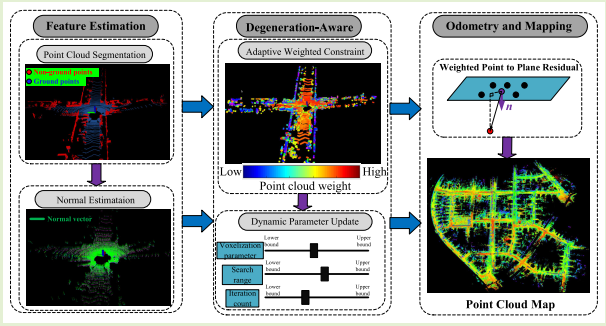
NA-LOAM: Normal-Based Adaptive LiDAR Odometry and Mapping

Fengli Yang, Wangfang Li, and Long Zhao

***Abstract*—Light detection and ranging (LiDAR)-based simultaneous localization and mapping (SLAM) exhibits excellent performance in large-scale real-world scenarios and is widely applied in robot navigation systems. However, the adaptability of LiDAR-based SLAM algorithms in different environments remains a challenge. The fixed parameter settings and local information-based weighting strategies can influence the performance and reliability of LiDAR-based SLAM algorithms across various environments and application scenarios. To address the above issues, this article introduces a method based on point cloud normals to evaluate the degree of environmental degradation. This approach adaptively weights point clouds and dynamically adjusts optimization hyperparameters. Specifically, we first utilize distinct lookup tables for ground and nonground points based on the scanning structure of the LiDAR, allowing for the rapid computation of the point cloud normals. Subsequently, we used the weighted covariance matrix (WCM) of normal vectors to assess the degree of environmental degradation. Finally, based on the degradation level, we dynamically adjust optimization hyperparameters and compute the weight of each point. The proposed method demonstrates higher accuracy and robustness in diverse environments through validation on the KITTI benchmark and real-world scenarios.**



***Index Terms*—Light detection and ranging (LiDAR) odometry, mapping, point cloud normals, simultaneous localization and mapping (SLAM).**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) stands as a pivotal technology in the field of robotics, allowing mobile robots to autonomously navigate and map unknown environments. Unlike vision sensors, light detection and ranging (LiDAR), as an active sensor, measures object depth by emitting laser pulses and capturing the reflected light, thus remaining unaffected by environmental factors such as lighting conditions. Additionally, the high accuracy and reliability of LiDAR sensors make LiDAR-based SLAM have significant potential in practical applications [1], [2], [3].

In recent years, driven by advancements in laser sensor technology, various new types of LiDAR sensor have emerged, primarily including mechanical LiDAR and solid-state LiDAR [4]. Mechanical LiDAR typically employs rotating laser emitters and receivers to obtain 3-D information about the surrounding environment through rotational scanning. Therefore, compared to solid-state LiDAR, it has a wider field of view and finds wide applications. Therefore, the focus of this article is on mechanical LiDAR-based SLAM.

For LiDAR-based SLAM technology, point cloud registration is a crucial component that determines positioning accuracy and map quality. Currently, the most well-known method for point cloud registration is the iterative closest point algorithm (ICP) [5]. Most popular LiDAR SLAM techniques are based on ICP and its variants [6], [7], utilizing iterative error minimization techniques to perform point cloud registration and estimate the correct pose difference between two point clouds [8]. However, traditional LiDAR odometry (LO) [9], [10] typically relies on default hyperparameter configurations and local information based weighting strategies, which may result in a lack of adaptability to different environments and inadequate handling of noisy or unreliable sensor data [11]. Subsequently, the number and quality of the correspondence become degraded. In situations lacking sufficient constraints, the optimization problem may become ill-conditioned, leading to suboptimal solutions and potential errors in pose

Manuscript received 2 July 2024; revised 17 August 2024; accepted 17 August 2024. Date of publication 27 August 2024; date of current version 2 October 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 42274037, in part by the Aeronautical Science Foundation of China under Grant 2022Z022051001, and in part by the National Key Research and Development Program of China under Grant 2020YFB0505804. The associate editor coordinating the review of this article and approving it for publication was Dr. Vanita Arora. *(Corresponding author: Long Zhao.)*

The authors are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100091, China (e-mail: Fengli_Yang@buaa.edu.cn; liwangfangx@gmail.com; buaa_dnc@buaa.edu.cn).

Digital Object Identifier 10.1109/JSEN.2024.3446998

estimation [12]. Consequently, enhancing the adaptive ability of LO based on LiDAR point cloud information is a research-worthy task [13].

Currently, numerous methods [14], [15], [16], [17], [18] have been proposed to address this issue. On one hand, some methods [14], [15], [16] utilize local information from point clouds for sampling and weighting, ensuring the convergence of optimization problems. Nevertheless, the main issue with this approach is that it only considers the local information of the point cloud and does not assess the overall distribution of the point cloud. On the other hand, some methods [17], [18] take into account that in degenerate environments, the distribution of normal vectors is often unbalanced. Hence, these methods assess whether the current scan is degraded by computing the point cloud's normal vectors. Furthermore, the hyperparameters are dynamically adjusted according to the degradation level. However, these methods do not consider how to weight the point cloud to balance the subsequent optimization problem. In addition, these methods lie in the use of principal component analysis (PCA) to calculate the normal vectors. The additional computational overhead renders this method unsuitable for real-time applications [19]. To rapidly compute the normal vectors of point clouds, some methods leverage the scanning characteristics of mechanical LiDAR sensors. Badino et al. [19] assume local depth consistency in point clouds and propose a fast approximate least squares (FALS) method to estimate the normal vectors of each point based on precomputed lookup tables. However, for sparse point clouds, the assumption of local depth consistency may become invalid. This is particularly evident for ground points, where the installation orientation of the LiDAR sensor is typically parallel to the ground, and the vertical resolution of the LiDAR is significantly lower than its horizontal resolution. Consequently, adjacent points may exhibit significant difference in depth, while their height difference are typically small.

In light of the aforementioned challenges, this article introduces a real-time Normal-based adaptive LO and mapping (NA-LOAM) method. First, for ground and nonground points, we compute the normal vectors for each point using precomputed height and depth lookup tables, and weight the normal vectors based on the variance of local depth or height for each point. Next, we analyze the distribution of normal vectors and used the weighted covariance matrix (WCM) of normal vectors to assess the degree of degeneration. Finally, we dynamically adjust optimization parameters and weight matching point pairs based on the degradation level. The proposed algorithm dynamically adjusts parameters and weighting strategies based on environmental changes, thereby enhancing adaptability and stability across different scenarios.

The main contributions of this work are threefold.

1) This article proposes a novel ground-based FALS (Ground-FALS) estimator for ground normal vectors. It considers the approximate equality of heights for ground points rather than depths. This approach allows for a more accurate estimation of ground point normal vectors while ensuring real-time performance.
2) By considering the distribution of point cloud normal, this article used the WCM of normal vectors to assess the degree of degradation in the current scan. Based on this assessment, point clouds are weighted and optimization parameters are flexibly adjusted.
3) In testing with the KITTI benchmark dataset and real-world environments, compared to existing state-of-the-art approaches, our proposed method demonstrates accurate detection of degraded scenes, showing higher flexibility and stability.

The remainder of this article is organized as follows. Related work is reviewed in Section II. Section III describes the system architecture of this article. Section IV provides a detailed description of the process of dynamically adjusting parameters and weighting point clouds in our algorithm. Section V describes the experimental setup and presents the results obtained from the KITTI datasets and real experiments. Finally, Section VI summarizes the conclusion of the proposed method.

II. Related Works

A. LiDAR Odometry

LiDAR sensors leverage the high directionality, monochromaticity, and brightness of laser light to achieve noncontact long-distance measurements, thereby rapidly obtaining high-precision point clouds [20]. Consequently, LiDAR-based odometry algorithms have received widespread research attention in recent years [9], [10], [21].

LOAM [9], a classical method for LO and mapping, effectively categorizes point clouds into edge and planar points during the scan matching process, enabling the efficient capture of geometric features in the environment. By performing scan matching on feature point clouds from two consecutive time frames, LOAM can estimate the trajectory of the robot in real-time. Lego-LOAM [10] is an enhanced version of LOAM, achieving more efficient LO through point cloud segmentation and ground optimization methods. This method utilizes ground and point cloud segmentation to eliminate point cloud noise. Additionally, the algorithm employs an inertial measurement unit (IMU) to assist in point cloud registration. In the pose optimization module, the odometry factors and loop closure factors are added to the factor graph, and optimization results are obtained using the GTSAM library [22]. F-LOAM [21] is a fast LO method designed to meet real-time requirements by achieving higher processing speed through direct registration of map point clouds. Furthermore, the method balances the matching process by weighting point clouds based on curvature information surrounding them. RF-LOAM [23] addresses the issue of dynamic obstacles in urban environments by proposing an FA-RANSAC method based on feature information and adaptive thresholds for obstacle filtering. Subsequently, the sensor's pose is estimated using static feature points. While these methods perform well in specific scenarios, they lack adaptability to different environments. To address this limitation, we propose a degradation detection method to dynamically adjust optimization hyperparameters and weight point cloud features, enhancing the flexibility and stability of the method.

B. Adaptive Methods

In the real world, environments vary greatly, and LO may operate in diverse settings such as indoor, outdoor, or urban street environments. However, neglecting adaptability may result in algorithm failure in specific environments or application scenarios. For instance, using fixed parameter configurations designed for indoor environments in outdoor settings may lead to ineffective estimation of the robot's position and orientation, thus reducing the accuracy of localization and navigation. In response to this issue, several methods [10], [12], [14] have been proposed.

1) Weight-Based Methods: These methods balance the contribution of points in optimization by weighting the point cloud, thereby enhancing the robustness and accuracy of the algorithm. Gelfand et al. [16] discovered that selecting too many points from featureless regions of the data could lead to slow convergence, erroneous pose estimation, or even divergence. However, by choosing appropriate samples, convergence is possible for ICP. To balance the optimization process, F-LOAM [21] and Li et al. [15] consider dynamically weighting point clouds using curvature and intensity information, respectively. However, these methods only consider the local information of point clouds and do not consider the degradation of the entire point cloud.

2) Scenario-Based Methods: These methods dynamically adjust parameters by assessing the degradation level of the scene, ensuring the algorithm's performance across various environments. Shi et al. [24] addressed the issue of information scarcity in degraded environments such as long corridors and large halls for 2-D LiDAR. They proposed a degeneration-aware LO based on point cloud normal vectors. This method dynamically adjusts the motion weight to achieve adaptability to different environments. Similarly, Ji et al. [17] extend the method to 3-D LiDAR, enhancing adaptability to different environments by adjusting point cloud filtering parameters. AdaLIO [18] applies a similar idea, designing two sets of parameters based on degradation conditions to increase their corresponding quantities and avoid under-constraints. However, a major drawback of the above methods is that they calculate point cloud normal vectors based on PCA, and the additional computational overhead makes these methods unsuitable for real-time applications. Furthermore, these approaches adjust the entire point cloud without considering the contribution of individual point.

C. Point Cloud Normal Estimation

The normals of point clouds play a crucial role in LO. By analyzing the normals of point clouds, LO can perform data association and matching to determine the robot's position and orientation in the environment. Additionally, normals aid in distinguishing between ground and nonground points, identifying obstacles, and detecting irregular structures in the environment, thereby further enhancing the robustness and performance of the LO. The methods for estimating point cloud normals are primarily based on local fitting. While these methods provide reliable normal estimations to a certain extent, they may encounter challenges in terms of computational complexity and accuracy when dealing with complex scenes and large-scale point clouds [25].

The literature [19] discusses several least squares formulations for estimating surface normals and compares their computational efficiency. Additionally, a novel approach is proposed to obtain surface normals by calculating the derivatives of the surface from a spherical range images. Leveraging the characteristics of mechanical scanning LiDAR, LOG-LIO [25] proposes a ring-FALS method to estimate the normal for each point. This method involves preestablishing a lookup table. During subsequent normal estimation, it only requires depth information for the rapid estimation of normals for each point. However, for ground point clouds, height consistency of LiDAR points within a local range is more reasonable than depth consistency. Motivated by the aforementioned methods, in this article, the ground and nonground points after segmentation are processed using two precomputed lookup tables to rapidly compute their respective normals.

III. System Overview

The system framework of our proposed approach, NA-LOAM, is illustrated in Fig. 1. The entire system is mainly divided into two parts: point cloud adaptive preprocessing and LO and mapping. The main innovation of this article is primarily reflected in the adaptive preprocessing. Its role is to evaluate the degradation of the scene based on the normal vectors of the current point cloud. Subsequently, it adaptively adjusts optimization parameters and dynamically weights the point cloud, providing more uniform and accurate point cloud data for the subsequent LO module. This ensures that the optimization does not diverge in the direction of weak constraints.

The adaptive preprocessing module consists of two core components: feature estimation and degeneration-aware. The feature estimation module first segments the LiDAR point cloud into ground and nonground points. Subsequently, ground and nonground points are projected into range and height images, respectively. Finally, the normal vectors for each LiDAR point are efficiently calculated using precomputed query tables. The degeneration-aware module first calculates the WCM based on the normal vectors of the point cloud, determining the degree of degeneration by examining the magnitude of its eigenvalues. Then, it adaptively adjusts optimization parameters based on the degradation level. At last, the point cloud is weighted based on the differences between the normal vector of each point and the constraint direction.

The LO and mapping module, based on the information from the point cloud and optimization parameters, utilizes point-to-plane constraints to determine the relative transformation between the current frame and the local map. Furthermore, keyframes are detected based on pose changes, and the point clouds of these keyframes are registered into the global map.

IV. Methodology

A. Feature Estimation Module

In general, a 3-D LiDAR point $\boldsymbol{p}$ can be represented by a spherical coordinate system as

$$\boldsymbol{p} = r\boldsymbol{v} = r\begin{bmatrix}\cos\theta\cos\varphi\\ \sin\theta\cos\varphi\\ \sin\varphi\end{bmatrix} \tag{1}$$

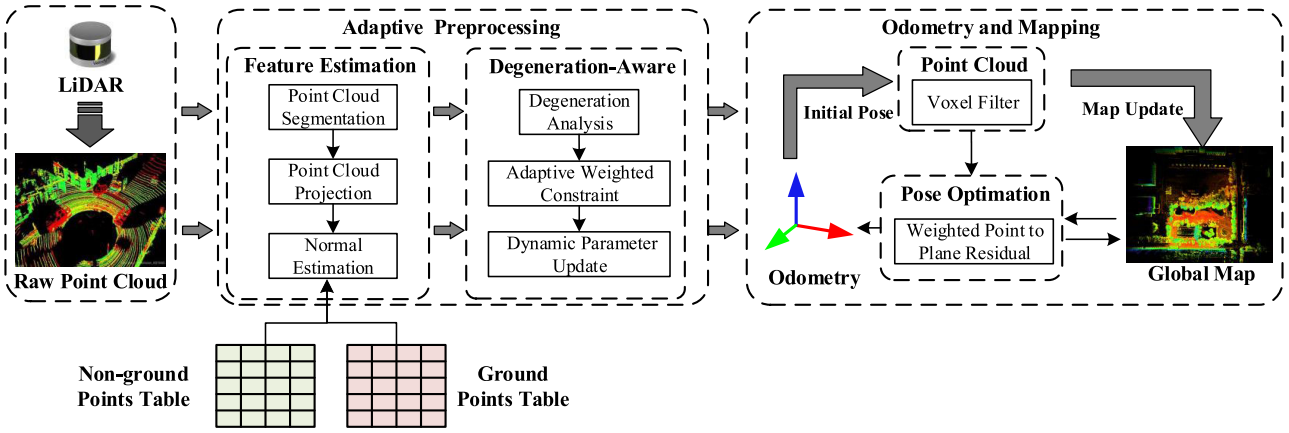


Fig. 1. System framework of the proposed NA-LOAM.

where r is the range, θ the azimuth, and φ the elevation component. $\boldsymbol{v} = \begin{bmatrix}\cos\theta\cos\varphi & \sin\theta\cos\varphi & \sin\varphi\end{bmatrix}^{\mathrm{T}}$ is a unit vector that encompasses the spatial information of the LiDAR point.

Given a subset of points $\{\boldsymbol{p}_i, i = 1, 2, \ldots, k\}$ near a LiDAR point $\boldsymbol{p}$, the normal vector of its tangent plane can be estimated using the least squares method

$$e = \sum_{i=1}^{k} \left(\boldsymbol{p}_i{}^{\mathrm{T}}\boldsymbol{n} - d\right)^2, \quad \text{s.t. } \|\boldsymbol{n}\|_2 = 1 \tag{2}$$

where $\boldsymbol{n}$ is the unit normal vector of the plane, and d represents the intercept of the plane. By substituting (1) into (2) and dividing by d^2, we can obtain

$$e = \sum_{i=1}^{k} r_i^2 \left(\boldsymbol{v}_i^{\mathrm{T}}\bar{\boldsymbol{n}} - \frac{1}{r_i}\right)^2 \tag{3}$$

where $\bar{\boldsymbol{n}} = \boldsymbol{n}/d$ is defined up to a scale factor. In ring-FALS [25], it assumes that the depth of the point cloud within a small region is approximately consistent, i.e., $r \approx r_i$. Therefore, (3) can be rewritten as

$$\hat{e} = \sum_{i=1}^{k} \left(\boldsymbol{v}_i^{\mathrm{T}}\hat{\boldsymbol{n}} - \frac{1}{r_i}\right)^2 \tag{4}$$

where $\hat{\boldsymbol{n}}$ is the approximate normal. The closed solution of $\hat{\boldsymbol{n}}$ can be obtained by solving the following equation:

$$\hat{\boldsymbol{n}} = \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{b}} \tag{5}$$

where $\hat{\boldsymbol{M}} = \sum_{i=1}^{k} \boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}}$ can be precomputed, $\hat{\boldsymbol{b}} = \sum_{i=1}^{k} \boldsymbol{v}_i/r_i$.

Ground vehicles inevitably come into contact with the ground. Therefore, ground points typically occupy the majority of the entire point cloud. This makes ground segmentation a crucial component of SLAM systems. The Patchwork [26] is a classical ground segmentation algorithm. The algorithm first divides the point cloud into multiple local regions, where points within each region are considered to potentially belong to the same plane. Then, for each local region, local plane fitting is performed to estimate the plane's normal vector and distance from the origin. Next, the fit planes are evaluated to determine if they exhibit ground characteristics, such as by calculating residuals or assessing plane consistency. Based on the evaluation results, local regions that match ground features are selected and identified as ground points. In this article, patchwork is employed to segment the point cloud into ground and nonground points.

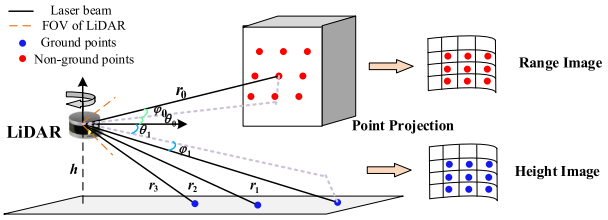


Fig. 2. Projection model of LiDAR on ground vehicle.

As shown in Fig. 2, for ground vehicles, the installation of the LiDAR typically results in a larger incidence angle for ground points. Additionally, the vertical resolution of the LiDAR is significantly lower than its horizontal resolution, causing adjacent ground points to be more distant and exhibit a greater depth difference. This condition deviates from the assumption of ring-FALS. However, in general, the height variation of ground points is small. Therefore, (1) can be rewritten as follows:

$$\boldsymbol{p} = h\tilde{\boldsymbol{v}} = h\begin{bmatrix}\cos\theta\cos\varphi/\sin\varphi \\ \sin\theta\cos\varphi/\sin\varphi \\ 1\end{bmatrix} \tag{6}$$

where $h = r\sin\varphi$ is the height of the ground point and $\tilde{\boldsymbol{v}} = \begin{bmatrix}\cos\theta\cos\varphi/\sin\varphi & \sin\theta\cos\varphi/\sin\varphi & 1\end{bmatrix}^{\mathrm{T}}$ represents the horizontal structural information of the point relative to the LiDAR. The normal vector of a ground point can be calculated by the following equation:

$$\tilde{\boldsymbol{n}} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{b}} \tag{7}$$

where $\tilde{\boldsymbol{M}} = \sum_{i=1}^{k} \tilde{\boldsymbol{v}}_i\tilde{\boldsymbol{v}}_i^{\mathrm{T}}$, $\tilde{\boldsymbol{b}} = \sum_{i=1}^{k} \tilde{\boldsymbol{v}}_i/h_i$. Similar to the range map [25], we construct a height map for ground points, and with a precomputed height lookup table, the normal vector for each ground point can be calculated. For nonground points, we use the ring-FALS to compute normal vectors.

Next, we need to calculate the weight $w_{\boldsymbol{n}}$ of the normal vectors, defined as the confidence of the normal vectors. The aforementioned method assumes consistent heights for points

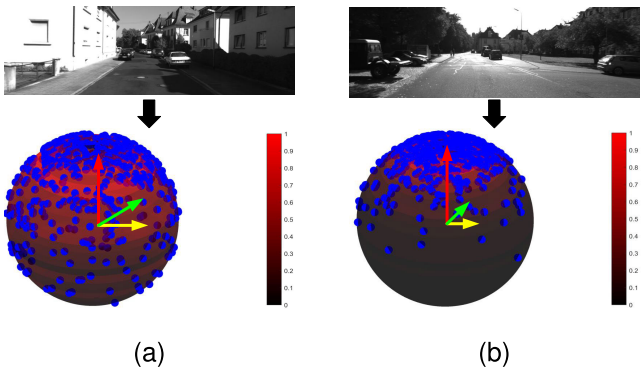


Fig. 3. Different distribution of point cloud normal vectors. (a) Urban scene. (b) Open scene. The blue points represent the unit normal vectors of each laser point. The color of the unit sphere represents the dispersion of the normal vectors: the lighter the color, the more concentrated the distribution, while the darker the color, the more dispersed the distribution. The directions of the three arrows correspond to the eigenvectors, while their lengths represent the magnitudes of the eigenvalues.

in the ground point cloud within a local range, and consistent depths for points in the nonground point cloud. Therefore, by computing the variance σ of the point cloud within a local region in the range map and height map, respectively, we define $w_{\boldsymbol{n}}$ as

$$w_{\boldsymbol{n}} = \exp(-\sigma). \tag{8}$$

If the variance σ is small, it indicates less variation in height or depth within the local region, leading to a higher weight. Conversely, if the variance σ is larger, the weight decreases.

B. Degeneration-Aware Module

The normal vector of point cloud is closely related to the cost function of pose optimization. Additionally, in degraded environments, the distribution of normal vectors in the point cloud is unbalanced [18]. Therefore, analyzing the distribution of point cloud normal vectors helps evaluate the degradation of the environment. We selected two typical scenes from the KITTI dataset [27] to compute the normal vectors of the point clouds. The estimated normal vectors were then normalized and projected onto a unit sphere, with their distribution illustrated in Fig. 3. In this figure, the color of the unit sphere indicates the degree of normal vector concentration, while the blue points represent the unit normal vector of each laser point. As shown in Fig. 3(a), in this scene, buildings and cars are distributed on both sides of the street, resulting in a uniform distribution of normal vectors. However, in open scenes, due to the dominance of ground points in the entire point cloud, there are more normal vectors along the Z-axis [as shown in Fig. 3(b)].

To assess the distribution of point cloud normal vectors, we employ weighted PCA to determine the primary directions and dispersion of normal vectors in various directions [28]. This method allows us to more accurately capture the actual distribution of normal vectors by incorporating weighting factors. First, the WCM of the normal vectors are calculated as

$$\boldsymbol{C} = \sum_{i=1}^{N} w_{\boldsymbol{n}_i} \boldsymbol{n}_i \boldsymbol{n}_i^{\mathrm{T}}. \tag{9}$$

By performing eigen decomposition, we obtain the eigenvalues λ_k and eigenvectors $\boldsymbol{u}_k$ of matrix $\boldsymbol{C}$, $k = 1, 2, 3$, arranged in descending order as $\lambda_1 > \lambda_2 > \lambda_3$. As shown in Fig. 3, the arrows in three directions represent the eigenvector directions, with their lengths indicating the magnitudes of the eigenvalues. By calculating the ratios between eigenvalues, we quantitatively measure the degree of degradation in the scene

$$D = \frac{\lambda_1}{\lambda_3}. \tag{10}$$

Physically, D represents the ratio of the sum of projections of the normal vector onto eigenvectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_3$. Therefore, if D is larger, there are more constraints in the $\boldsymbol{u}_1$ direction, and the red arrow in Fig. 3 becomes longer. This situation often occurs on open roads such as highways. To facilitate the evaluation of the degree of degradation, D is normalized by projecting it onto [0, 1]

$$\mu = -\frac{1}{D} + 1. \tag{11}$$

Furthermore, dynamic weighting of the point cloud is performed based on the distribution of the normal vector $\boldsymbol{n}$. First, we project the normal vector $\boldsymbol{n}$ onto $\boldsymbol{u}_1$

$$s = \left\| \boldsymbol{n}^{\mathrm{T}} \boldsymbol{u}_1 \right\|. \tag{12}$$

As s increases, it indicates a smaller angle between $\boldsymbol{n}$ and $\boldsymbol{u}_1$. Due to the large number of point clouds distributed near $\boldsymbol{u}_1$, the weight of this part of the points will be reduced. Furthermore, considering the degradation factor μ, if the degradation level is more severe, the weight should be smaller. Therefore, we use the following equation to weight the point $\boldsymbol{p}_i$:

$$w_{\boldsymbol{p}_i} = \ln(2 - s_i * \mu). \tag{13}$$

Finally, we adaptively adjust the optimization parameters based on the degradation level, including voxelization parameter γ, search range σ, and iteration count τ. Note that this article only discusses adaptive parameter tuning methods under locatable conditions. Extreme environments, such as tunnels, where constraints along the tunnel direction degrade, are not within the scope of this study. The purpose of point cloud voxelization is to approximate the points within a voxel range as a single point. A larger voxelization parameter γ results in more filtered point cloud data, effectively improving the computational speed of the method. However, when the point cloud degradation is severe, more point cloud is needed to provide constraints, requiring a decrease in the value of γ. The search range σ represents an outlier rejection parameter. If the distance is greater than this threshold, the point is considered an outlier, indicating that its corresponding plane cannot be found. The choice of σ depends on the initial pose error and point cloud noise. However, in degenerate environments, where the initial pose estimation error is significant, it is necessary to increase the value of σ and acquire more correspondences. τ represents the number of iterations for optimization. A smaller τ significantly reduces the computation time of the method. However, if the degeneracy is detected, it is necessary to increase τ to ensure convergence to the optimal solution.

Therefore, we dynamically update these parameters using the following equation:

$$\gamma = \mu(\gamma_{\min} - \gamma_{\max}) + \gamma_{\max} \tag{14}$$

$$\sigma = \mu(\sigma_{\min} - \sigma_{\max}) + \sigma_{\max} \tag{15}$$

$$\tau = \text{int}(\mu(\tau_{\min} - \tau_{\max}) + \tau_{\max}) \tag{16}$$

where $\gamma_{\max}$ and $\gamma_{\min}$ represent the maximum and minimum values for the voxelization parameter, while $\sigma_{\max}$ and $\sigma_{\min}$ represent the maximum and minimum values for the search range, and $\tau_{\max}$ and $\tau_{\min}$ represent the maximum and minimum values for the number of iterations. int$(\cdot)$ is used to convert a floating number to an integer.

C. LO and Mapping

For LO, we estimate the correct pose difference between two point clouds using a scan-to-map registration. Compared with scan-to-scan, it can significantly reduce drift [29]. For a point $\boldsymbol{p}_{\mathcal{S}}$ in the current scan, initially, it is transformed from the LiDAR coordinate system (L) to the world coordinate system (W) using the initial pose estimate $\boldsymbol{T}_L^W$. Subsequently, its neighboring planes are searched in the surrounding map. Finally, an iterative error minimization technique is employed to estimate the relative pose based on point-to-plane constraints

$$f_{\mathcal{S}} = \boldsymbol{n}_{\mathcal{S}}^{W\mathrm{T}}\left(\boldsymbol{T}_L^W \boldsymbol{p}_{\mathcal{S}}^L - \boldsymbol{p}_{\mathcal{S}}^W\right) \tag{17}$$

where $\boldsymbol{n}_{\mathcal{S}}^W$ and $\boldsymbol{p}_{\mathcal{S}}^W$ represent the normal vector and one point of the neighboring plane in the world coordinate system, respectively.

Additionally, we introduce the weight $w_{\boldsymbol{p}_i}$ of the point cloud to balance the inconsistency in optimization. Therefore, the pose estimation is modeled as a weighted least squares problem

$$\boldsymbol{T}_L^{W*} = \min_{\boldsymbol{T}_L^W} \sum_{i=1}^{N} w_{\boldsymbol{p}_i} f_{\mathcal{S}_i}. \tag{18}$$

Using the Levenberg–Marquardt algorithm [10], the optimal pose estimation can be iteratively solved. Specifically, the Jacobian matrix of (18) can be calculated by

$$\boldsymbol{J}_{\mathcal{S}} = w_{\boldsymbol{p}_i} \frac{\partial f_{\mathcal{S}}}{\partial\left(\boldsymbol{T}_L^W \boldsymbol{p}_{\mathcal{S}}^L\right)} \frac{\partial\left(\boldsymbol{T}_L^W \boldsymbol{p}_{\mathcal{S}}^L\right)}{\partial(\delta\xi)} = w_{\boldsymbol{p}_i} \boldsymbol{n}_{\mathcal{S}}^{W\mathrm{T}} \boldsymbol{J}_{\xi} \tag{19}$$

where $\xi = [\boldsymbol{t}, \vartheta]^{\mathrm{T}} \in \mathfrak{se}(3)$ is the corresponding Lie algebra with respect to the translation and the rotation [15]. Hence, $\boldsymbol{J}_{\xi} = \left[\boldsymbol{I}_{3\times 3} \;\; -[\boldsymbol{T}_L^W \boldsymbol{p}_{\mathcal{S}}^L]_{\times}\right]$ can be estimated by applying left perturbation model [21], and $[\cdot]_{\times}$ represents the antisymmetric matrix of the vector. Therefore, the relative transformation matrix $\Delta\boldsymbol{T}_L^W$ is calculated as

$$\Delta\boldsymbol{T}_L^W = \left(\sum_{i=1}^{N} \boldsymbol{J}_{\mathcal{S}_i}{}^{\mathrm{T}} w_{\boldsymbol{p}_i} \boldsymbol{J}_{\mathcal{S}_i}\right)^{-1} \left(\sum_{i=1}^{N} -\boldsymbol{J}_{\mathcal{S}_i}{}^{\mathrm{T}} w_{\boldsymbol{p}_i} f_{\mathcal{S}_i}\right). \tag{20}$$

The solution is iteratively refined until it converges to an optimal result $\Delta\boldsymbol{T}_L^W$. Consequently, the transformation matrix $\boldsymbol{T}_L^{W*}$ can be calculated as

$$\boldsymbol{T}_L^{W*} = \Delta\boldsymbol{T}_L^W \cdot \tilde{\boldsymbol{T}}_L^W \tag{21}$$

TABLE I
RUNNING TIME OF NORMAL ESTIMATION (UNIT: ms)

	Segmentaion	Projection	Computation	Smoothing	Total
PCL(single thread)	–	–	–	–	92.14
PCL(OMP 10 threads)	–	–	–	–	37.64
Ring-FALS	–	6.96	4.44	5.67	17.07
Ground-FALS	12.53	2.12	2.17	5.08	22.90

where $\tilde{\boldsymbol{T}}_L^W$ is the initial pose provided by the constant velocity model.

V. EXPERIMENTS

A. Experiment Setup

Section IV presented the theoretical part of our study. Next, we designed a large number experiments to evaluate the performance of the proposed method. We first evaluated the performance of normal vector estimation on the M2DGR dataset [30], which was collected using a ground platform equipped with a Velodyne-32 LiDAR. Furthermore, We validated the feasibility of the proposed degeneration detection method on the KITTI benchmark [27]. Finally, we conducted evaluations of the proposed LO on real-world datasets. We assessed the accuracy of the odometry using the KITTI benchmark. Moreover, we evaluated the mapping performance using the private dataset.

The operating device for all experiments was a notebook computer with i5-1240P processor and 16 GB RAM. The running platform was the Robot Operating System (ROS) noetic. Our open source implementation is available at https://github.com/BuaaYfl/NA-LOAM.

B. Evaluation of Normal Estimation

To demonstrate the effectiveness of our method more clearly and intuitively, this article chose to visualize the extraction results using the sequence gate 03 from the M2DRG dataset, because we lack a straightforward ground truth to calculate the normals of point clouds. Moreover, we conducted a comparative analysis between our method (Ground-FALS) and ring-FALS [25]. For better comparison, we only showed the extraction results of ground point normal vector (see Fig. 4). Note that, both methods used Patchwork [26] for ground segmentation. It is readily observable from Fig. 4(d) and (e) that the proposed method performs better than ring-FALS, as it exhibits more uniformly distributed normal vectors. Additionally, we also tested the average processing times of normal estimation for a single LiDAR scan. For better comparison, we also add the traditional PCL [31] normal vector estimation tool, which uses KDtree for nearest neighbor search and least squares method to estimate the normal vector. PCL can also use the OpenMP [32] tool to speed up operations. The results are shown in Table I. This table shows that compared to PCL, both ring-FALS and ground-FALS require less processing time, regardless of whether OpenMP is used. This is mainly because PCL needs to use KDtree to search for nearest neighbor points. Additionally, for each laser point, eigenvalue decomposition is required to estimate the normal vectors (Velodyne-32 has 57 600 laser points per frame), which

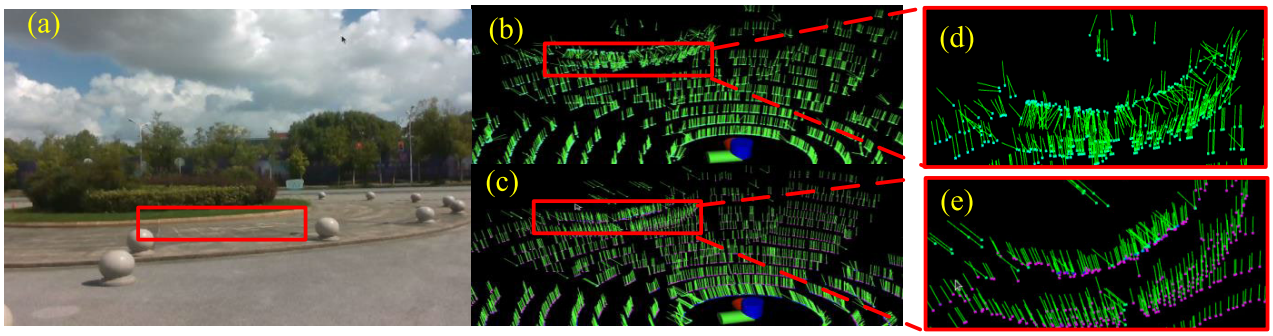


Fig. 4. Normal vector estimation results from the sequence gate 03 of M2DRG dataset. (a) Real scene. (b) and (d) Results based on ring-FALS [25], and (c) and (e) are the results based on our method. The green line represents normalized normals and the red point represents the point cloud.

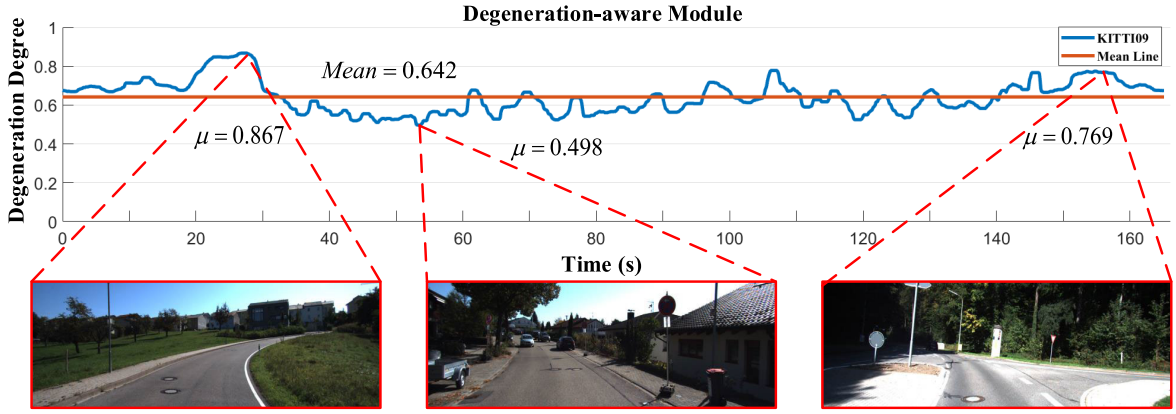


Fig. 5. Degeneration detection results from the sequence 09 of KITTI dataset. The images in the red box correspond to the scene at the current moment.

significantly increases the algorithm's processing time. And the proposed method has similar time cost comparing with ring-FALS. The main difference is evident in the segmentation stage. Although point cloud segmentation may increase the runtime, it also filters out noise to reduce the subsequent processing time.

C. Degeneration Detection

Then, we tested the performance of degeneration detection on the KITTI benchmark. To better showcase the effectiveness of the degeneration detection method, we chose the sequence 09 from the KITTI benchmark for testing. This sequence includes both country and urban road segments. The country segments contains more open, with only a few trees on either side of the road. In contrast, the urban segments are rich in features. The degeneration detection results are shown in Fig. 5. Fig. 5 shows that when the degradation factor μ increases, the effective features in the scene decrease, leading to scene degradation. This further validates the proposed method can accurately identify degraded road segments. To enhance the reliability of the degradation detection module, we integrate this algorithm into the adaptive preprocessing module (as shown in Fig. 1) to assess the accuracy of the odometry. The detailed results are presented in the following section.

D. Evaluation of LO

Based on the degree of degeneration, we integrates the adaptive weighting module and dynamic parameter update module into the F-LOAM [21] framework, proposing a novel odometry method. Specifically, in the odometry module, we initialize the pose estimation using a constant velocity model and iteratively optimize the pose using point-to-plane constraints. In this section, we conducted some experimental tests on the LO system, to provide a comprehensive and systematic evaluation of the proposed method. First, we selected the well-known KITTI benchmark to perform precision evaluation on the LO system. By comparing with the ground truth, we calculated the relative pose estimation error [27] and analyzed its performance in various scenarios. Second, we assessed LiDAR mapping using a private dataset, validating the feasibility of the method by comparing the generated maps with the actual environment.

1) Experiments on Public Dataset: The KITTI benchmark, was captured with Velodyne HDL-64E LiDAR sensor, where 10 sequences provide GNSS/IMU ground truth (the sequence 03 has been officially removed). We compared our method (NA-LOAM) with state-of-the-art LO methods including A-LOAM [9],[1] F-LOAM [21][2] (our baseline), Lego-LOAM [10],[3], and KISS-ICP [33].[4] Note that, for a fair comparison, the loop closure for LeGO-LOAM is disabled. All the results were obtained by experimenting with its open-source code.

[1]https://github.com/HKUST-Aerial-Robotics/A-LOAM
[2]https://github.com/wh200720041/floam
[3]https://github.com/RobustFieldAutonomyLab/LeGO-LOAM
[4]https://github.com/PRBonn/kiss-icp

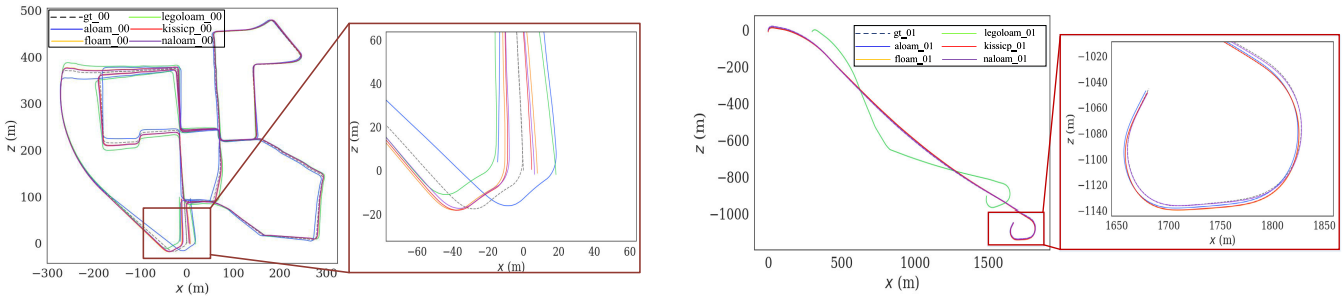


Fig. 6. Trajectory of our algorithm and the comparative algorithm on the KITTI 00 and 01 datasets.

TABLE II
LO EVALUATION COMPARISON ON KITTI BENCHMARK

Sequence	Environment	A-LOAM	F-LOAM	LeGO-LOAM	KISS-ICP	NA-LOAM(FW)	NA-LOAM(FP)	NA-LOAM(FV)
00	Urban	0.81/2.14	**0.60**/1.43	0.75/1.81	0.63/1.54	0.67/1.47	0.69/1.50	0.62/**1.41**
01	Highway	0.49/1.97	0.57/2.17	2.55/21.99	0.55/2.14	0.51/1.98	0.50/1.85	**0.47/1.80**
02	Urban+Country	0.82/2.23	0.62/1.65	1.00/2.65	0.54/**1.36**	0.55/1.56	0.59/1.60	**0.52**/1.46
04	Country	0.59/0.71	0.59/0.67	0.70/1.11	0.57/0.95	0.57/0.70	0.58/0.71	**0.54/0.66**
05	Urban	**0.43**/1.00	**0.43**/0.94	0.82/1.87	0.51/1.17	0.49/0.97	0.50/0.99	0.46/**0.92**
06	Urban	0.46/1.25	**0.23/0.44**	0.75/2.29	0.33/0.65	0.24/0.51	0.27/0.54	**0.23**/0.49
07	Urban	0.57/1.24	0.46/1.23	0.78/1.53	**0.36/0.53**	0.71/1.30	0.74/1.33	0.66/1.25
08	Urban+Country	0.50/1.49	0.52/1.39	0.88/2.31	0.56/1.52	0.52/1.37	0.55/1.41	**0.49/1.36**
09	Urban+Country	0.52/1.27	0.51/1.25	0.99/2.44	0.50/1.28	0.48/1.23	0.46/1.24	**0.46/1.20**
10	Urban+Country	0.58/1.02	0.60/1.52	1.01/2.51	0.72/1.81	0.59/1.07	0.64/1.27	**0.57/1.01**
Average		0.58/1.43	0.51/1.27	1.02/4.51	0.53/1.29	0.53/1.21	0.55/1.24	**0.50/1.16**

Mean relative pose error over trajectories of 100 to 800 m: relative rotational error in degrees per 100 m / relative translational %. Bold numbers indicate the best performance in terms of translational error and rotation error. Loop detection is not used in all methods.

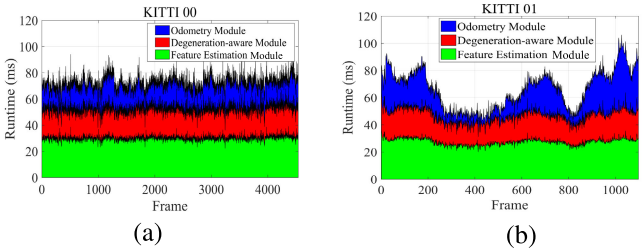


Fig. 7. Runtime per frame of NA-LOAM. (a) Runtimes of NA-LOAM on KITTI 00 and (b) runtimes of NA-LOAM on KITTI 01.

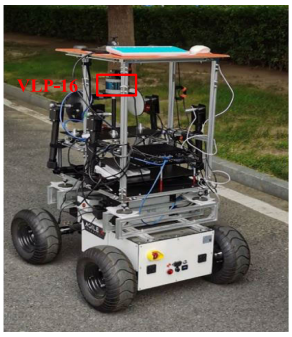


Fig. 8. Ground vehicle equipped is used for data acquisition.

First, we visualized the trajectories of two typical scenes (KITTI 00 and 01), as shown in the Fig. 6. It can be observed from Fig. 6 that our proposed algorithm demonstrates competitiveness compared to the state-of-the-art algorithm, exhibiting better localization accuracy and performance in these scenarios. Then, we analyzed the performance of the aforementioned algorithms on the entire KITTI dataset, quantitatively assessing localization accuracy using the official KITTI odometry evaluation tool. Additionally, we explored the effects of the adaptive weighted constraint module and dynamic parameter update module on the accuracy of the LO. Therefore, we define the full version as NA-LOAM (Full Version, FV), the version with fixed weight (e.g., $w_{p_i} = 1$) as NA-LOAM (Fix Weight, FW), and the version with fixed parameters as NA-LOAM (Fix Parameter, FP). All the results are presented in Table II.

Table II shows that our method outperforms these state-of-the-art LiDAR SLAM methods in most cases. Specifically, compared with the baseline method F-LOAM, our method significantly improves the translation accuracy. The main reason is that the pose constraint is not only related to the point cloud normals but also to the point cloud positions, with the contribution of translational degeneration solely dependent on the normals [29]. In addition, we also notice that our method performs better than F-LOAM in sequence 01, 02,

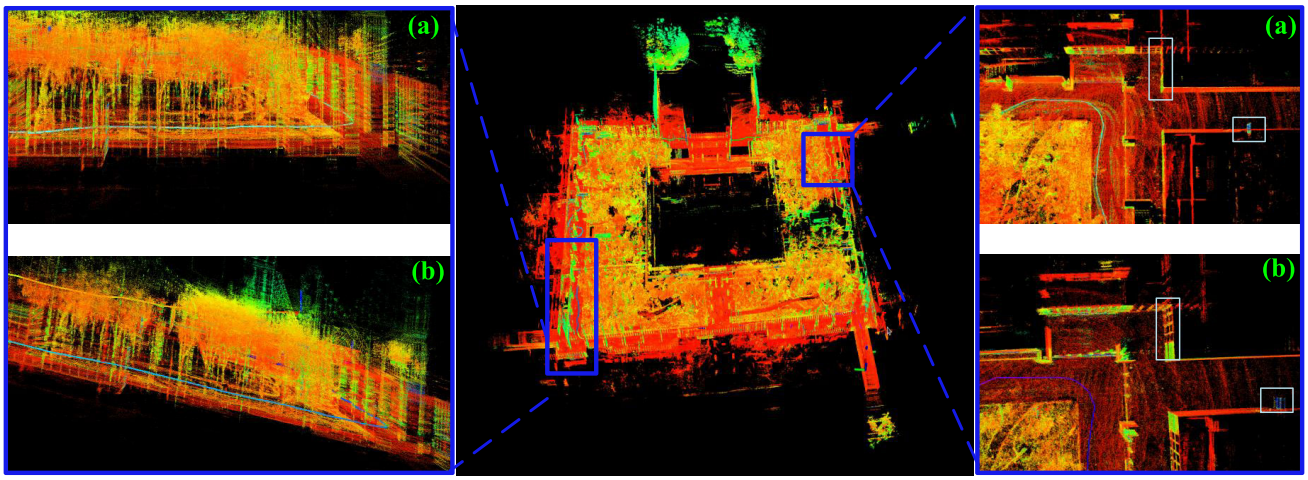


Fig. 9. Mapping results and some details of NA-LOAM and F-LOAM in real environment. (a) Represents the mapping details of NA-LOAM, which adaptively adjusts thresholds and weights based on the surrounding environment to ensure the stability of the algorithm. (b) Represents the mapping details of F-LOAM, which, using default parameter configurations, fail to filter out erroneous point cloud information, resulting in map errors and trajectory drift.

04, 08, 09, 10. This is mainly because these sequences contain degeneration scenes, and our algorithm is able to take corresponding measures by detecting the degeneration scenes. Furthermore, the results in Table II confirm the effectiveness of the adaptive weighted constraint module and dynamic parameter update module, i.e., the performance degradation from NA-LOAM (FV) to NA-LOAM (FW) or NA-LOAM (FP). However, in most cases, the dynamic parameter update module performs better in terms of accuracy compared to the adaptive weighted constraint module. The reason is that, in most scenarios, the degradation level of the environment is not high, and there is a small difference in point cloud weights. However, in sequence 01, which is on a highway road, the degradation level is higher, and the adaptive weighted constraint module has a significant impact.

2) Time Analysis: In addition, to verify the real-time performance of our algorithm, we conducted a time analysis of each module of NA-LOAM on the KITTI dataset, and the results are shown in Table III. Furthermore, we visualized the time consumption for sequences 00 and 01, as shown in Fig. 7. From Fig. 7 and Table III, it can be observed that the feature estimation module takes approximately 30 ms, mainly for extracting the normal vectors of the point cloud. Compared to the M2DGR dataset, the time consumption is higher, mainly because the KITTI dataset uses the HDL-64 LiDAR for data acquisition, resulting in a significantly larger number of point clouds per frame. Additionally, the degeneration-aware module uses point cloud normal vectors to calculate the WCM, taking approximately 20 ms. Finally, the odometry module including initial value estimation, feature matching, and iterative optimization, takes approximately 20 ms. In summary, on the KITTI dataset, the total time consumption of NA-LOAM does not exceed 100 ms, enabling real-time processing of the KITTI dataset collected at a frequency of 10 Hz.

3) Experiments on Private Dataset: Finally, we evaluated the proposed LO system on a private dataset collected using our equipment. The dataset was obtained by a ground vehicle equipped with a Velodyne VLP-16 LiDAR sensor navigating

TABLE III
RUNNING TIME PER FRAME OF NA-LOAM ON THE KITTI DATASETS (UNIT: ms)

Sequence	Feature Estimation	Degeneration-aware	Odometry	Total
00	30.04	22.29	19.87	72.21
01	27.57	19.17	19.33	66.07
02	31.47	22.56	20.47	74.50
04	28.11	20.14	19.14	67.39
05	30.17	20.32	20.03	70.52
06	28.22	18.97	19.62	66.81
07	29.36	20.34	19.51	69.21
08	29.42	19.87	20.01	69.30
09	27.14	21.44	20.30	68.88
10	28.72	20.28	18.87	67.87

in a real garden environment, as shown in Fig. 8. The vehicle traversed a route forming a loop in a garden real environment, including various elements such as buildings, long corridors, shrubs, and trees. Using our method, a corresponding environmental map was constructed from the LiDAR point cloud, as illustrated in Fig. 9(a). It can be observed that the environmental map constructed by our method is relatively accurate and consistent with the real-world scene. Notably, there is no occurrence of overlap in the entire environmental map. It can be observed that the environmental map constructed by our method is relatively accurate and consistent with the real-world scene. In addition, we visualized the mapping results of the baseline algorithm F-LOAM, as shown in Fig. 9(b). F-LOAM exhibits mapping errors at corners, where the road information is complex. Using default parameters fails to effectively filter out erroneous point cloud data, leading to map overlapping and trajectory drift. Due to the prevalence of unreliable point cloud information from environmental factors such as vegetation, F-LOAM fails to adaptively weight reliable

TABLE IV
TRAJECTORY ERRORS ON THE PRIVATE DATASET (UNIT: m)

Sequence	A-LOAM	F-LOAM	Lego-LOAM	KISS-ICP	NA-LOAM
Garden	7.11	1.19	13.22	2.01	**0.92**

information, resulting in significant upward drift throughout the entire map. Finally, we compared the trajectory errors of multiple algorithms on this dataset. Due to the lack of GPS ground truth, but with the trajectory's endpoint and starting point at the same location, we introduced end-to-end trajectory error to quantitatively evaluate algorithm accuracy. As shown in Table IV, our algorithm exhibits the highest localization accuracy, with a positioning error reduction of 22.69% compared to F-LOAM.

VI. CONCLUSION

In this article, we present a novel LO method, NA-LOAM, which evaluates scene degradation based on the normal vectors of the current point cloud, and then dynamically updates the parameters and adaptively weights the point cloud. These preprocessing modules ensure a more uniform and precise point cloud data for the subsequent LO and mapping modules. The performance of the proposed method is evaluated on various datasets, including KITTI benchmark and private datasets, showcasing its effectiveness in different scenarios, especially regarding adaptability to different environments. The proposed method excels in handling challenging scenarios, including those with degradation, and outperforms the state-of-the-art LiDAR SLAM methods in most cases. There are some limitations for our method. First, our method only considers navigation using a single sensor. In addition, the method may encounter challenges in highly dynamic or cluttered environments, requiring further robustness validation. Future research could focus on refining the algorithm's adaptability to complex scenarios and extending its capabilities for seamless integration with other sensor modalities. Additionally, our algorithm was only tested for localization performance in locatable environments. Further evaluation is needed to assess its performance in completely degraded environments, such as tunnels.

REFERENCES

[1] D. Chang, S. Huang, R. Zhang, M. Hu, R. Ding, and X. Qin, "WiCRF2: Multi-weighted LiDAR odometry and mapping with motion observability features," *IEEE Sensors J.*, vol. 23, no. 17, pp. 20236–20246, Sep. 2023.

[2] S. Liang, Z. Cao, J. Yu, M. Tan, and S. Wang, "A tight filtering and smoothing fusion method with feature tracking for LiDAR odometry," *IEEE Sensors J.*, vol. 22, no. 13, pp. 13622–13631, Jul. 2022.

[3] Y. Wang, Y. Lou, W. Song, and Z. Tu, "A tightly-coupled framework for large-scale map construction with multiple non-repetitive scanning LiDARs," *IEEE Sensors J.*, vol. 22, no. 4, pp. 3626–3636, Feb. 2022.

[4] B. Zhou, D. Xie, S. Chen, H. Mo, C. Li, and Q. Li, "Comparative analysis of SLAM algorithms for mechanical LiDAR and solid-state LiDAR," *IEEE Sensors J.*, vol. 23, no. 5, pp. 5325–5338, Mar. 2023.

[5] F. Pomerleau, "Applied registration for robotics: Methodology and tools for ICP-like algorithms," Ph.D. dissertation, ETH Zurich, 2013.

[6] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2008, pp. 19–25.

[7] S. Chen et al., "NDT-LOAM: A real-time LiDAR odometry and mapping with weighted NDT and LFA," *IEEE Sensors J.*, vol. 22, no. 4, pp. 3660–3671, Feb. 2022.

[8] T. Li et al., "P^3-LOAM: PPP/LiDAR loosely coupled SLAM with accurate covariance estimation and robust RAIM in urban canyon environment," *IEEE Sensors J.*, vol. 21, no. 5, pp. 6660–6671, Mar. 2021.

[9] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. X*, Jul. 2014, pp. 1–9.

[10] 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.

[11] R. Chen and L. Zhao, "Multi-level autonomous integrity monitoring method for multi-source PNT resilient fusion navigation," *Satell. Navigat.*, vol. 4, no. 1, p. 21, Dec. 2023.

[12] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 809–816, doi: 10.1109/ICRA.2016.7487211.

[13] R. Chen and L. Zhao, "Two-level integrity-monitoring method for multi-source information fusion navigation," *Remote Sens.*, vol. 16, no. 1, p. 120, Dec. 2023.

[14] J.-E. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2480–2485.

[15] H. Li, B. Tian, H. Shen, and J. Lu, "An intensity-augmented LiDAR-inertial SLAM for solid-state LiDARs in degenerated environments," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[16] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Proc. 4th Int. Conf. 3-D Digit. Imag. Model*, 2003, pp. 260–267.

[17] M. Ji, W. Shi, Y. Cui, C. Liu, and Q. Chen, "Adaptive denoising-enhanced LiDAR odometry for degeneration resilience in diverse terrains," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–15, 2024.

[18] H. Lim, D. Kim, B. Kim, and H. Myung, "Adalio: Robust adaptive LiDAR-inertial odometry in degenerate indoor environments," in *Proc. 20th Int. Conf. Ubiquitous Robots*, 2023, pp. 1–24.

[19] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3084–3091.

[20] C. Hong, S.-H. Kim, J.-H. Kim, and S. M. Park, "A linear-mode LiDAR sensor using a multi-channel CMOS transimpedance amplifier array," *IEEE Sensors J.*, vol. 18, no. 17, pp. 7032–7040, Sep. 2018.

[21] 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, doi: 10.1109/IROS51168.2021.9636655.

[22] J. Tang, X. Zhang, Y. Zou, Y. Li, and G. Du, "A high-precision LiDAR-inertial odometry via Kalman filter and factor graph optimization," *IEEE Sensors J.*, vol. 23, no. 11, pp. 1–10, Jul. 2023.

[23] J. Li, X. Zhang, Y. Zhang, Y. Chang, and K. Zhao, "RF-LOAM: Robust and fast LiDAR odometry and mapping in urban dynamic environment," *IEEE Sensors J.*, vol. 23, no. 23, pp. 29186–29199, Dec. 2023.

[24] W. Shi, S. Li, C. Yao, Q. Yan, C. Liu, and Q. Chen, "Dense normal based degeneration-aware 2-D lidar odometry for correlative scan matching," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–16, 2023.

[25] K. Huang, J. Zhao, Z. Zhu, C. Ye, and T. Feng, "LOG-LIO: A LiDAR-inertial odometry with efficient local geometric information estimation," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 459–466, Jan. 2024.

[26] H. Lim, M. Oh, and H. Myung, "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3D LiDAR sensor," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6458–6465, Oct. 2021.

[27] 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.*, Jun. 2012, pp. 3354–3361.

[28] L. Delchambre, "Weighted principal component analysis: A weighted covariance eigendecomposition approach," *Monthly Notices Roy. Astronomical Soc.*, vol. 446, no. 4, pp. 3545–3555, Feb. 2015.

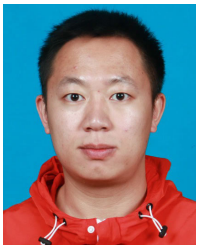
[29] T. Tuna, J. Nubert, Y. Nava, S. Khattak, and M. Hutter, "X-ICP: Localizability-aware LiDAR registration for robust localization in extreme environments," *IEEE Trans. Robot.*, vol. 40, no. 1, pp. 452–471, Jul. 2024.

[30] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, "M2DGR: A multi-sensor and multi-scenario SLAM dataset for ground robots," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2266–2273, May 2022.

[31] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.

[32] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel Programming in OpenMP*. San Mateo, CA, USA: Morgan Kaufmann, 2001.

[33] 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.



Fengli Yang received the B.S. degree in information and computing science and the M.S. degree in operations research and cybernetics from Yunnan University, Kunming, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

His research interests include light detection and ranging (LiDAR) simultaneous localization and mapping (SLAM), and multisensor fusion.

Wangfang Li received the B.S. degree from the Department of Automation, Shandong University, Jinan, Shandong, China, in 2022. He is currently pursuing the master's degree in control science and engineering with Beihang University, Beijing, China.

His research interests include computer vision, SLAM, and LiDAR odometry.

Long Zhao received the B.S. degree in mathematics education from Inner Mongolia Normal University, Inner Mongolia, China, in 1998, the M.S. degree in control theory and control engineering Harbin Engineering University, Harbin, China, in 2001, and the Ph.D. degree in precision instrument and machinery from Beijing University of Aeronautics and Astronautics, Beijing, China, in 2004.

He is currently a Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, and the Head of the Digital Navigation Center. His research interests include the geophysical navigation, vision navigation, and multisensor adaptive integrated navigation.