011673	0.459089
	Our	2.174258	0.898967	**0.969243**	0.001600	1.008764	0.457673
09	A-LOAM	**2.154416**	1.505133	1.490334	0.401394	1.559616	**0.368625**
	FAST_LIO2	3.075804	1.508063	1.488676	0.379569	1.557537	0.389443
	Our	3.065354	**1.504548**	**1.485061**	**0.378929**	**1.553861**	0.388354

The table shows the error in the translation section of the RPE in meters.



(a) (b)

Fig. 7. Visual comparison of the trajectory projections on the x-y plane obtained by running four different algorithms on the campus environment data set. (a) Park dataset. (b) Street dataset.



(a) (b)

Fig. 8. Trajectories output in the x-y-z directions by running four different algorithms on the campus environment dataset. (a) Park dataset. (b) Street dataset.

As shown in Fig. 7, it can be observed that the algorithm proposed in [3] exhibited significant drift in the trajectories that obtained from both environments, indicating poor robustness of the algorithm. Although the method proposed in [20] obtained a complete trajectory in the Street dataset, suffered from severe drift and incomplete trajectory in Park dataset. The algorithm in [14] produced a complete trajectory but fails to form valid loop closures. As seen in Figs. 7 and 8, our method can generate complete trajectories and effectively detect loop closures, and the output trajectory in the z direction is smoother, indicating lower elevation errors.

C. Time Consumption of Loop Closure Detection

This module is compared with the A-LOAM algorithm with Scan Context (SC-A-LOAM) to evaluate the computational efficiency of the loop closure detection process using the publicly available KITTI dataset (05, 06, 07, 09), and the number of keyframes extracted by the algorithm is also recorded, as shown

TABLE III
COMPARISON OF AVERAGE LOOP CLOSURE DETECTION TIME FOR DIFFERENT SEQUENCES AND ALGORITHMS IN KITTI DATASET

Seq	Algorithm	keyFrames	time(ms/loopFrames)
05	SC-A-LOAM	1686	451.07 ms
	Our	1508	160.36 ms
06	SC-A-LOAM	914	598.36 ms
	Our	749	242.45 ms
07	SC-A-LOAM	506	524.04 ms
	Our	491	113.33 ms
09	SC-A-LOAM	1264	629.46 ms
	Our	1130	161.67 ms

TABLE IV
COMPARISON OF ATE AND RPE FOR DIFFERENT SEQUENCES AND GLOBAL DESCRIPTOR ALGORITHMS IN KITTI DATASET

Seq		SC	ISC	Our
05	ATE	1.135677	**1.122686**	1.124527
	RPE	1.198434	1.198432	**1.196875**
06	ATE	5.485114	5.488297	**5.483459**
	RPE	1.669588	1.669608	**1.667864**
07	ATE	0.795448	0.743867	**0.572007**
	RPE	1.011915	1.011760	**1.008764**
09	ATE	4.729015	3.908313	**3.762784**
	RPE	1.558184	1.557524	**1.553861**

The table shows the RMSE in meters between the estimated trajectory and the ground truth trajectory.

in Table III. According to the data, our method achieved the best computational time for four sequences. For each sequence, compared to the SC-A-LOAM algorithm, the time consumption of loop closure detection of the proposed algorithm is reduced by 64.4%, 59.4%, 78.3%, and 74.3%, respectively, achieving improved algorithm efficiency while ensuring algorithm accuracy.

D. Performance Analysis of Global Descriptor

A new method for computing global descriptors was proposed. We used four sequences (05, 06, 07, 09) from the KITTI dataset to validate its effectiveness. The RMSE of the ATE and RPE between the trajectories obtained using three different global descriptor computation methods and the ground truth was compared using the EVO evaluation tool, and the results are shown in Table IV. It can be observed that for the 05 sequence, the ATE error of our is slightly higher than ISC. However, for the sequences 06, 07, and 09, our method achieved the lowest error values. Compared to the SC and ISC methods, our method shows an average improvement of approximately 16.18% and 8.97% for ATE, 0.23% and 0.21% for RPE, respectively.

E. Analysis of Mapping

In the experiment of outdoor 3-D point cloud map reconstruction, we used the KITTI sequence 07 and self-collected dataset for analysis. Four LiDAR SLAM methods (SC-A-LOAM, Optimized sc-f-loam, LIO-SAM, FAST-LIO2) are compared to validate the mapping effectiveness of the proposed method in this article. The results are shown in Fig. 9–12.

According to the global map in Fig. 9, which can be observed that the lane markings and directional contours are clearly visible in the mapping results of all three algorithms. However, in Fig. 9(a), the point cloud is dense and the generated map exhibits cumulative drift; In Fig. 9(b), the mapping result is blurry and obstacles on the lane cannot be clearly displayed. The environmental map established in Fig. 9(c) has blurred boundaries and cannot clear obstacles on the displayed road. The point cloud in Fig. 9(d) is dense, and the global map is blurry, but the local map Fig. 10(d) is clear and has clear boundaries. However, the map established in Fig. 9(e) of the algorithm in this article eliminates most of the "ghosts" generated by cumulative drift. Fig. 10 shows that the mapping results of our algorithm was neat on straight roads and corners, with no significant drift. It clearly displays obstacles such as cars on the road, indicating improved robustness of the algorithm.

Meanwhile, in order to verify the accuracy of the proposed algorithm on the outdoor self-built dataset, the global mapping effect of the above five algorithms running the Park dataset is shown in Fig. 11. It can be seen from the figure that the SC-A-LOAM algorithm and the Optimized sc-f-loam algorithm are fuzzy, with large cumulative errors and serious drift. FAST-LIO2 algorithm has clear map boundaries and obvious color contrast, but the point cloud is dense and cannot effectively detect the closed loop. Then, the map established by the LIO-SAM algorithm and this algorithm is clear and can effectively detect the closed loop (Fig. 12) and make trajectory correction. However, the previous map established by LIO-SAM algorithm will disappear when the running time is too long. The map established by the algorithm in this article eliminates most of the accumulated drift and has clear boundaries, which will have high robustness and positioning accuracy in actual scenes.

For the experiment of 3-D point cloud map reconstruction in indoor scenes, this article collects the datasets by VLP-16 and N100 inertial navigation in the indoor environment of the library. By comparing the four kinds of LiDAR SLAM method (SC-A-LOAM, Optimized-sc-f-loam, LIO-SAM, FAST-LIO2) to verify the effect of the proposed method.

Fig. 13 shows the indoor environment for data collection. Fig. 14 shows the mapping results obtained by five different algorithms. It can be seen from the figure that (a) the mapping is fuzzy and the outline of the building can be generally seen but not clearly. (b) The mapping effect of the algorithm cannot see obstacles and clear routes in the environment. (c) The contour of the algorithm is clear, and the indoor environment structure can be clearly seen. (d) The algorithm creates large shadows, dense point clouds, and cannot clearly distinguish indoor objects. (e) The map built by the algorithm in this article is relatively clear, there is no double shadow, and the environment structure can be seen, but the point cloud is relatively sparse, and the map is not dense enough. To sum up, LIO-SAM and the algorithm in this article have the best effect of indoor environment mapping.

F. CPU and Memory Usage Analysis

This section mainly analyzes the CPU and memory usage of different algorithms running different datasets by comparing

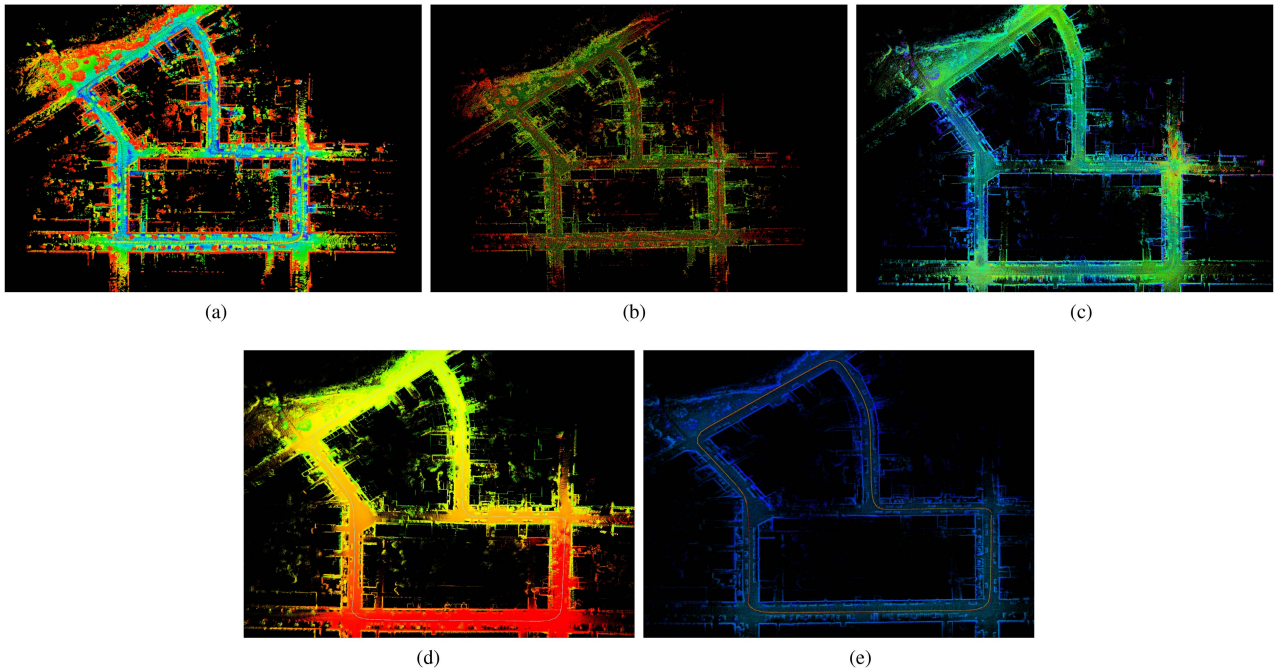


Fig. 9. Visualization of the global map reconstruction for KITTI sequence 07. (a) for A-LOAM algorithm with Scan Context loop closure detection, (b) for F-LOAM algorithm with loop closure detection, (c) shows the LIO-SAM algorithm, (d) shows the mapping results of the FAST-LIO2 algorithm, (e) for the proposed method in this article. (a) SC-A-LOAM. (b) Optimized-sc-f-loam. (c) LIO-SAM. (d) FAST-LIO2. (e) Our.

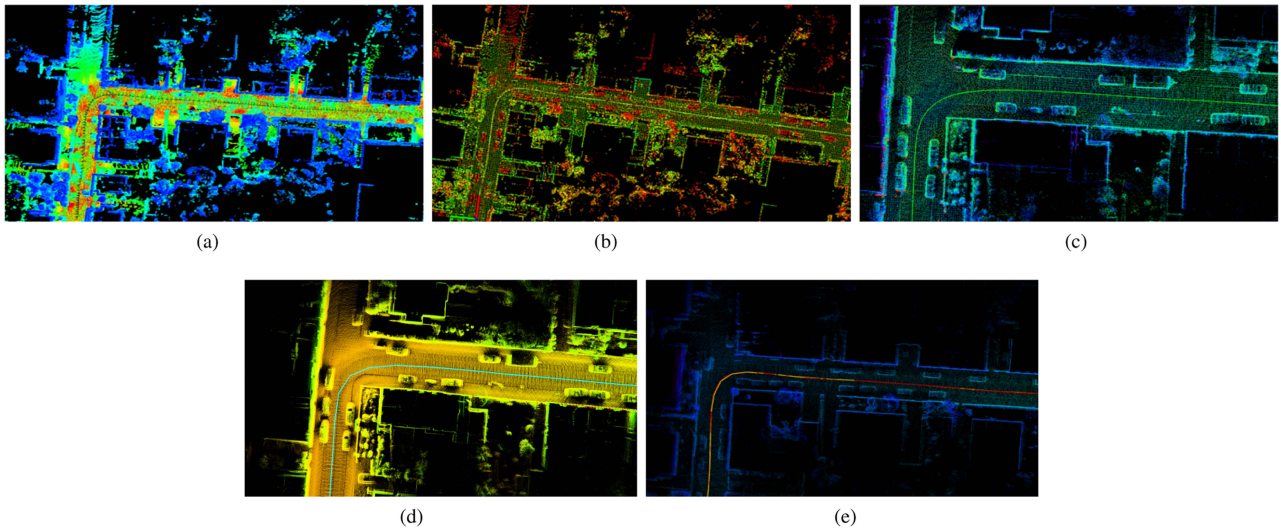
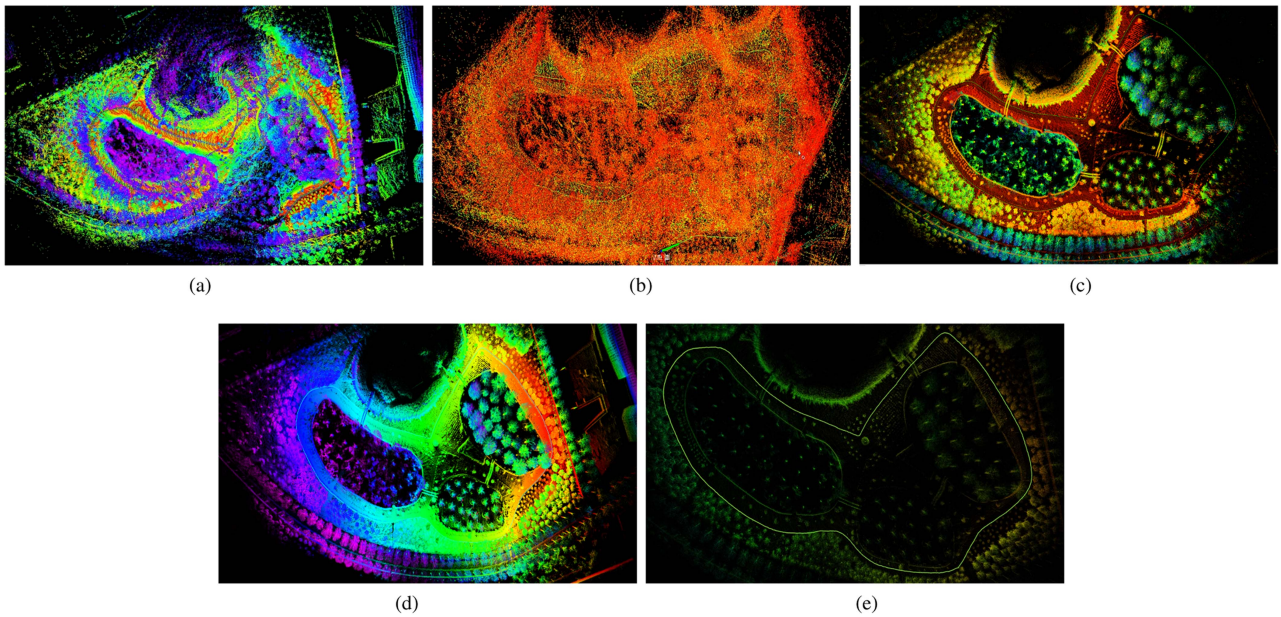


Fig. 10. Visualization of the local map reconstruction for KITTI sequence 07 (showing the map reconstruction on straight roads and at corners). (a) SC-A-LOAM. (b) Optimized-sc-f-loam. (c) LIO-SAM. (d) FAST-LIO2. (e) Our.

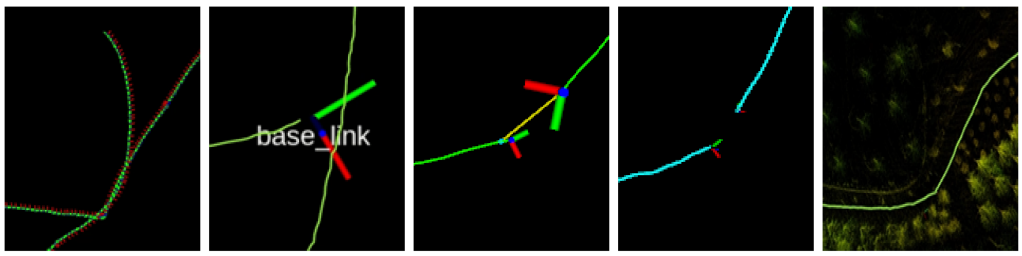
five different algorithms running three different datasets (KITTI 07 sequence, outdoor park, and indoor library). The results are shown in Fig. 15, which show the maximum CPU and memory usage when running the dataset.

As can be seen from the data in the figure, for the KITTI dataset 07 sequence, the CPU usage was 82%, 48.0%, 71.7%, 100%, 59.8%, and the memory usage was 42.8%, 36.8%, 37.9%, 99.2%, 37.5%, respectively. Optimized-sc-f-loam is seemed low with poor accuracy, followed by the algorithm in this article. For the self-collected outdoor dataset, the CPU usage is 100%, 99.0%, 60.6%, 96.0%, 46.5%, and the memory usage is 45.0%, 54.7%, 40.9%, 88.9%, 63.3%, respectively. The lowest CPU usage is the algorithm in this article, with a little memory usage increasing. For the self-collected indoor dataset, the CPU usage was 98.0%, 49.0%, 48.0%, 53.5%, 39.8%, and the memory usage was 28.9%, 28.4%, 29.9%, 46.7%, 30.9%, respectively. The CPU usage was the lowest in this algorithm with a little memory usage increasing. According to the above analysis, it can be seen that the CPU usage of the proposed algorithm is small in different datasets, but for some outdoor datasets, our



(a) (b) (c) (d) (e)

Fig. 11. Global Map of Park dataset. (a) SC-A-LOAM. (b) Optimized-sc-f-loam. (c) LIO-SAM. (d) FAST-LIO2. (e) Our.



(a) (b) (c) (d) (e)

Fig. 12. Loop closures detection of Park dataset. (a) and (b) show severe trajectory drift and the absence of proper loop closures. (c) The closed loop can be detected. (d) Cannot form a closed loop. (e) The proposed method performs loop closure correction. (a) SC-A-LOAM. (b) Optimized-sc-f-loam. (c) LIO-SAM. (d) FAST-LIO2. (e) Our.

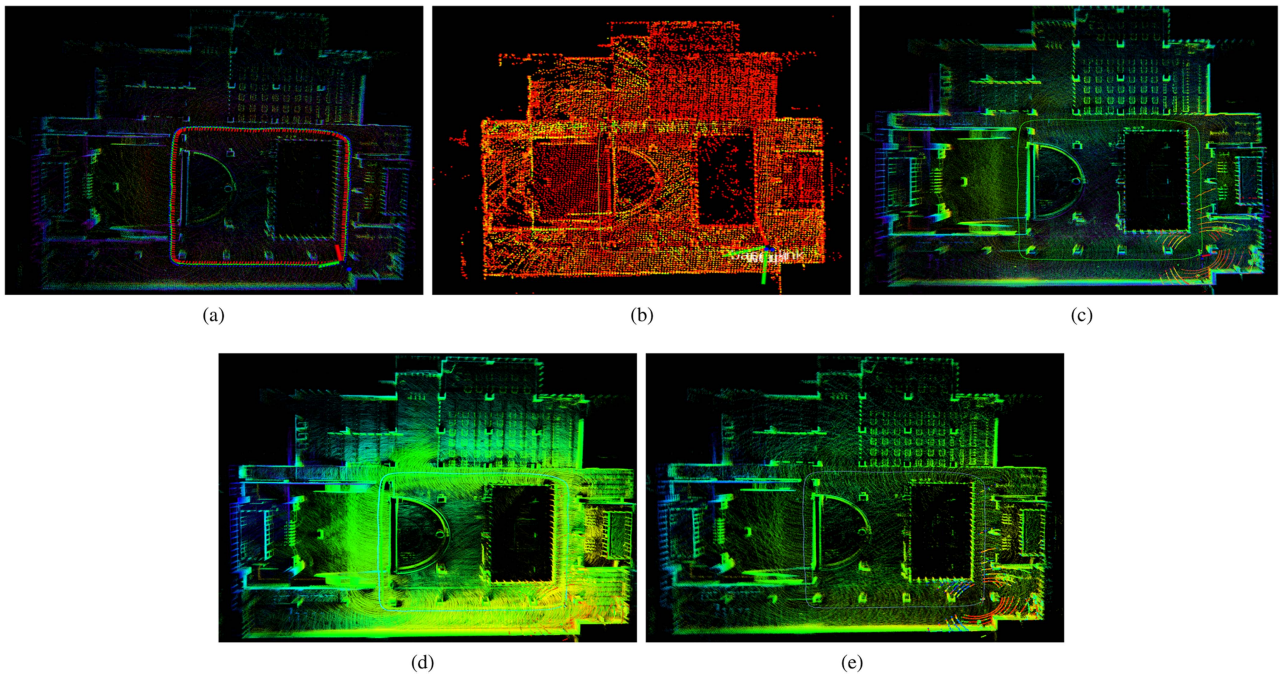
(a) (b)

Fig. 13. Indoor data collection environment. Walk around the library to record data.

algorithm will produce a little memory usage increasing due to accurate mapping in complex environment.

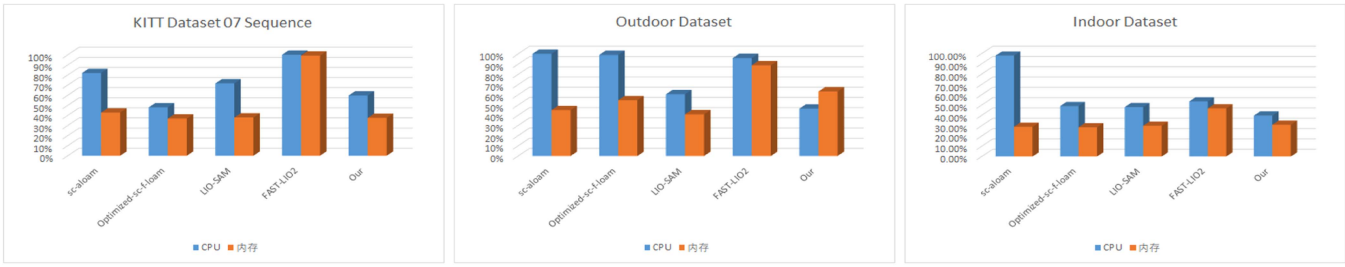
V. Discussion

In this article, we proposed a LiDAR-IMU tightly-coupled SLAM method with IEKF and loop closure detection to address the problems of decreased localization and mapping accuracy caused by the accumulation of errors in a wide range of indoor and outdoor scenes. A loop closure detection method based on HISC global descriptor and a trajectory global optimization method based on GTSAM are proposed. The trajectory accuracy was measured by comparing with the ground-truth value of the public KITTI dataset, which show that in most cases, the proposed algorithm has lower trajectory errors and is close to the ground truth, as well as achieve more accurate pose estimation and improve localization accuracy. Then, the Global Descriptor experimental results show that the characteristics of points can



(a) (b) (c) (d) (e)

Fig. 14. Indoor library global map. (a) SC-A-LOAM. (b) Optimized-sc-f-loam. (c) LIO-SAM. (d) FAST-LIO2. (e) Our.



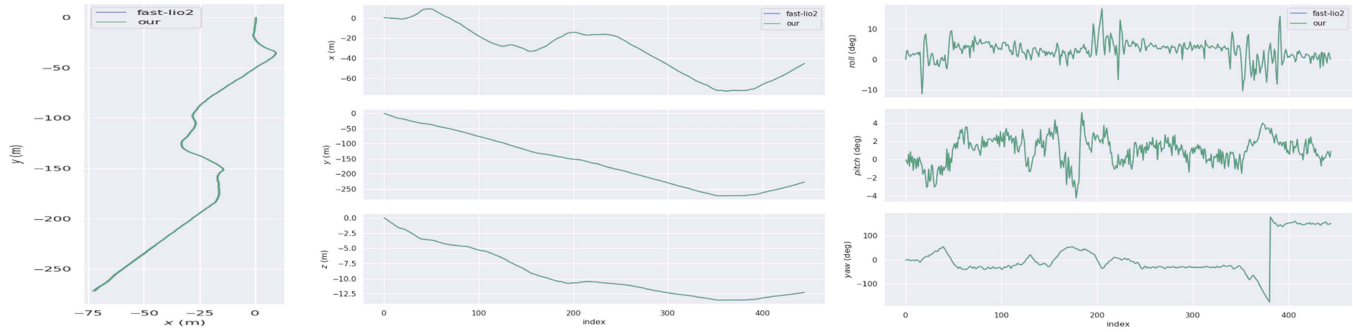
(a) (b) (c)

Fig. 15. CPU and memory usage when running three dataset using five different algorithms. (a) KITTI 07 sequence, (b) self-collected outdoor park datasets, (c) self-collected indoor library datasets.

be displayed more comprehensively by combining the intensity information and the height information, which reducing the probability of error loop closure detection and improving the accuracy. At the same time, the backend global optimization reduces the drift of the mapping trajectory. When constructing the global descriptor, we set a threshold for dividing the point cloud into grids. From the experimental results of both public and self-built datasets, it can be seen that the consistency of intensity information in the same subspace is basically ensured. However, if in different lighting conditions, reflective surfaces, or occlusions, a smaller threshold may need to be set to ensure strength consistency, which may lead to a slight increasing in time consumption. Time comparison with SC-A-LOAM on KITTI dataset indicates that the positional constraints constructed on extracted feature points can address high point cloud volume, reduce estimation cost, and shorten system runtime. Through the introduction of loop closure detection and global optimization, the problems of motion trajectory drift and map overlap are solved.

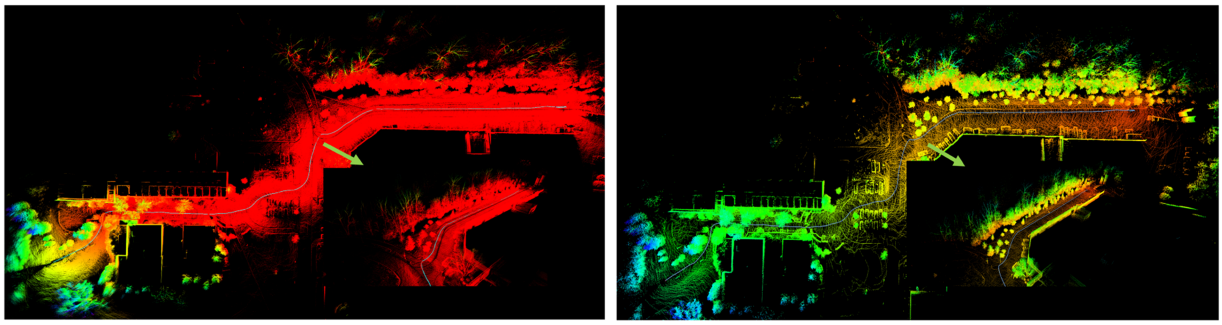
Furthermore, to verify the generalization ability of the algorithm, we conducted experiments on the Velodyne dataset of FAST-LIO2. As ground truth was not provided, we compared the trajectory and mapping result of the benchmark framework and the improved network on this dataset, as shown in Figs. 16–17. The trajectory results show that our algorithm coincides completely with the FAST-LIO2 trajectory. According to the mapping results, although the point cloud density of our drawing is not as dense as that of FAST-LIO2, the result of mapping is clear with less memory consumption and time consumption. Overall, the proposed algorithm can quickly, accurately and clearly reconstruct 3-D scenes in real time in the lowest CPU usage during runtime, with a little memory usage increasing.

Based on the above experimental results, the SC-A-LOAM algorithm and the Optimized sc-f-loam algorithm have poor trajectory accuracy and mapping effect on the outdoor park dataset, with severe trajectory drift and blurred mapping; the FAST-LIO2 algorithm cannot detect and form a closed loop when running on a dataset with loopback, and occupies a large amount of



(a) (b) (c)

Fig. 16. Trajectory results of FAST-LIO2 and Our algorithms on the publicly available Velodyne dataset are shown in the figure. They are 2-D visualization results, x-y-z three-axis visualization results, and RPY visualization results.



(a) (b)

Fig. 17. Figure shows FAST-LIO2 and our algorithm for global and local mapping. (a) FAST-LIO2. (b) Our.

CPU and memory; The proposed algorithm and LIO-SAM have high mapping accuracy in indoor and outdoor environments, good real-time performance, can accurately identify and detect loops, with low CPU usage during runtime. The experiments have verified that our proposed algorithm has high accuracy of positioning map construction, which can be applied to urban roads, forest parks, and spacious indoor environments, and can build environmental maps in real time with low CPU usage and a little memory increasing.

VI. Conclusion

In order to solve the problem of accuracy degradation caused by cumulative errors in a wide range of environments and the failure of localization map construction caused by LiDAR SLAM degradation in scenes with insufficient structural features, this article-based FAST-LIO2 proposed a LiDAR-IMU tightly-coupled SLAM method with IEKF and loop closure detection. The proposed method has been thoroughly evaluated in a variety of environments on both public datasets and data collected on self-built data acquisition platforms. The results show that compared with A-LOAM, FAST-LIO2, and LIO-SAM, the proposed method can achieve similar or better accuracy, and the CPU usage is low, and good real-time performance of the algorithm. Future work will continue to optimize back-end mapping and reduce memory usage.

References

[1] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.

[2] Z. Ni, M. Li, R. Lin, L. Sun, and S. Liu, "Research on SLAM based on fusion of 3D LiDAR and RTK," *Manuf. Automat.*, vol. 42, pp. 51–54, 2020.

[3] L. Liao, C. Fu, B. Feng, and T. Su, "Optimized SC-F-LOAM: Optimized fast LiDAR odometry and mapping using scan context," in *Proc. 6th CAA Int. Conf. Veh. Control Intell.*, 2022, pp. 1–6.

[4] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A LiDAR-inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 8899–8906, doi: 10.1109/icra40945.2020.9197567.

[5] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot. Sci. Syst. X*, 2015, pp. 1–9, doi: 10.15607/rss.2014.x.007.

[6] X. Yang and S. Li, "Principles, current status, and trends of visual slam loop closure detection for mobile robots," *J. Electron. Meas. Instrum.*, vol. 36, pp. 1–12, 2022.

[7] Z. Zhou and S. Di, "LiDAR SLAM algorithm based on intensity scan context loop closure detection," *J. Chin. Inertial Technol.*, vol. 30, pp. 738–745, 2022.

[8] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4802–4809, doi: 10.1109/iros.2018.8593953.

[9] R. Dube, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5266–5272, doi: 10.1109/icra.2017.7989618.

[10] X. Ji, L. Zuo, C. Zhang, and Y. Liu, "LLOAM: LiDAR odometry and mapping with loop-closure detection based correction," in *Proc. IEEE Int. Conf. Mechatron. Autom.*, 2019, pp. 2475–2480, doi: 10.1109/icma.2019.8816388.

[11] Y. Ling, T. Liu, and S. Shen, "Aggressive quadrotor flight using dense visual-inertial fusion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1499–1506, doi: 10.1109/icra.2016.7487286.

[12] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Ins. Tech., Tech. Rep, vol. 2, p. 4, 2012.

[13] G. Xue, J. Wei, R. Li, and J. Cheng, "LeGO-LOAM-SC: An improved simultaneous localization and mapping method fusing LeGO-LOAM and scan context for underground coalmine," *Sensors*, vol. 22, no. 2, Jan. 2022, Art. no. 520, doi: 10.3390/s22020520.

[14] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022, doi: 10.1109/tro.2022.3141876.

[15] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 22–29, doi: 10.1109/iros.2010.5649043.

[16] C. Zheng, F. Ke, and Q. Tang, "Research on laser inertial tightly coupled slam based on graph optimization," *Electron. Meas. Technol.*, vol. 46, pp. 35–42, 2023.

[17] S. Agarwal et al., "Ceres solver: Tutorial & reference," Google Inc., vol. 2, no. 72, p. 8, 2012.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361, doi: 10.1109/cvpr.2012.6248074.

[19] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765, doi: 10.1109/iros.2018.8594299.

[20] gisbi kim, "SC-A-LOAM," *GitHub*, Oct. 2021. [Online]. Available: https://github.com/gisbi-kim/SC-A-LOAM

[21] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 2095–2101, doi: 10.1109/icra40945.2020.9196764.

[22] X. Xu et al., "A review of multi-sensor fusion slam systems based on 3D LiDAR," *Remote. Sens.*, vol. 14, no. 12, 2022, Art. no. 2835.

[23] C. Wang, G. Zhang, and M. Zhang, "Research on improving LIO-SAM based on intensity scan context," *J. Phys. Conf. Ser.*, vol. 1827, Mar. 2021, Art. no. 012193, doi: 10.1088/1742-6596/1827/1/012193.

[24] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3923–3929, doi: 10.1109/iros.2013.6696917.

[25] J. Tang et al., "LiDAR scan matching aided inertial navigation system in GNSS-denied environments," *Sensors*, vol. 15, pp. 16710–16728, Jul. 2015, doi: 10.3390/s150716710.

[26] W. Zhen, S. Zeng, and S. Soberer, "Robust localization and localizability estimation with a rotating laser scanner," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6240–6245, doi: 10.1109/icra.2017.7989739.

[27] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018, doi: 10.1109/tro.2018.2853729.

[28] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142, doi: 10.1109/iros45743.2020.9341176.

[29] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021, doi: 10.1109/lra.2021.3064227.

[30] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D LiDAR inertial odometry and mapping," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 3144–3150, doi: 10.1109/icra.2019.8793511.

[31] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *Int. J. Robot. Res.*, vol. 31, pp. 216–235, Feb. 2012, doi: 10.1177/0278364911430419.

[32] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental k-d tree for robotic applications," 2021, *arXiv:2102.10808*.

Huimin Pan received the bachelor's degree in computer science and technology from Yantai University, Shandong, China, in 2020. She is currently working toward the master's degree in computer technology with Northwest A&F University, Shaanxi, China.

Her research interests include computer vision, LiDAR SLAM, and multisensor fusion.

Dongfeng Liu received the Ph.D. degree in agriculture from China Agricultural University, Beijing, China, in 2010.

He is currently working on crop phenotype research with the Shenzhen Agricultural Science and Technology Promotion Center, Shenzhen, China, mainly engaged in crop phenotype automation mainly work.

Jingzheng Ren received the bachelor's degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2021. He is currently working toward the master's degree in computer science with Northwest A&F University, Shaanxi, China.

His research interests involve visual SLAM, multisensor fusion SLAM, 3-D reconstruction, and robotics.

Tianxiong Huang received the master's degree in horticulture from South China Agricultural University, Guangzhou, China, in 2014.

At present, she is engaged in the comprehensive work of agricultural science and technology project management with the Shenzhen Agricultural Science and Technology Promotion Center, Shenzhen, China.

Huijun Yang received the Ph.D. degree in agricultural electrification and automation from Northwest A&F University, Shaanxi, China, in 2013.

She is currently working with Northwest A&F University. Her research interests include computer graphics, intelligent agriculture, artificial intelligence, SLAM, point clouds, phenomics, etc.