An Efficient LiDAR SLAM with Angle-Based Feature Extraction and Voxel-based Fixed-Lag Smoothing

Nuo Li, Yiqing Yao, Xiaosu Xu, Yiyang Peng, Zijian Wang, Hongyu Wei

Abstract—**LiDAR simultaneously localization and mapping (SLAM), which provides accurate pose estimation and map construction, has a broad range of applications in autonomous driving. However, in some complex scenarios, such as degraded environments, SLAM performance is not up to the requirements of an autonomous driving system. To aim at the above-mentioned problems, we propose an accurate and robust LiDAR SLAM method. First, in order to avoid the uncertainty of LiDAR viewpoint variation, an angle-based feature extraction method is proposed based on the equiangular distribution property of the point cloud. Second, in view of the fact that odometry errors accumulate, we constructed a fixed-lag smoothing to jointly optimize the poses of multiple keyframes. In addition, to improve the environmental representation of point cloud maps within a sliding window, we maintain two types of abstract voxel maps. Finally, a voxel-based feature matching method based on voxel geometry constraints is proposed for refinement of the pose transformation within a sliding window. The performance and efficiency of the proposed method are evaluated on the public KITTI, Mulran, The Newer College dataset benchmarks and the dataset collected by our sensor system. The experimental results show that accurate feature extraction, efficient voxel feature matching, and consistent fixed-lag smoothing help our LiDAR SLAM method achieve better performance in multiple spatially and temporally large-scale scenarios compared to other existing state-of-the-art methods.**

Index Terms—**Feature extraction, point cloud maps, smoothing, voxel feature matching, LiDAR simultaneous localization and mapping (SLAM)**.

I. Introduction

Continuously and reliable localization and mapping are essential prerequisites for autonomous driving vehicles to accomplish perception [1], decision-making [2], and collaboration [3] tasks. To accomplish these tasks in complex scenarios, the localization and mapping of self-driving cars usually rely on a variety of sensors, such as GPS/IMU, LiDAR, and cameras. In recent years, LiDAR SLAM [4], [5] and visual SLAM [6], [7] have received extensive attentions from researchers. LiDAR is a sensor that is unaffected by ambient light, has accurate distance sensing, and has a wide range of perception of the surrounding environment. Consequently, LiDAR SLAM offers numerous advantages, including the ability to achieve accurate pose estimation in outdoor environments with six degrees of freedom (DoF). On the other hand, LiDAR SLAM plays a crucial role in various autonomous driving tasks, such as object detection and tracking [8], point cloud map construction [9], and maintenance [10]. As a result, it is widely recognized that LiDAR SLAM offers a viable solution for autonomous driving platforms seeking accurate localization and map construction.

To reduce the computational complexity of redundant point clouds, a number of feature-based LiDAR SLAM methods have been proposed. LOAM [11], as a classical 3D LiDAR SLAM algorithm, extracts edge and plane features in the environment based on curvature. Similar to LOAM, LeGO-LOAM [12] introduces ground segmentation and noise filtering to extract reliable edge and plane features. However, none of them account for the influence of the LiDAR viewpoint on the curvature, which results in the inaccuracy of the extracted feature points. Guo et al. [13] take into account the effect of viewpoints on curvature and calculate the geometrical configuration of neighboring point clouds on the same beam using Principal Component Analysis (PCA) [14], which differentiate edge and plane feature points. However, the computational cost increasing is unavoidable due to perform PCA on a large number of point clouds. Therefore, SLAM is in urgent need of an efficient feature extraction method that can take into account the effect of viewpoints and also extract accurate feature points.

In addition, several excellent local optimization algorithms have been proposed to attenuate the cumulative error of the odometer. LOAM [10] uses scan-to-map matching to optimize the frame-to-frame matching poses. Although it can reduce the cumulative error, maintaining the interrelated feature points within the local map is a difficult task. Unlike scan-to-map, in HDL-graph-SLAM [15], keyframe-to-keyframe matching is used to refine the frame-to-keyframe poses. However, the matching way only associates the previous keyframe, and the accurate pose transformations need to be obtained by time-

Color versions of one or more of the figures in this article are available online at Nuo li, Yiqing Yao, Xiaosu Xu, Yiyang Peng, Zijian Wang, Hongyu Wei are with School of instrument science and technology with Southeast University, Nanjing, China. {Nuo Li, 230209012@seu.edu.cn} {yiqing-yao@seu.edu.cn}. (Corresponding author: Yiqing Yao)

This work was supported in part by 2022 Ministry of Education's "Chunhui Jihua" Cooperative Research Project under Grant HZKY20220128, and in part by Jiangsu Provincial Department of Science and Technology (No. BM2023013)

Nuo Li, Yiqing Yao, Xiaosu Xu, Yiyang Peng, Zijian Wang are with the School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China, and also with the Key Laboratory of Micro-Inertial Instrument and Advanced Navigation Technology, Ministry of Education, Southeast University, Nanjing 210096, China. (email: 230239012@seu.edu.cn; yiqing-yao@seu.edu.cn; xxs@seu.edu.cn; 350264217@qq.com; 230218957@seu.edu.cn)

Hongyu Wei is with at the Institute of Artificial Intelligence, Shanghai University, Shanghai 200444, China (email: 13052059708@163.com)

consuming GICP. In LIO-SAM [16], an incremental smoothing method is used to optimize the keyframe poses. However, the registration of a large number of feature point clouds affects the efficiency. Similarly, a fixed-lag smoothing method [17] has been proposed to address tight multi-sensor fusion, enabling multi-modal landmark tracking and accurate pose estimation. While the technique of feature matching within a sliding window of the fixed-lag smoothing achieves high pose accuracy, it is time-consuming.

Therefore, in this paper, we present a real-time LiDAR SLAM framework to address the feature extraction and local optimization issues mentioned earlier. To tackle the feature extraction problem, we propose an efficient angle-based method. First, we coarsely segment the point cloud using a distance radius threshold to eliminate noise and obtain a locally continuous point cloud. Then, we employ an angle-based curvature calculation method to accurately and stably extract edge and plane features, while mitigating the influence of the LiDAR viewpoint. For the local optimization problem, we propose a voxel-based fixed-lag smoothing method. Using the feature point cloud, we cluster edge and plane features into respective voxels, allowing for a compact representation of the local geometry. Voxel geometry constraints are then applied to refine the keyframe pose transformation. Ultimately, a fixed-lag smoothing module optimizes the keyframe pose within the sliding window, ensuring a highly continuous pose estimation. Additionally, we conduct a series of experiments to validate the effectiveness of our method, thoroughly analyzing pose estimation accuracy, global trajectory error, and method processing time.

In summary, the contributions of this paper are as follows.

1) We introduce a feature-based and voxel-based LiDAR SLAM framework that utilizes a consistent set of point features to construct a geometric representation of voxel features, thus enhancing the efficiency and accuracy of the SLAM framework.

2) Building on the geometric construction of the point cloud, we propose a novel method for defining angle-based curvature to rapidly extract stable feature points.

3) We propose a fixed-lag smoothing method based on voxel matching, in which efficient voxel matching is performed within a sliding window to jointly optimize multiple keyframe poses and reduce the cumulative error of the odometry.

The rest of the paper is organized as follows. Section II reviews the related work of this paper. Section III defines the basic concepts of the parameters involved in the method and presents the details of the proposed method. Section IV designs a number of experiments to validate the effectiveness of the proposed method. Section V concludes the article.

II. Related Work

Feature-based LiDAR SLAM employs alignment of feature point clouds for pose estimation. Numerous methods have been proposed by scholars to enhance the robustness and accuracy of feature-based LiDAR SLAM systems, with respect to both feature extraction and local optimization.

A. Feature Extraction

Feature extraction methods aim to extract representative features from the original point cloud, the vast majority of which are structured features, such as straight lines, planes, spheres. To extract such features, several feature extraction methods have been proposed by many scholars. As the most popular LiDAR SLAM algorithm, LOAM [11] divides feature into edge features and plane features based on the local curvature on the same scan line, which improves the efficiency of point cloud alignment. On the basis, LeGO-LOAM [12] effectively reduces the number of unstable features by ground segmentation and noise removal before feature extraction. Guo et al. [13] uses PCA to compute the geometric configurations of neighboring point clouds on the same LiDAR beam to achieve the extraction of edge features and plane features. WiCRF [18] extracts the features with the adaptive curvature algorithm, which enhances the effect of curvature on the LiDAR point of view by scaling the geometric configuration. WiCRF employs PCA to check the extracted feature points. Since only the feature points are checked, the computational cost is greatly reduced.

B. Local Optimization

Accurate feature extraction procedure improves the accuracy of the pose estimation, but odometry error accumulates in complex environments. With the development of nonlinear optimization methods, several local optimization schemes have been proposed. LOAM [11] performs scan-to-map matching on scan-to-scan basis to reduce odometry drift. HDL-graph-SLAM [15] optimizes the front-end pose using keyframe-to-keyframe matching based on a distribution method. Based on this, Wang et al. [19] proposed a feature-based odometry and HGICP keyframe matching -based optimization approach to improve the accuracy of the pose transformation. The incremental smoothing method was proposed in LIO-SAM [16] for joint optimization of LiDAR and inertial measurement unit (IMU) constraints. Similarly, Shu et al. [20] constructed a tightly-coupled multimodal mapping of LiDAR and camera within a sliding window, which efficiently realizes a joint pose estimation based on the fixed-lag smoothing method. In addition, Wisth et al. [17] proposed a fixed-lag smoothing method for tightly coupled multi-sensor fusion to robustly track the landmarks and quickly estimate the poses.

III. Proposed Efficient Lidar Slam

A. Problem Statement and System Structure

In this work, we consider the LiDAR coordinate frame by L. The world coordinate frame of LiDAR system is fixed to the first LiDAR frame L_1, which is denoted as W. $\mathbb{F}_k = \{p_1^k, p_2^k, \cdots, p_s^k\}$ indicate the point cloud sets perceived at k-th sweep, where $p_i^k(i \in (1,s))$ is a point. We represent the 6-DoF pose transformation between the LiDAR frame to the world frame by $\boldsymbol{T} \in SE(3)$ which consists of relative rotation

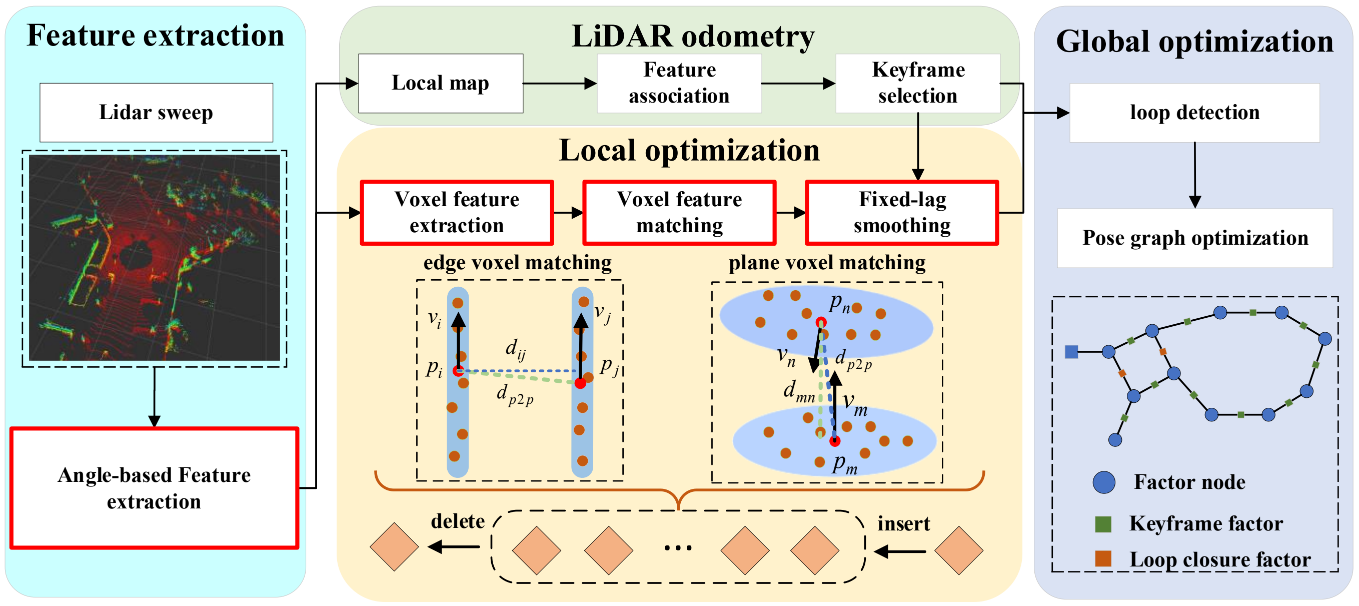

Fig. 1. Overview of the proposed LiDAR SLAM method. The edge and plane features after angle-based feature extraction are transmitted to the LiDAR odometry module. Within the sliding window, voxel feature matching and fixed lag smoothing are used to refine the keyframe poses. Subsequently, loop detection is used for global optimization and point cloud map construction.

$\boldsymbol{R} \in SO(3)$ and relative translation $\boldsymbol{t} \in \mathbb{R}^3$. We extract edge point set $\mathcal{E}$ and plane point set $\mathcal{P}$ from $\mathbb{F}_k$ for estimating the pose of the current frame. The problem of LiDAR odometry can be represented as:

$$\hat{T}_C^* = \arg\min_{\hat{T}_c} \sum_{{}^{\mathcal{E}}_{c}p_i \in \mathbb{F}_c^{\mathcal{E}}} C_{\mathcal{E}}(\hat{T}_c) + \sum_{{}^{\mathcal{P}}_{c}p_i \in \mathbb{F}_c^{\mathcal{P}}} C_{\mathcal{P}}(\hat{T}_c). \tag{1}$$

After LiDAR odometry, we obtain a set of states:

$$\boldsymbol{T}_{1:n} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \dots, \boldsymbol{T}_k \dots, \boldsymbol{T}_n\}. \tag{2}$$

Then, a batch of keyframes K_k is selected to form the keyframe set $K_n (1 \le k \le n)$, from which edge voxel features $\mathbb{K}_k^{\mathcal{E}}$ and plane voxel features $\mathbb{K}_k^{\mathcal{P}}$ are extracted, where $f_{\mathcal{E}}^i$ represents the edge voxel residual factor, and $f_{\mathcal{P}}^x$ represents the plane voxel residual factor. Unlike the full smoothing method [16], a fixed-lag smoothing method performs batch refinement within a fixed time window and keeps only a subset of the most recent keyframe poses, while marginalizing the old poses. For a set of historical states:

$$\boldsymbol{X}_{1:k} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \dots, \boldsymbol{T}_k\}. \tag{3}$$

A fixed-lag smoothing can be represented as a least squares' minimization problem:

$$X_{1:k}^* = \arg\min_{X_{1:k}} \left\{ E_p + \sum_{k\in n} \sum_{i\in\mathbb{K}_k^{\mathcal{E}}} \|f_{\mathcal{E}}^i\|_{\Sigma_{\mathcal{E}}}^2 + \sum_{k\in n} \sum_{x\in\mathbb{K}_k^{\mathcal{P}}} \|f_{\mathcal{P}}^x\|_{\Sigma_{\mathcal{P}}}^2 \right\}. \tag{4}$$

where $E_p = \|r_0\|_{\Sigma_0}^2$ represents the marginalization factor, $\Sigma_0, \Sigma_{\mathcal{E}}$, and $\Sigma_{\mathcal{P}}$ represent the covariances of the marginalization factor, edge voxel residuals, and plane voxel residuals, respectively. After solving the least squares problem, in order to maintain a fixed window size, it is necessary to perform marginalization, removing the no longer needed keyframes and their corresponding feature points from the optimization problem to reduce computational complexity and maintain the fixed window size.

Fig. 1 illustrates the LiDAR SLAM system, composed of four components: feature extraction, LiDAR odometry, local optimization, and global pose graph optimization. The feature extraction module takes the original LiDAR point cloud as input and produces structured edge and plane feature sets. The LiDAR odometry module matches these features with local feature maps to estimate the pose of each frame and update the map. The third module, local optimization, refines keyframe poses using advanced edge and plane voxel features. This module involves three steps: voxel feature extraction, voxel feature matching, and local fixed-lag smoothing. Finally, the global pose graph optimization module utilizes the optimized poses for constant graph-based optimization.

B. Feature Extraction

To address the motion distortion issue in LiDAR point cloud data, we performed motion compensation on each frame. Considering the high scanning frequency of LiDAR, we modeled the motion between two scans using a constant velocity model. Subsequently, we projected all points captured within a sweep period back to the starting time of the frame. For a given point p_i^k, where k denotes the k-th sweep and t_k represents its corresponding timestamp, we estimated the 6-DoF pose transformation for the small-time interval $\delta t = t_i - t_s$ between consecutive scans using linear interpolation:

$$T_k(\delta t) = T_{k-1} exp\left(\frac{\delta t}{t_e - t_s} \log(T_{k-2}^{-1} T_{k-1})\right). \tag{5}$$

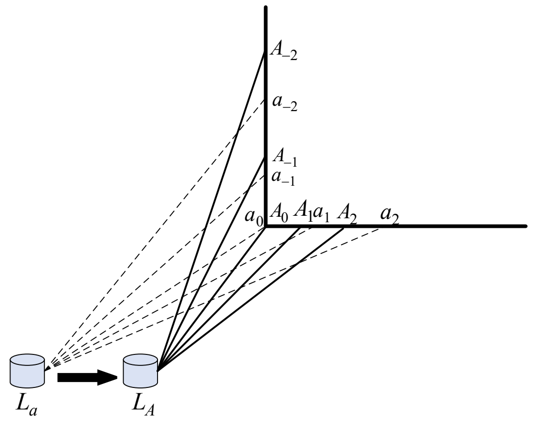

Fig. 2. The diagram of the viewpoint of the LiDAR influence on feature extraction

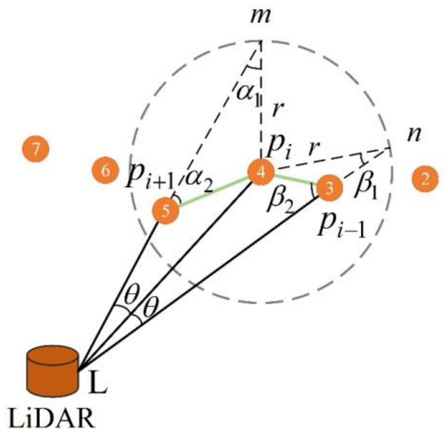

Fig. 3. The diagram of extracting edge and plane feature points based on spatial angle.

where t_s and t_e are starting and ending time of L_k, and function $exp()$ transforms a Lie algebra into Lie group. The corrected point cloud $\mathbb{F}_k^*$ can be denoted by:

$$\mathbb{F}_k^* = \{T_k(\delta t)p_i^k | p_i^k \in \mathbb{F}_k\}. \tag{6}$$

Next, we extract features from each frame of the corrected point cloud $\mathbb{F}_k^*$. The feature extraction technique employed in this study is similar to that described in LOAM, where we classify edge and plane features by evaluating the smoothness between a point and its local plane. In degraded circumstances, due to sparse environmental features and repetitive structures, the number of plane features is insufficient. So it becomes important to accurately extract and match edge points. According to traditional feature extraction methods, curvature ϑ is calculated based on the sum of distances between the point cloud and the LiDAR, expressed as:

$$\vartheta = \frac{1}{|S| \cdot \|p_i\|} \left\| \sum_{p_j \in S, j \neq i} (p_j - p_i) \right\|. \tag{7}$$

where S denotes the set of neighboring points on the same laser beam as the point cloud p_i, and p_j represents any point within S. However, as shown in Fig. 2, when LiDAR moves from L_a to L_A, for the same point cloud $A_0(a_0)$, curvature ϑ_a and ϑ_A can be represented as

$$\vartheta_a = \frac{(\|L_a a_{-2}\| + \|L_a a_{-1}\| + \|L_a a_1\| + \|L_a a_2\| - 4\|L_a a_0\|)}{|S_a| \cdot \|a_0\|} \tag{8}$$

$$\vartheta_A = \frac{(\|L_A A_{-2}\| + \|L_A A_{-1}\| + \|L_A A_1\| + \|L_A A_2\| - 4\|L_A A_0\|)}{|S_A| \cdot \|A_0\|} \tag{9}$$

where due to the change in viewpoint, $\|L_a a_{-2}\| + \|L_a a_{-1}\| + +\|L_a a_1\| + \|L_a a_2\| - 4\|L_a a_0\| \neq \|L_A A_{-2}\| + \|L_A A_{-1}\| + \|L_A A_0\| + \|L_A A_1\| + \|L_A A_2\|$, so $\vartheta_a \neq \vartheta_A$. This indicates that due to changes in perspective, the curvature of the same points has changed, which will affect distance-based feature extraction algorithms. To address this issue, we propose a novel method for assessing smoothness based on curve angles, taking into account the equiangular distribution characteristics of the point cloud. For the point $A_0(a_0)$, we obtain is the angle formed by adjacent point clouds at the point A_0.

As shown in Fig. 3, we collect point clouds with center point p_i and radius r, and these point clouds will be as the nearest point set $\mathcal{T}_p$ in the same row as point p_i. We choose consecutive edge points within the range of r values, and disconnected edge points outside the range. Each point in $\mathcal{T}_p$ is extended along the outgoing direction of the point until it intersects the circle. Our objective is to calculate the angle between the connecting curves of the points and nearest neighbor points. This angle serves as a measure of smoothness. To derive the formula for calculating this angle, we consider various triangles. We use ΔLmp_i to represent the triangle with vertices L, m and p_i. Similarly, ΔLnp_i represents the triangle with vertices L, n, and p_i, Δmp_ip_{i-1} represents the triangle with vertices m, p_i, and p_{i-1}, and Δmp_ip_{i+1} represents the triangle with vertices m, p_i, and p_{i+1}.

In triangles ΔLmp_i and ΔLnp_i

$$\frac{r}{sin\theta} = \frac{\|p_i\|}{sin\alpha_1} \tag{10}$$

$$\frac{r}{sin\theta} = \frac{\|p_i\|}{sin\beta_1} \tag{11}$$

Thus, $\alpha_1 = \beta_1$.
In triangle Δmp_ip_{i-1}

$$\frac{\|p_i - p_{i-1}\|}{sin\alpha_1} = \frac{r}{sin\alpha_2} \tag{12}$$

In triangle Δmp_ip_{i+1}

$$\frac{\|p_{i+1} - p_i\|}{sin\beta_1} = \frac{r}{sin\beta_2} \tag{13}$$

Thus,

$$sin\alpha_2 = \frac{\|p_i\|sin\theta}{\|p_i - p_{i-1}\|} \tag{14}$$

$$sin\beta_2 = \frac{\|p_i\|sin\theta}{\|p_{i+1} - p_i\|} \tag{15}$$

The smoothness evaluation based on curve angle can be estimated by:

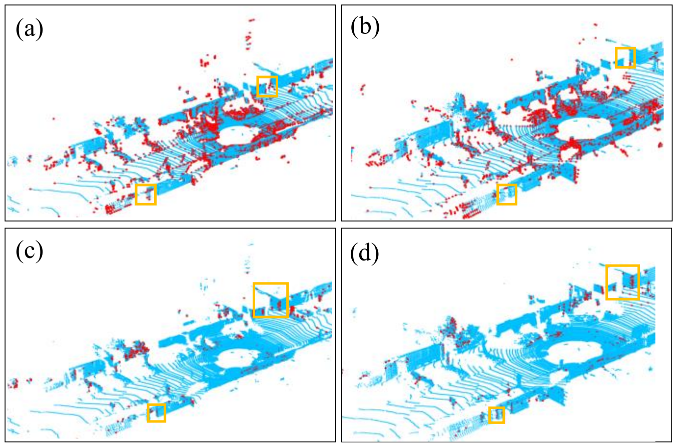

Fig. 4. Comparison of results of angle and curvature edge point extraction. Blue points represent original point cloud, red points represent edge point cloud. (a) and (b) are from the same frame, while (c) and (d) are from the same frame. Fig. (a) and (b) are based on curvature feature extraction, while fig. (c) and (d) are based on angle feature extraction. The feature points within the yellow box transition from presence to absence from (a) to (b), whereas the feature points within the yellow box in (c) and (d) can still be correctly extracted.

$$l_i = arcsin\frac{\|p_i\| \cdot sin\theta}{\|p_i - p_{i-n}\|} + arcsin\frac{\|p_i\| \cdot sin\theta}{\|p_{i+n} - p_i\|}$$
$$c_i = \frac{1}{N}\sum_{n=1}^{N}(l_i - 2\theta) - 180. \qquad (16)$$

where, θ is the angle between consecutive the point clouds. The LiDAR device we used has a fixed horizontal angular resolution. To reduce computational consumption, we select $N = 2$ points in both the clockwise and counterclockwise directions for each point. This comprehensive evaluation allows us to calculate the smoothness of each point. We iterate over all the points in the scan and obtain their corresponding c_i values, which represent the smoothness. These points are then sorted in descending order based on the magnitude of their c_i values. Larger values indicate edge features, while smaller values indicate plane features. Specifically, we first divide each LiDAR beam into six sub-regions on average. Subsequently, we start selecting from highest to lowest based on sorted values, choosing point clouds with c_i value greater than threshold ϕ as edge points. Finally, we ensure that each sub-region has exactly 20 edge points. Based on the analysis and our practical engineering experience, we have decided to set the threshold ϕ to 3.5. For the selection of plane points, the same approach as FLOAM[27] is adopted, non-edge points are selected as plane points. For each scan, we have a set of feature points denoted as $\mathbb{F}'_k = \{\mathbb{F}_k^{\mathcal{E}} \cup \mathbb{F}_k^{\mathcal{P}}\}$. This set consists of both edge features $\mathbb{F}_k^{\mathcal{E}}$ with larger smoothness values and plane features $\mathbb{F}_k^{\mathcal{P}}$ with smaller smoothness values. For the points with a radius greater than r, they are typically scattered edge points. Taking into account the previously extracted edge point quantity, if the quantity is less than Ne, we add the edge points with a radius greater than r. Based on our experimental and engineering experience, the value of Ne is set to 3000. Fig. 4 illustrates a comparison graph depicting the edge point extraction between curvature-based and our angle-based feature extraction. A shift of 5 frames between (a) and (b) results in the inability to extract feature points within the frame of (a) in (b) due to changes in viewpoint. Conversely, although there is also a 5-frame difference between (c) and (d), all feature points within the frame can be extracted. This highlights that angle-based feature extraction is less influenced by changes in viewpoint.

C. LiDAR Odometry

we find the feature correspondence between the frame $\mathbb{F}'_k$ and the local feature map calculate the odometry pose transformation. $\mathcal{M}$ consists of edge feature map $\mathcal{E}$ and plane feature map $\mathcal{P}$. These feature maps are generated from observations of frames $\{L_1, \cdots, L_c, \cdots, L_n\}$ and described as:

$$\mathcal{E} = \{\mathbb{F}_1^{\hat{T}\mathcal{E}} \cup \cdots \cup \mathbb{F}_c^{\hat{T}\mathcal{E}} \cup \cdots \cup \mathbb{F}_n^{\hat{T}\mathcal{E}}\}$$
$$\mathcal{P} = \{\mathbb{F}_1^{\hat{T}\mathcal{P}} \cup \cdots \cup \mathbb{F}_c^{\hat{T}\mathcal{P}} \cup \cdots \cup \mathbb{F}_n^{\hat{T}\mathcal{P}}\}. \qquad (17)$$

where $\mathbb{F}_c^{\hat{T}\mathcal{E}} = \hat{T}\mathbb{F}_c^{\mathcal{E}}$ and $\mathbb{F}_c^{\hat{T}\mathcal{P}} = \hat{T}\mathbb{F}_c^{\mathcal{P}}$ are the edge and plane feature map in world coordinate system $\{W\}$. Features within the LiDAR range in the current frame are retained using a pass-through filter, and repeated features within the same voxel unit are downsampled using a voxel filter. The resulting feature maps for scan-to-map matching are denoted as $\mathcal{E}_c$ and $\mathcal{P}_c$, stored in KD-trees.

To find matching relationships between current frames and the local map, the correspondence between edge points ${}_c^{\mathcal{E}}p_i$ in $\mathbb{F}_c^{\mathcal{E}}$ and plane features ${}_c^{\mathcal{P}}p_i$ in $\mathbb{F}_c^{\mathcal{P}}$ are established. This correspondence is utilized in an optimization problem that aims to estimate the optimal transformation of the current frame in the world coordinate system. The optimization problem's cost function is defined as the point-to-edge distance $C_{\mathcal{E}}(\hat{T}_c)$ and the point-to-plane distance $C_{\mathcal{P}}(\hat{T}_c)$, expressed as:

$$C_{\mathcal{E}}(\hat{T}_c) = \frac{(\hat{T}_c\,{}_c^{\mathcal{E}}p_i - {}_c^{\mathcal{E}}p_\alpha) \times (\hat{T}_c\,{}_c^{\mathcal{E}}p_i - {}_c^{\mathcal{E}}p_\beta)}{\|{}_c^{\mathcal{E}}p_\alpha - {}_c^{\mathcal{E}}p_\beta\|}$$
$$C_{\mathcal{P}}(\hat{T}_c) = (\hat{T}_c\,{}_c^{\mathcal{P}}p_i - {}_c^{\mathcal{P}}\bar{p}_i) \cdot {}_c^{\mathcal{P}}n_i. \qquad (18)$$

here, ${}_c^{\mathcal{E}}p_\alpha$ and ${}_c^{\mathcal{E}}p_\beta$ are the two closest points to ${}_c^{\mathcal{E}}p_i$, while ${}_c^{\mathcal{P}}\bar{p}_i$ and ${}_c^{\mathcal{P}}n_i$ are the mean and normal vector of the closest plane to the point ${}_c^{\mathcal{P}}p_i$. Symbol $\cdot$ denotes the cross product, and symbol $\times$ represents the dot product.

The cost function of the optimization problem is then defined as:

$$\min_{\hat{T}_c} \sum_{{}_c^{\mathcal{E}}p_i \in \mathbb{F}_c^{\mathcal{E}}} C_{\mathcal{E}}(\hat{T}_c) + \sum_{{}_c^{\mathcal{P}}p_i \in \mathbb{F}_c^{\mathcal{P}}} C_{\mathcal{P}}(\hat{T}_c). \qquad (19)$$

The optimal pose transformation $\hat{T}_c$ is obtained by applying the Gauss-Newton iterative method to solve the nonlinear optimization problem. Finally, this transformation is used to convert the feature points of the current frame to the map and update the map data.

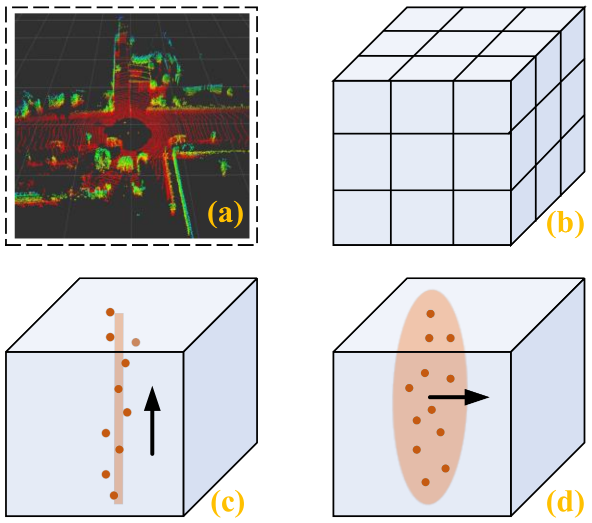

Fig. 5. The diagram of edge voxel and plane voxel feature extraction. (a) Edge and plane point features. (b) Voxel segmentation octree model. (c) Representation of voxel edge features. (d) Representation of voxel plane features. The black arrows in (c) and (d) denote the direction vector and normal vector respectively.

D. Local Optimization with Voxel Features Matching

Furthermore, although we extract accurate edge points, this is specific to feature matching on a single frame. In degraded scenes, optimizing only the current frame with the scan-to-map approach can cause the localization of the current frame to diverge. Therefore, we propose a voxel-based fixed-lag smoothing module for local optimization. Our method avoids this issue by jointly optimizing multiple keyframe factors, allowing the poses of neighboring frames to smooth the current frame when degradation occurs. This improves the robustness of SLAM in degraded scenes.

1) local keyframe optimization: when the relative rotation and translation between the current frame and the previous keyframe exceed a certain threshold, the current frame is considered a keyframe, denoted as K_k, and added to the collection of keyframes $K_n(1 \leq k \leq n)$. Subsequently, a fixed-lag smoothing optimization is performed while maintaining a feature map of cubic voxels within the range of the current keyframe. The optimal pose of the keyframe is determined as $\tilde{T}_k$ and initialized with $\hat{T}_c$, with $\tilde{T}_k = \hat{T}_c$. The optimal keyframe pose $\boldsymbol{T} = [\tilde{T}_1, \tilde{T}_2, \cdots \tilde{T}_k]^T$ is obtained by minimizing a certain criterion:

$$\min_{\boldsymbol{T}} \left\{ E_p + \sum_{k\in n} \sum_{i\in \mathbb{K}_k^{\mathcal{E}}} \left\| f_{\mathcal{E}}^i \right\|_{\Sigma_{\mathcal{E}}}^2 + \sum_{k\in n} \sum_{x\in \mathbb{K}_k^{\mathcal{P}}} \left\| f_{\mathcal{P}}^x \right\|_{\Sigma_{\mathcal{P}}}^2 \right\}. \tag{20}$$

where $\mathbb{K}_k^{\mathcal{E}}$ and $\mathbb{K}_k^{\mathcal{P}}$ are used to represent the k-th keyframe features. $f_{\mathcal{E}}^i$ and $f_{\mathcal{P}}^x$ are the residuals associated with the LiDAR i-th edge voxel and x-th plane voxel of k-th keyframe, and $\Sigma_{\mathcal{E}}$ and $\Sigma_{\mathcal{P}}$ are their corresponding covariances. These will be described by formulas (21) and (22) in detail in *Sec.III-D-3)*. E_p denotes the marginalization factor. The nonlinear objective function is solved using the efficient incremental optimization solver ISAM2[21]. It is worth noting that nonlinear optimization and marginalization methods are performed repeatedly in fixed-lag smoothing.

2) Voxel-based Geometry Feature Extraction: Unlike feature-based local optimization methods, we adopt voxel-based feature matching to improve efficiency. This method utilizes the structural representation of voxel to save computation time. Once the odometry obtains a relatively accurate pose, we can perform voxel feature matching more rapidly between keyframes. Therefore, by constructing and correlating voxel features, we can quickly solve the pose estimation between keyframes.

we add K_k in the keyframe map K_m to ensure accurate matching of geometric features. To enhance the efficiency of keyframe optimization, we propose a voxel-based method for extracting and matching geometric features. Due to the relatively small number of geometric features compared to point features, and the construction of the K-Dimensional tree (KD-tree) only using centroids, our approach significantly reduces computational time. We categorize K_k and K_m into four types based on their edge and plane feature points. Advanced geometric feature extraction based on voxel analysis is then applied to each type of point cloud set, and edge-to-edge voxel residuals and plane-to-plane voxel residuals are used for refining keyframe pose.

To convert a point cloud into a higher-level voxel feature representation, we follow a process called voxel feature extraction. This involves using PCA to calculate the centroid and covariance of the occupied voxel. The point cloud data is divided into uniform voxels, with a voxel size of v_s. In order to perform voxel feature extraction, we only consider voxels that contain more than v_n point clouds, as shown in Fig. 5. The properties of a voxel v_i can be determined using the set of point clouds within it. These properties can be expressed:

$$\begin{gathered} v_i = \{N_{v_i}, \bar{V}_i, \lambda_0, \lambda_1, \lambda_2, \mathfrak{N}_0, \mathfrak{N}_1, \mathfrak{N}_2\} \\ N_{v_i} = l \\ \bar{V}_i = \frac{1}{l}\sum_{j=1}^{l} p_j^{v_i} \\ \sigma_i = \frac{1}{l-1}\sum_{j=1}^{l} (p_j^{v_i} - \bar{V}_i)(p_j^{v_i} - \bar{V}_i)^T \\ \lambda_0 < \lambda_1 < \lambda_2 \\ \sigma_i \mathfrak{N}_i = \lambda_i \mathfrak{N}_i. \end{gathered} \tag{21}$$

where N_{v_i} represents the number of point clouds within v_i, $\bar{V}_i$ is the centroid, σ_i represents the covariance with eigenvalue $\lambda_0, \lambda_1, \lambda_2$ and $\mathfrak{N}_i$ is eigenvector associated to λ_i. We perform voxelization separately for the edge feature points of K_k and the plane feature points of K_m. Voxel classification is based on eigenvalues, where if the maximum eigenvalue is greater than twice the minimum eigenvalue and the subminimum eigenvalue, the voxel v_i is considered an edge voxel feature

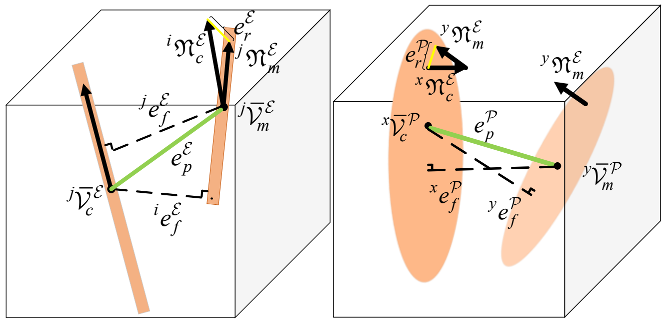
${}^{i}\mathfrak{N}_c^{\mathcal{E}}$
$e_r^{\mathcal{E}}$
${}^{j}\mathfrak{N}_m^{\mathcal{E}}$
${}^{j}e_f^{\mathcal{E}}$
${}^{j}\bar{V}_m^{\mathcal{E}}$
$e_p^{\mathcal{E}}$
${}^{j}\bar{V}_c^{\mathcal{E}}$
${}^{i}e_f^{\mathcal{E}}$
${}^{y}\mathfrak{N}_m^{\mathcal{E}}$
$e_r^{\mathcal{P}}$
${}^{x}\mathfrak{N}_c^{\mathcal{E}}$
${}^{y}\mathfrak{N}_m^{\mathcal{E}}$
${}^{x}\bar{V}_c^{\mathcal{P}}$
$e_p^{\mathcal{P}}$
${}^{y}\bar{V}_m^{\mathcal{P}}$
${}^{x}e_f^{\mathcal{P}}$
${}^{y}e_f^{\mathcal{P}}$

Fig. 6. The diagram of associative matching of edge and plane voxel features. green, yellow, and orange lines indicate edge and plane voxel matching constraints. Thick orange lines and plane indicate edge and plane voxel features, and the thick black lines indicate direction and normal vectors.

$v_i^{\mathcal{E}}$. After voxel-based feature extraction, the properties of edge voxel on K_k and K_m can be represented as:

$$\begin{cases} \mathbb{K}_k^{\mathcal{E}} = \{{}^{\mathcal{E}}_{k}v_1, \cdots {}^{\mathcal{E}}_{k}v_i, \cdots {}^{\mathcal{E}}_{k}v_s\} \\ \mathbb{K}_m^{\mathcal{E}} = \{{}^{\mathcal{E}}_{m}v_1, \cdots {}^{\mathcal{E}}_{m}v_j, \cdots {}^{\mathcal{E}}_{m}v_t\} \end{cases}. \tag{22}$$

where s represents the number of edge voxel in K_k, and ${}^{\mathcal{E}}_{c}v_i$ is i-th voxel $(1 \le i \le s)$ in K_k. t represents the number of edge voxel in K_m, and ${}^{\mathcal{E}}_{m}v_j$ is j-th voxel $(1 \le j \le t)$ in K_m.

Similarly, the eigenvalue distribution of plane voxels usually contains two large and one small. If both the maximum and the sub-maximum eigenvalues are greater than twice the minimum eigenvalues, the voxel v_i is considered as a plane voxel $v_i^{\mathcal{P}}$. After voxel-based feature extraction, the properties of plane voxel on K_k and K_m can be represented as:

$$\begin{cases} \mathbb{K}_k^{\mathcal{P}} = \{{}^{\mathcal{P}}_{k}v_1, \cdots {}^{\mathcal{P}}_{k}v_x, \cdots {}^{\mathcal{P}}_{k}v_p\} \\ \mathbb{K}_m^{\mathcal{P}} = \{{}^{\mathcal{P}}_{m}v_1, \cdots {}^{\mathcal{P}}_{m}v_y, \cdots {}^{\mathcal{P}}_{m}v_q\} \end{cases}. \tag{23}$$

where p represents the number of plane voxel in K_k, and ${}^{\mathcal{P}}_{k}v_x$ is x-th voxel $(1 \le x \le p)$ in K_m. q represents the number of plane voxel in K_m, and ${}^{\mathcal{P}}_{m}v_y$ is y-th voxel $(1 \le y \le q)$ in K_m.

3) Voxel-based Geometry Feature Matching: In this subsection, the centroids of the edge voxels and plane voxels in K_m are used to construct the KD-tree, and the direction vectors of the edge voxels and the normal vectors of the plane voxels are used to select high-quality geometric constraints. According to the pose $\hat{T}_k$ of K_k, we can transform $\mathbb{K}_k^{\mathcal{E}}$ and $\mathbb{K}_k^{\mathcal{P}}$ on K_k to $\mathbb{K}_m^{\mathcal{E}}$ and $\mathbb{K}_m^{\mathcal{P}}$ on K_m. For each edge voxel ${}^{\mathcal{E}}_{k}v_i$ and each plane voxel ${}^{\mathcal{P}}_{k}v_x$ in K_k, the edge-to-edge and plane-to-plane residuals can be defined as:

$$f_{\mathcal{E}}^{i}(\hat{T}_k) = {}^{\mathcal{E}}_{k}\mathfrak{N}_i \times (\hat{T}_k {}^{\mathcal{E}}_{k}\bar{V}_i - {}^{\mathcal{E}}_{m}\bar{V}_j). \tag{24}$$

$$f_{\mathcal{P}}^{x}(\hat{T}_k) = {}^{\mathcal{P}}_{k}\mathfrak{N}_x \times (\hat{T}_k {}^{\mathcal{P}}_{k}\bar{V}_x - {}^{\mathcal{P}}_{m}\bar{V}_y). \tag{25}$$

where $f_{\mathcal{E}}^{i}(\hat{T}_k)$, $f_{\mathcal{P}}^{x}(\hat{T}_k)$ are edge-to-edge and plane-to-plane residual, respectively. ${}^{\mathcal{E}}_{k}\bar{V}_i$ is the voxel centroid in ${}^{\mathcal{E}}_{k}v_i$, and ${}^{\mathcal{E}}_{m}\bar{V}_j$ are the closest centroids to ${}^{\mathcal{E}}_{k}\bar{V}_i$, which found by KD-tree search on K_m. ${}^{\mathcal{E}}_{k}\mathfrak{N}_i$ is the direction vector in ${}^{\mathcal{E}}_{k}v_i$. ${}^{\mathcal{P}}_{k}\bar{V}_x$ is the voxel centroid in ${}^{\mathcal{P}}_{k}v_x$, and ${}^{\mathcal{P}}_{m}\bar{V}_y$ is the closest centroid to ${}^{\mathcal{P}}_{k}\bar{V}_x$, which found by KD-tree search on K_m. ${}^{\mathcal{P}}_{k}\mathfrak{N}_x$ is the normal vector of ${}^{\mathcal{P}}_{k}v_x$. Since the edge voxel and plane voxel are represented by the centroid and the vector, this step will be very efficient. As is shown in Fig. 6, to ensure a tight association based on centroid search, inspired by [22], candidate associations are bounded by the following three independent criteria:

$$\begin{cases} e_p^{\mathcal{E}} = \|\hat{T}_k {}^{\mathcal{E}}_{k}\bar{V}_i - {}^{\mathcal{E}}_{m}\bar{V}_j\| < d_p \\ e_f^{\mathcal{E}} = \|f_{\mathcal{E}}^{i}(\hat{T}_k)\| < d_f \\ e_r^{\mathcal{E}} = \min(\|\hat{R}_k {}^{\mathcal{E}}_{k}\mathfrak{N}_i + {}^{\mathcal{E}}_{m}\mathfrak{N}_j\|, \|\hat{R}_k {}^{\mathcal{E}}_{k}\mathfrak{N}_i - {}^{\mathcal{E}}_{m}\mathfrak{N}_j\|) < d_r \end{cases}. \tag{26}$$

$$\begin{cases} e_p^{\mathcal{P}} = \|\hat{T}_k {}^{\mathcal{P}}_{k}\bar{V}_x - {}^{\mathcal{P}}_{m}\bar{V}_y\| < d_p \\ e_f^{\mathcal{P}} = \|f_{\mathcal{P}}^{x}(\hat{T}_k)\| < d_f \\ e_r^{\mathcal{P}} = \min(\|\hat{R}_k {}^{\mathcal{P}}_{k}\mathfrak{N}_x + {}^{\mathcal{P}}_{m}\mathfrak{N}_y\|, \|\hat{R}_k {}^{\mathcal{P}}_{k}\mathfrak{N}_x - {}^{\mathcal{P}}_{m}\mathfrak{N}_y\|) < d_r \end{cases}. \tag{27}$$

where $e_p^{\mathcal{E}}$, $e_f^{\mathcal{E}}$, $e_r^{\mathcal{E}}$ and $e_p^{\mathcal{P}}$, $e_f^{\mathcal{P}}$, $e_r^{\mathcal{P}}$ are the centroid-to-centroid distance difference, association distance difference, and vector difference values for edge voxels and plane voxels, respectively. d_p, d_f, d_r are the threshold values for the three distance residuals. A matching relationship is visually tightly associated only if three criteria are satisfied. $\hat{R}_k$ is the rotation matrix of $\hat{T}_k$. It is worth mentioning that since the estimated direction vectors ${}^{\mathcal{E}}_{c}\mathfrak{N}_i$ and normal vectors ${}^{\mathcal{P}}_{c}\mathfrak{N}_x$ may be oriented in the opposite direction to the map associated voxels, we ensure that the minimum value of the direction vector residual $e_r^{\mathcal{E}}$ and normal vector residual $e_r^{\mathcal{P}}$ is less than d_r.

Finally, we jointly optimize the residuals from edge voxel and plane voxel feature match. we obtain the keyframe pose $\tilde{T}_k$ by seeking the minimization of the following combined residual function in a nonlinear least squares problem:

$$\min_{\tilde{T}_k}\left(\sum_{i=1}^{s} f_{\mathcal{E}}^{i}(\tilde{T}_k) + \sum_{x=1}^{p} f_{\mathcal{P}}^{x}(\tilde{T}_k)\right). \tag{28}$$

By solving nonlinear optimization, we can get the refined keyframe pose $\tilde{T}_k$.

E. Global Consistent Mapping

Globally consistent mapping takes the keyframe point clouds and keyframe poses as input, builds a global point cloud map by aligning the poses of all keyframes. Based on HDL-graph-SLAM [15], we store the poses and point clouds of keyframes in a pose graph, and the edges between two nodes are represented by poses optimized based on multi-scale plane bimodal constraints. The g2o framework [23] used to optimize the pose graph by calculating the optimal global cost function. A globally keyframe pose transformation will be obtained.

IV. Experiment

Our experiments were conducted on a machine running

Ubuntu [1]18.04 and equipped with an Intel i7 CPU, 16GB of RAM. All the algorithms were implemented in C++ and ROS was used for inter-node communication. We used the absolute trajectory error (ATE) of the pose trajectory as the evaluation metric for SLAM accuracy, comparing and evaluating the localization trajectories of different algorithms using evo tools.

A. Parameter Settings

Parameter Description		Value
edge voxel	Distance between voxel center	0.2m
	Distance between vector distance	0.2m
	Distance from points to voxel	2.0m
plane voxel	Distance between voxel center	0.2m
	Distance between vector distance	0.2m
	Distance from points to voxel	2.0m
the sliding window length		8

B. Datasets

Our framework's accuracy is demonstrated through comprehensive evaluation and analysis on four datasets: three public datasets (KITTI, Mulran, and The Newer College) and one dataset collected by us at the Southeast University campus.

The KITTI dataset [24] consists of data sequences captured from 22 diverse outdoor environments, encompassing three main environment types: urban with buildings, rural with vegetation, and highway scenes with wide roads and few surrounding features. Eleven of these sequences (00-10) include laser point cloud data and ground truth trajectory values. The raw laser point cloud data is collected at a frequency of 10 Hz using a velodyne HDL-64E LiDAR mounted on an autonomous driving platform. The ground truth trajectory, which provides accurate values, is obtained using a high precision OXTS RT GPS/INS combined navigation system.

The Mulran dataset [25] was gathered using two distinct distance sensors: a 64-beam LiDAR (Ouster OS1-64) and a radar (Navtech CIR204-H). The dataset includes data from four different environments: city, campus, riverside, and a fully planned and developing city. The ground truth trajectories are generated through simultaneous localization and mapping (SLAM) using a fiber optic gyro, a virtual reference station GPS, and an iterative closest point (ICP) algorithm. It should be noted that due to the positioning of the LiDAR and radar installations, the LiDAR has a blind spot of 70°.

The New College dataset [26] was captured by walking through Oxford New College using a handheld device at typical walking speeds. The point cloud data was generated using an Ouster OS-1 (Gen 1) 64-line LiDAR. *short_experiment* and *long_experiment* are long-time sequences used to evaluate the localization performance of our method over extended durations. The *short_experiment* has a length of 13,632 frames, while the *long_experiment* sequence has a length of 25,053 frames.

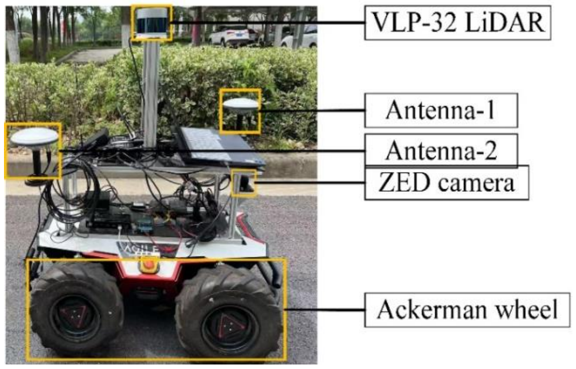

Fig. 7. Our experimental platform for sensor data collection.

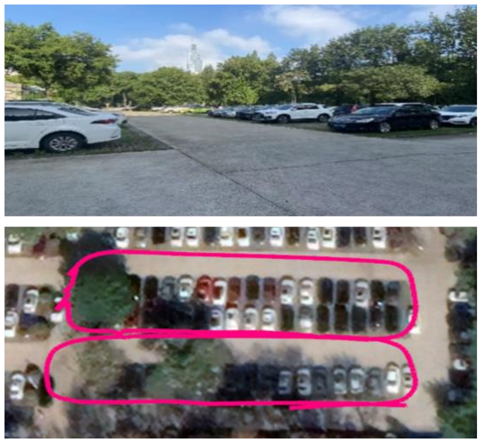
Fig. 8. The diagram of the Southeast University campus parking lot and the ground truth values of the trajectories we collected on Google Maps.

The SEU dataset was collected using a sensor suite comprising an IMU, an M600 mini multimode, a Velodyne 32-E LiDAR, a ZED2 camera with a built-in multi-frequency GNSS receiver. The trajectory truths were provided by the RTK. The composition of our sensor suite is shown in Fig. 7. The data was recorded in the parking lot on campus as shown in Fig. 8.

C. Comparison with state-of-the-art methods

We compare our method with state-of-the-art real-time 3D LiDAR SLAM methods[1]. These include A-LOAM, a classical LiDAR SLAM system that operates in real-time; LeGO-LOAM, a SLAM method that incorporates ground optimization; F-LOAM [27], which is an odometry framework that performs scan-to-map; MULLS, a multi-feature extraction and matching SLAM method; LiTAMIN2 [28], an efficient Iterative Closest Point (ICP) matching method; HDL-graph-SLAM, a SLAM algorithm based on distributed policy matching; PCA-SLAM, a SLAM scheme based on PCA feature extraction; FD-SLAM, a joint feature and distributed matching strategy SLAM system; and VoxelMap[29] , a novel LiDAR odometry based on voxel and filtering framework.

D. Ablation Study

In this section, we conducted ablation experiments to evaluate the performance of each model of the proposed

[1] https://www.bilibili.com/video/BV1ai421Z7wS/

TABLE I
ATE [m]THE RMSE COMPARISON OF PROPOSED ODOMETRY WITH OTHER METHODS ON THE KITTI AND SEU DATASET

Methods	00	01	02	03	04	05	06	07	08	09	10	Parking lots
Our VFLS	4.95	16.39	**8.02**	**0.77**	0.33	2.90	0.67	**0.54**	3.88	1.86	1.29	0.51
Our FLS	4.96	16.83	8.04	0.80	0.33	2.72	0.71	0.54	3.79	1.88	1.35	0.51
Our Odometry	4.89	18.38	8.03	0.87	0.33	2.72	0.75	0.56	3.95	1.95	1.31	0.55
VoxelMap-VFLS(our)	2.68	**3.99**	10.00	0.91	0.20	**0.96**	**0.61**	0.56	**2.46**	1.69	**0.84**	**0.14**
VoxelMap	**2.62**	4.44	10.08	0.96	**0.19**	1.06	1.09	0.69	2.58	1.96	1.51	0.30
FLOAM	5.04	19.24	8.29	0.92	0.36	3.13	0.79	0.66	4.08	**1.51**	1.37	1.66
HDL-odom	6.21	419.68	19.87	1.10	26.64	3.05	1.32	0.66	4.30	3.01	2.94	0.49
A-LOAM	4.52	18.96	117.22	0.84	0.39	2.65	0.78	0.61	3.63	**1.51**	1.44	0.57
LeGO-LOAM	5.98	192.63	53.01	0.97	0.54	4.37	0.94	1.16	3.50	2.36	2.21	0.33

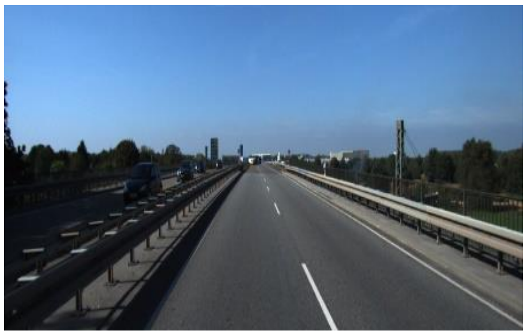
Fig. 9. The diagram of highway scenes with repetitive and degraded features(KITTI 01).

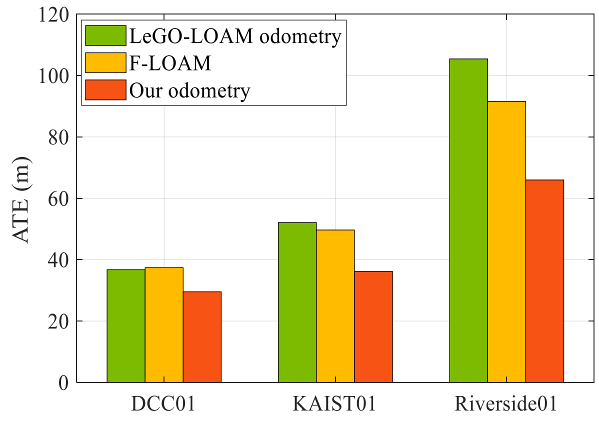

Fig. 10. Histogram of RMSE comparison of the odometry trajectory errors on the Mulran dataset.

method. Specifically, we removed angle-based feature extraction, voxel feature matching, and fixed-lag smoothing to test the effectiveness of the method. On the dataset mentioned earlier, we only used angle-based feature extraction for our odometry. Our fixed-lag smoothing (FLS) employed both angle-based feature extraction and fixed-lag smoothing. Our voxel-based fixed-lag smoothing (VFLS) incorporated angle-based feature extraction and voxel-based fixed-lag smoothing. We compared the performance of these models with other state-of-the-art algorithms. our proposed angle-based feature extraction odometry is an improvement on F-LOAM.

1) angle-based feature extraction: To validate the benefits of our proposed angle-based feature extraction method, we

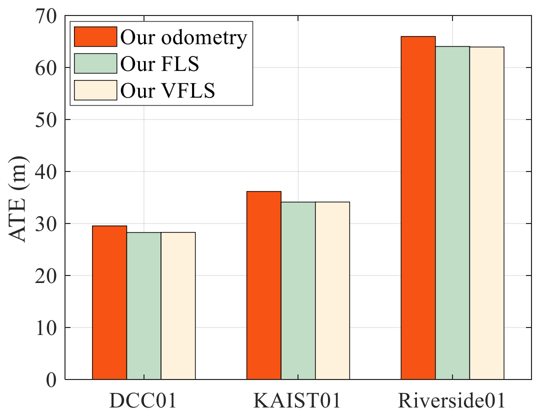

Fig. 11. Histogram of RMSE comparison of different local optimization errors on the Mulran dataset.

compared it with LeGO-LOAM and F-LOAM using the KITTI dataset. Table I presents the results of these runs. From the table, it is evident that our method outperforms the aforementioned odometries in most cases. In a degraded environment, as is shown in Fig. 9 (a frame of KITTI 01), traditional curvature-based feature extraction methods are easily affected by the viewpoint of the LiDAR. Our odometry method incorporates viewpoint-independent angle-based feature extraction, which takes into account geometric variations present in the surrounding point cloud. This approach facilitates the extraction of precise edge features even in degraded scenes.

The experimental results of the odometry on the Mulran are depicted in Fig. 10. Our odometry approach demonstrates notable enhancements in accuracy as compared to other methods employed on these datasets. Unlike KITTI datasets, the Mulran dataset encompasses scenes characterized by replicated features and prolonged time intervals. Nevertheless, our odometry method successfully extracts feature points, conducts precise feature association and matching, even in such demanding scenarios.

2) voxel-based Fixed-lag Smoothing Optimization: In this section, we conduct ablation experiments using our odometry, our FLS (fix-lag smoothing), and our VFLS (Voxel-based fixed-lag smoothing). It should be noted that our FLS and our

TABLE II
ATE [%]THE RMSE COMPARISON OF PROPOSED SLAM WITH OTHER METHODS ON THE KITTI DATASET

Methods	00	01	02	03	04	05	06	07	08	09	10	**Avg**
A-LOAM	0.12	0.78	2.31	0.14	0.10	0.12	0.06	0.09	0.11	0.09	0.15	0.370
LeGO-LOAM	0.16	7.85	1.05	0.17	0.14	0.20	0.08	0.17	0.11	0.14	0.24	0.937
F-LOAM	0.14	0.78	**0.16**	0.16	0.09	0.14	0.06	0.09	0.13	0.09	0.15	0.181
LiTAMIN2	0.03	0.56	0.21	0.14	0.18	0.16	0.06	0.07	0.07	0.12	**0.10**	0.155
MULLS-SLAM	0.54	0.62	0.69	0.61	0.35	0.29	0.29	0.27	0.83	0.51	0.61	0.510
HDL-graph-SLAM	0.02	18.28	0.38	**0.14**	0.17	0.04	0.03	0.06	**0.05**	0.35	0.25	1.797
PCA-SLAM [13]	0.07	0.42	0.18	**0.14**	0.10	0.08	0.07	**0.01**	0.13	0.11	0.16	0.134
FD-SLAM [18]	**0.02**	0.20	0.20	0.18	**0.05**	**0.03**	0.04	0.04	**0.05**	0.08	0.15	0.095
Proposed	**0.02**	**0.12**	0.19	**0.14**	0.06	0.04	**0.03**	0.06	0.07	**0.07**	0.14	**0.085**

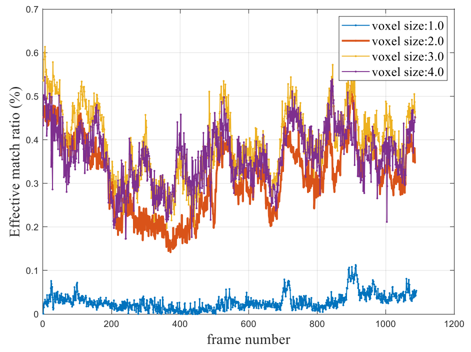

Fig. 12. Comparison figure of α-curves for four voxel sizes.

TABLE III
THE AVERAGE NUMBER OF SUCCESSFULLY MATCHED PLANES AND THE AVERAGE NUMBER OF EXTRACTED PLANES FOR THE FOUR VOXEL SIZES

Voxel size	number of successfully associated planes (avg)	number of extracted planes(avg)
1.0	34.7	1118.1
2.0	**205.3**	**623.9**
3.0	149.7	369.3
4.0	94.1	253.9

VFLS differ in their utilization of voxel-based feature matching. The trajectory accuracies of these three methods on the KITTI dataset are presented in Table I. From the table, it is evident that VFLS consistently outperforms both our odometry and FLS across most sequences, highlighting the effectiveness of our proposed voxel feature matching and fixed-lag smoothing module. Furthermore, Fig. 11 illustrates the trajectory accuracy performance of our odometry, FLS, and VFLS on the Mulran dataset, demonstrating the robustness of our fixed-lag smoothing method across diverse scenarios. Finally, in the last column of Table I, we included the performance of our proposed method and other methods on the SEU parking lot dataset. To demonstrate the effectiveness and compatibility of our proposed method in different frameworks, we incorporated voxel-based fixed-lag smoothing into the VoxelMap backend(VoxelMap-VFLS) and

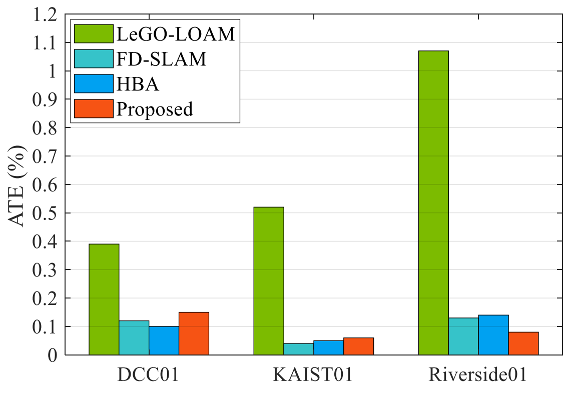

Fig. 13. Histogram of RMSE comparison of SLAM trajectory errors on the Mulran dataset.

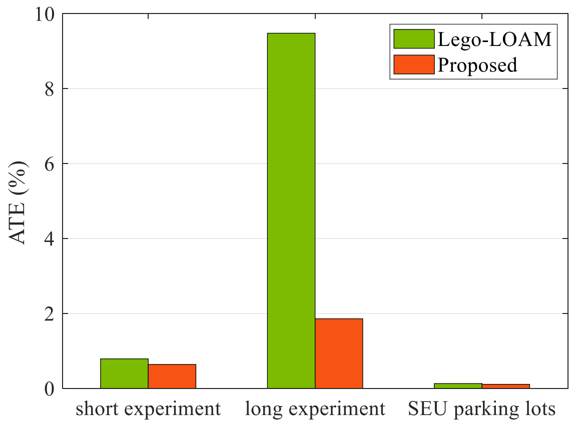

Fig. 14. Histogram of RMSE comparison of SLAM trajectory errors on the Newer College dataset.

compared it with the accuracy performance of VoxelMap. The comparison results are shown in Table I, and the results show that voxel-based fixed-lag smoothing enhances the accuracy of VoxelMap in the majority of sequences.

Furthermore, we set the success rate of plane matching as a metric and conducted comparative experiments with four different voxel sizes. The success rate of plane matching α can be expressed as:

$$\alpha = \frac{\sigma_p}{\rho_p}. \qquad (29)$$

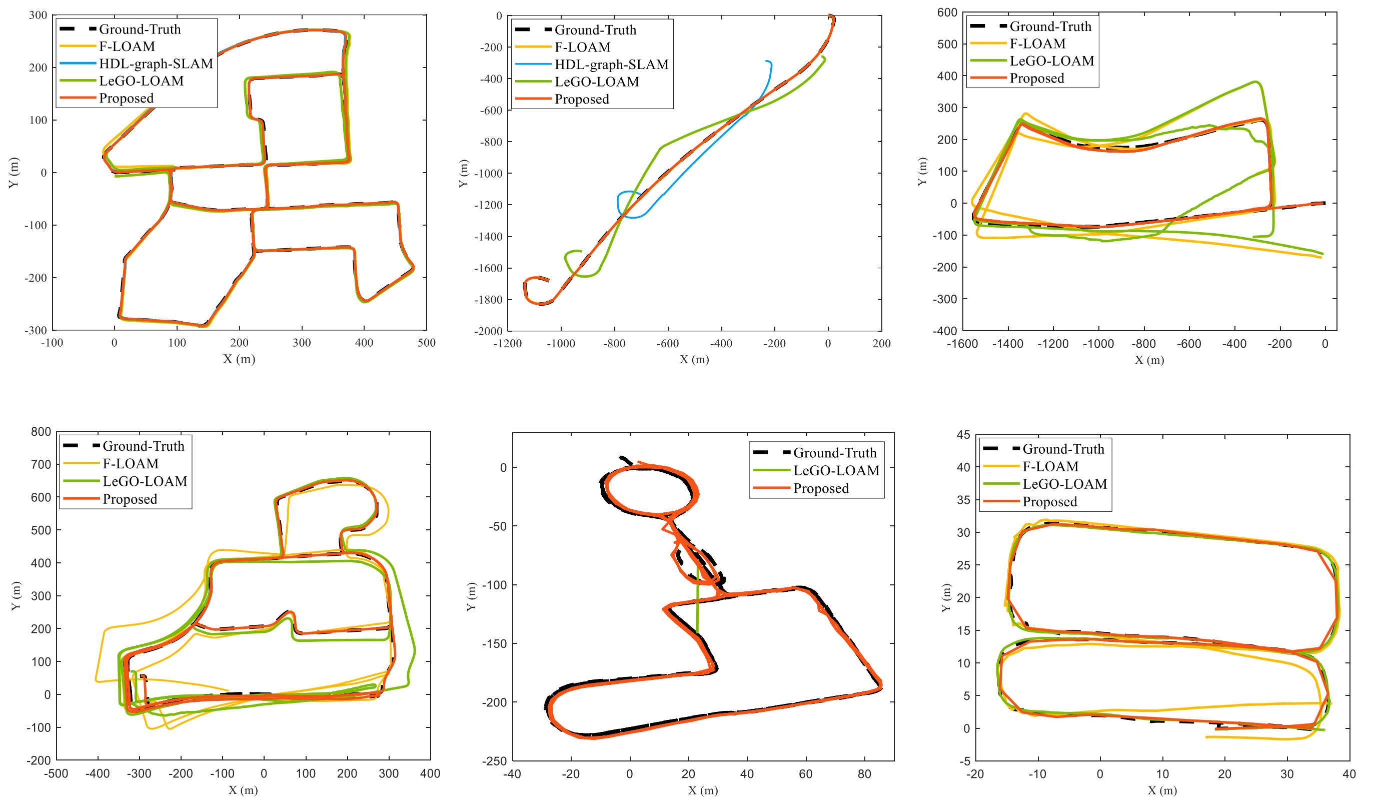

Fig. 15. Results of the SLAM trajectory comparisons of various methods using four datasets. (a) KITTI 00, (b) KITTI 01, (c) KAIST01, (d) Riverside01, (e) long_experiment, (f) SEU parking lots.

TABLE IV
THE AVERAGE TIME [ms] CONSUMPTION FOR DIFFERENT METHODS IN KITTI DATASET

Algorithms	Feature Extraction	LiDAR Odometry	Local Optimization	Total Time
A-LOAM (KITTI)	12.5	28.7	58.9	100.1
PCA-SLAM (KITTI)	51.6	45.1	97.7	194.4
FD-SLAM (KITTI)	2.7	3.4	114.4	120.5
Feature-based optimization (KITTI)	9.7	42.3	80.9	132.9
Our VFLS (KITTI)	9.7	49.0	16.6	**75.3**
FD-SLAM (Mulran)	18.53	22.38	131.37	172.3
Our VFLS (Mulran)	4.53	37.11	48.87	**90.5**

where ρ_p denotes the number of planes extracted in the current frame, and σ_p denotes the number of voxel associated plane feature successfully in the current frame. The curves of the matching success rates for four different voxel sizes are shown in Fig. 12. When the voxel sizes are 2, 3, and 4, the matching success rates are similar. Subsequently, the matching success rates and the extracted planes for the four different voxel sizes are shown in Table III. With the voxel size of 2, the maximum number of planes can be extracted while maintaining the matching success rate, and the number of successfully matched planes is also the highest. In order to balance extracting sufficient plane information and maintaining fast computational efficiency, we chose a voxel size of 2m for the plane voxels. Compared to plane voxels, the size of edge voxels can be set smaller because they represent edge or irregular regions. Smaller edge voxels can improve the accuracy of edge feature matching. We choose a voxel size of 1m for the edge voxels.

E. SLAM Accuracy Analysis

In this section, we provide a detailed comparison and evaluation of our proposed approach with state-of-the-art LiDAR SLAM algorithms. We begin with a quantitative analysis, comparing the trajectory of our method with the ground truth. Additionally, we present an evaluation of our proposed method using four datasets.

Firstly, we assess the trajectory accuracy of our methods using the KITTI dataset. Table II summarizes the trajectory accuracy for each SLAM method. As shown in the table, our algorithm demonstrates good performance, achieving an ATE result of 0.11%. Our method is comparable to FD-SLAM and higher than the SLAM accuracy of other methods. In *Sec.IV-F*, we verify that we have higher time efficiency than FD-SLAM.

TABLE V
THE AVERAGE TIME [ms] CONSUMPTION FOR FEATURE-BASED AND VOXEL-BASED LOCAL OPTIMIZATION

	feature-based local optimization			**Our VFLS**		
	①	②	③	①	②	③
00	9.47	25.26	37.07	9.34	30.42	**13.62**
01	8.64	65.92	153.19	9.21	78.71	**22.06**
02	9.81	25.08	55.7	9.72	31.38	**15.2**
03	10.01	55.61	147.95	10.09	62.79	**18.21**
04	9.63	59.84	129.92	9.6	65.67	**19.8**
05	9.92	46.00	64.99	9.86	54.32	**17.21**
06	9.96	34.01	50.25	9.94	40.01	**15.68**
07	9.14	24.82	30.7	9.21	27.72	**11.32**
08	9.89	57.36	80.46	9.75	67.92	**20.08**
09	9.82	30.25	61.81	9.86	34.78	**14.32**
10	9.93	40.69	77.3	9.93	45.11	14.87
avg	9.65	42.25	80.85	9.68	48.99	16.58
total	132.8			**75.3**		

① denotes feature extraction; ② denotes LiDAR odometry; ③ denotes local optimization

Secondly, in the Mulran dataset, a widely-used benchmark known for its structural diversity, we compare our algorithm with several state-of-the-art methods, namely LeGO-LOAM, HDL-graph-slam, FD-SLAM, and HBA [30] (see Fig. 13). Among these methods, FD-SLAM and HBA are from their respective published works. In the Riverside sequence, the repetition of structural features like trees and bridges poses a significant challenge. However, our proposed method surpasses the competition by achieving the highest accuracy in position estimation, with an ATE of 0.09%.

Lastly, we conducted tests on the *short_experiment* and *long_experiment* sequences in The Newer College dataset, as well as the SEU parking lots. Fig. 14 presents the absolute trajectory errors of our proposed method alongside other advanced algorithms. The figure reveals that our method achieves the best results in both the *short_experiment* and *long_experiment* sequences, demonstrating its high accuracy. Furthermore, when compared to LeGO-LOAM with loop detection, our algorithm also performs well in the SEU parking lots.

Fig. 15 illustrates the trajectories of the advanced and proposed methods with their respective true benchmarks. As observed in the figure, the trajectories of F-LOAM, LeGO-LOAM, and HDL-graph-SLAM deviate from the ground truth benchmark.

F. Time Efficiency

In this section, we analyze the time consumption of each component of our method. Table IV presents the runtime comparison of our method with other methods. From the table, it can be observed that our algorithm has the shortest overall runtime. In the feature extraction stage, when dealing with the issue of LiDAR viewpoint, our feature extraction algorithm takes approximately 9.7 ms without utilizing time-consuming PCA analysis in KITTI datasets. In the local optimization stage, our method outperforms others in terms of runtime, which can be attributed to the utilization of voxels. Additionally, the reason we did not validate the HBA [30] algorithm is that it is used for point cloud map alignment after pose graph optimization (PGO), requiring point cloud and initial pose inputs.

Furthermore, In Table V, we also compare the time efficiency of feature-based local optimization with voxel-based local optimization (Our VFLS). The feature extraction part of the feature-based local optimization is referenced from [16]. It can be seen that our method only takes 75.3 ms in time. The method of voxelize edge features and plane features is equivalent to dimension reduction of the point cloud data. When performing feature association, we only need to compare the differences in position of the center point, the direction difference of the normal vector, and the distance from the point to the voxel with the surrounding edge voxels and plane voxels, to efficiently achieve voxel feature association. thereby demonstrating the efficiency of voxels. In contrast, feature-based methods, in addition to having multiple matching feature points, require finding associated points through KD-tree and involve repetitive computations of distances between points to edges and points to planes, thus reducing the matching efficiency.

V. CONCLUSION

In this paper, we propose a LiDAR SLAM method that combines angle-based feature extraction and voxel-based local optimization. Our improvements only focus on feature extraction and local optimization in odometry, but applying them to complete SLAM will also enhance performance, which has been verified in the experimental section. Our approach leverages the spatial angle information of the point cloud to achieve a more accurate and stable feature point cloud. Additionally, voxel feature extraction allows us to capture the structural properties of the environment more effectively. We have demonstrated that local optimization enhances the consistency of pose estimation. Through comparative experimental analysis, our framework exhibits robustness when tested on challenging datasets. Moving forward, we aim to explore novel techniques for voxel feature extraction and map updating, as well as enhance the algorithm's accuracy and efficiency in unstructured scenes.

REFERENCES

[1] X. Li, Y. Zhou, and B. Hua, "Study of a Multi-Beam LiDAR Perception Assessment Model for Real-Time Autonomous Driving," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–15, 2021.

[2] M. Yuan, J. Shan, and K. Mi, "Deep Reinforcement Learning Based Game-Theoretic Decision-Making for Autonomous Vehicles," *IEEE Robot. Autom. Lett*., vol. 7, no. 2, pp. 818–825, Apr. 2022.

[3] D. Tran, J. Du, W. Sheng, D. Osipychev, Y. Sun, and H. Bai, "A Human-Vehicle Collaborative Driving Framework for Driver Assistance," *IEEE Trans. Intell. Transp. Syst*., vol. 20, no. 9, pp. 3470–3485, Sep. 2019.

[4] Y. Cong *et al.*, "3D-CSTM: A 3D continuous spatio-temporal mapping method," *ISPRS J. Photogramm. Remote Sens.*, vol. 186, pp. 232–245, Apr. 2022.

[5] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robot. Autom. Lett*., vol. 8, no. 2, pp. 1029–1036, Feb. 2023.

[6] 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.

[7] X. Liu et al., "Brain-like position measurement method based on improved optical flow algorithm," ISA Trans., p. S0019057823004123, Sep. 2023.

[8] Youngmin Park, V. Lepetit, and Woontack Woo, "Extended Keyframe Detection with Stable Tracking for Multiple 3D Object Tracking," *IEEE Trans. Vis. Comput. Graph*., vol. 17, no. 11, pp. 1728–1735, Nov. 2011.

[9] L. Pan, K. Ji, and J. Zhao, "Tightly-Coupled Multi-Sensor Fusion for Localization with LiDAR Feature Maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5215–5221.

[10] B. Peng, H. Xie, and W. Chen, "ROLL: Long-term robust lidar-based localization with temporary mapping in changing environments," 2022, *arXiv*:2203.03923.

[11] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in realtime." in *Robot., Sci. Syst*, vol. 2, no. 9, pp. 1-9, 2014.

[12] 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. (IROS)*, Oct. 2018, pp. 4758–4765.

[13] S. Guo, Z. Rong, S. Wang, and Y. Wu, "A LiDAR SLAM with PCA-based feature extraction and two-stage matching," *IEEE Trans. Instrum.Meas*., vol. 71, pp. 1–11, 2022.

[14] Z. Kang, J. Yang, R. Zhong, Y. Wu, Z. Shi, and R. Lindenbergh, "Voxel-Based Extraction and Classification of 3-D Pole-Like Objects from Mobile LiDAR Point Cloud Data," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*., vol. 11, no. 11, pp. 4287–4298, Nov. 2018.

[15] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst*., vol. 16, no. 2, p.17298814198 4153, Mar. 2019.

[16] 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. (IROS)*, Oct. 2020, pp. 5135–5142.

[17] D. Wisth, M. Camurri, S. Das, and M. Fallon, "Unified Multi-Modal Landmark Tracking for Tightly Coupled Lidar-Visual-Inertial Odometry," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.

[18] D. Chang, R. Zhang, S. Huang, M. Hu, R. Ding, and X. Qin, "WiCRF: Weighted Bimodal Constrained LiDAR Odometry and Mapping with Robust Features," *IEEE Robot. Autom. Lett*., vol. 8, no. 3, pp. 1423–1430, Mar. 2023.

[19] J. Wang, M. Xu, G. Zhao, and Z. Chen, "Feature- and Distribution-Based LiDAR SLAM With Generalized Feature Representation and Heuristic Nonlinear Optimization," *IEEE Trans. Instrum. Meas*., vol. 72, pp. 1–15, 2023.

[20] C. Shu and Y. Luo, "Multi-Modal Feature Constraint Based Tightly Coupled Monocular Visual-LiDAR Odometry and Mapping," *IEEE Trans. Intell. Veh.*, vol. 8, no. 5, pp. 3384–3393, May 2023.

[21] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 3281–3288.

[22] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, "STD: Stable Triangle Descriptor for 3D place recognition," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 1897–1903.

[23] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2013, pp. 2100–2106.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," Int. J. Robot. Res., vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[25] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim, "MulRan: Multimodal Range Dataset for Urban Place Recognition," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 6246–6253.

[26] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The Newer College Dataset: Handheld LiDAR, Inertial and Vision with Ground Truth," in *Proc IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 4353–4360.

[27] 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. (IROS)*, Sep. 2021, pp. 4390–4396.

[28] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN2: Ultra-Light LiDAR-based SLAM using Geometric Approximation applied with KL-Divergence," in *Proc IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, May 2021, pp. 11619–11625.

[29] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robot. Autom. Lett*., vol. 7, no. 3, pp. 8518–8525.

[30] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-Scale LiDAR Consistent Mapping Using Hierarchical LiDAR Bundle Adjustment," *IEEE Robot. Autom. Lett*., vol. 8, no. 3, pp. 1523–1530.

Nuo Li received the B.S. degree in automation from the Wuhan Institute of Technology, Wuhan, China, in 2020. He is currently pursuing the Ph.D. degree in instrument science and technology with Southeast University, Nanjing, China. His research interests include Unmanned vehicle navigation, visual-LiDAR fusion localization and map construction.

Yiqing Yao received the Ph.D. degree in 2018 from Southeast University, Nanjing, China, where she is currently an associate professor in the School of Instrument Science and Engineering, Southeast University.

Her research interests include vehicle positioning, inertial navigation and its integration with other navigation systems.

Xiaosu Xu received the Ph.D. degree in instrument science and engineering from Southeast University, Nanjing, China, in 1991.

He is currently a professor with the School of Instrument Science and Engineering, Southeast University. His research interests include the integrated navigation system and intelligent sensor fusion mechanism.

Yiyang Peng received the B.S. degree in measurement and control technology and instrumentation from Nanchang Hangkong University, Nanchang, Jiangxi, in 2022. He is currently pursuing the M.S. degree in navigation, guidance and control with Southeast University. His research interests include visual and lidar navigation and its integration.

Zijian Wang received the M.S. degree from China University of Mining and Technology. He is currently working toward the Ph. D. degree in engineering from Southeast University, Nanjing. His current research interests include robot localization and 3-D vision.

Hongyu Wei received the Ph. D degree in Instruments Science and Technology from Southeast University in 2023. She is currently a lecturer at the Institute of Artificial Intelligence, Shanghai University. Her current research interests include multisensory fusion, ground unmanned platform navigation, visual and inertial SLAM.