

LOG-LIO2: A LiDAR-Inertial Odometry with Efficient Uncertainty Analysis

Kai Huang¹, Junqiao Zhao^{*,2,3}, Jiaye Lin^{2,3}, Zhongyang Zhu^{2,3}, Shuangfu Song¹, Chen Ye², Tiantian Feng¹

Abstract—Uncertainty in LiDAR measurements, stemming from factors such as range sensing, is crucial for LIO (LiDAR-Inertial Odometry) systems as it affects the accurate weighting in the loss function. While recent LIO systems address uncertainty related to range sensing, the impact of incident angle on uncertainty is often overlooked by the community. Moreover, the existing uncertainty propagation methods suffer from computational inefficiency. This paper proposes a comprehensive point uncertainty model that accounts for both the uncertainties from LiDAR measurements and surface characteristics, along with an efficient local uncertainty analytical method for LiDAR-based state estimation problem. We employ a projection operator that separates the uncertainty into the ray direction and its orthogonal plane. Then, we derive incremental Jacobian matrices of eigenvalues and eigenvectors w.r.t. points, which enables a fast approximation of uncertainty propagation. This approach eliminates the requirement for redundant traversal of points, significantly reducing the time complexity of uncertainty propagation from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ when a new point is added. Simulations and experiments on public datasets are conducted to validate the accuracy and efficiency of our formulations. The proposed methods have been integrated into a LIO system, which is available at <https://github.com/tiev-tongji/LOG-LIO2>.

I. INTRODUCTION

Uncertainty stemming from sensor limitations, measurement noises, and unpredictable physical environments plays a crucial role in robotic state estimation, as it affects the accurate weighting of distance metrics in the loss function [1]. One common approach is to model the measurement uncertainty using zero-mean Gaussian noise with respect to point coordinates and integrate these noises into the state estimation. However, such uncertainty modeling of LiDAR measurements lacks sufficient accuracy.

To address this issue, [2] develops a point uncertainty model that incorporates both range and bearing, while [3] further includes surface roughness. These uncertainties are then propagated to the geometric elements such as planes, which are utilized for weighting the point-to-element distance in scan-to-map registration, thereby enhancing the accuracy and reliability of the system [2], [4], [5].

¹Kai Huang, Shuangfu Song and Tiantian Feng are with the School of Surveying and Geo-Informatics, Tongji University, Shanghai, China (e-mail: huangkai@tongji.edu.cn; songshuangfu@tongji.edu.cn; fengtiantian@tongji.edu.cn).

²Junqiao Zhao, Jiaye Lin, Zhongyang Zhu and Chen Ye are with Department of Computer Science and Technology, School of Electronics and Information Engineering, Tongji University, Shanghai, China, and the MOE Key Lab of Embedded System and Service Computing, Tongji University, Shanghai, China (e-mail: zhaojunqiao@tongji.edu.cn; 2233057@tongji.edu.cn; 2310920@tongji.edu.cn; yechen@tongji.edu.cn).

³Institute of Intelligent Vehicles, Tongji University, Shanghai, China. Digital Object Identifier (DOI): see top of this page.

However, range uncertainty is also influenced by the incident angle [6], [7], a factor often overlooked by existing studies. Additionally, continuous propagation of uncertainty from numerous points, which exhibits linear time complexity $\mathcal{O}(n)$, can be computationally expensive, particularly for dense point cloud and real-time applications. Therefore, an efficient uncertainty propagation method is highly desirable.

In this paper, we present a comprehensive point uncertainty model that incorporates all relevant factors affecting LiDAR measurements, including range, bearing, incident angle, and surface roughness. This model leverages the projection operator [7] to separate the uncertainty into the ray direction and its orthogonal plane.

To accelerate the propagation of uncertainty from points to the geometric elements, we derive the incremental Jacobian matrices for eigenvalues and eigenvectors corresponding to specified points from Welford’s formulation [8]. The parametric uncertainty of the geometric elements is then updated incrementally by fast approximations.

We validate the efficiency and accuracy of the fast uncertainty approximation through simulation experiments. Subsequently, we integrate these methods into a LIO system named LOG-LIO2. By comparing the performance of LOG-LIO2 with state-of-the-art LIO systems, i.e. VoxelMap [5], LOG-LIO [9], and FastLIO2 [10], we demonstrate the performance improvement achieved by the proposed techniques.

The main contributions of this work are as follows:

- A comprehensive point uncertainty model with a fast calculation method, incorporating the uncertainty of range and bearing measurements from LiDAR, as well as incident angle and roughness concerning the target surface.
- A fast approximation of uncertainty propagation from points to eigenvalues and eigenvectors by leveraging the incremental Jacobian matrices, reducing the time complexity from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ by eliminating the need for repetitive calculations when a new point is added.
- We demonstrate the efficiency and accuracy of our methods from both simulation and public datasets. The methods proposed in this paper are incorporated into a LIO system which is available at <https://github.com/tiev-tongji/LOG-LIO2>.

II. RELATED WORKS

The integration of IMU has significantly improved the practical robustness of LIO systems [10]. However, a critical limitation persists in these systems: the uncertainty of each point is treated homogeneously, neglecting potential spatial and environmental variations that can significantly impact accuracy.

[2] investigates the physical measuring principles of LiDAR and derives the uncertainty model for each laser beam encompassing range and bearing uncertainties. Specifically, the bearing uncertainty is modeled as a \mathbb{S}^2 perturbation [11] in the tangent plane of the ray. However, the calculation of \mathbb{S}^2 perturbation requires the construction of two bases in the tangent plane, leading to increased computational cost.

[4] introduces a novel adaptive voxelization method that iteratively divides voxels until the point distribution matches the line or plane geometry. The paper derives the Jacobian and Hessian matrices of eigenvalues and eigenvectors w.r.t. the points within the voxel, leveraging these matrices for LiDAR bundle adjustment.

Building on these prior works, VoxelMap [5] integrates the point uncertainty model and adaptive voxelization technique into a LiDAR-based odometry system. The uncertainty associated with each point is then propagated to the plane parameters to apply weights to the point-to-plane distance in the state estimation. However, the traversing of all points during propagation reduces the real-time performance of the system.

In addition to inherent laser range and bearing uncertainties, the geometrical properties of the target surface also contribute significantly to observed point uncertainty. [6] and [7] analyze the geometrical relationship between the laser beam and the surface normal, incorporating the uncertainty arising from the incident angle into the ray direction. While they obtain a closed-form approximation of this uncertainty, it is overlooked by the SLAM community.

Distinct from prior works, [3] introduces a principled uncertainty model for LiDAR scan-to-map registration. This model incorporates the roughness of the measured uneven planes, thereby enhancing robustness but also introducing complexity in estimation.

Having the above, our focus is on developing a comprehensive point uncertainty model and deriving efficient uncertainty propagation methods, aiming to strike a balance between accuracy and computational efficiency.

III. PRELIMINARY

In this paper, we assume the transformation between LiDAR and IMU is calibrated in advance. We denote the statistics of point cloud $\mathcal{P} = \{\mathbf{p}_i, i = 1, \dots, k\}$ by $(\cdot)_k$, where the subscript k indicates the point cloud with k points. For simplicity, the parentheses "()" may be omitted when there is no ambiguity. The sum of the outer products of all points deviating from the center \mathbf{m} is denoted by \mathbf{S} , which is normalized to obtain the covariance matrix, \mathbf{A} .

$$\mathbf{m}_k = \frac{1}{k} \sum_{i=1}^k \mathbf{p}_i; \mathbf{A}_k = \frac{1}{k} \mathbf{S}_k = \frac{1}{k} \sum_{i=1}^k (\mathbf{p}_i - \mathbf{m}_k)(\mathbf{p}_i - \mathbf{m}_k)^T. \quad (1)$$

For the symmetric matrix \mathbf{A} , eigenvalue decomposition is performed as follows:

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3], \mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3) \quad (2)$$

where \mathbf{v}_i is the eigenvector corresponding to the eigenvalue λ_i with $\lambda_1 < \lambda_2 < \lambda_3$.

IV. POINT UNCERTAINTY MODEL

The point-wise uncertainty model quantifies the reliability of each input point, allowing SLAM systems to handle real-world complexities more effectively. Modeling the point uncertainty derived from the range and bearing measurements as [2], [4], [5] is natural since the coordinate is obtained from the relative position w.r.t. the LiDAR. However, the geometric properties of the target surface, i.e. incident angle and roughness, also contribute to the uncertainty of the LiDAR measurements [3], [6], [7]. Our uncertainty model takes into account both the properties of the sensor itself and the observed geometry.

We assume the point-wise uncertainty follows a Gaussian distribution. Inspired by [7] and as shown in Figure 1(a), we denote the magnitude of uncertainty along the ray \mathbf{v}_{r_i} direction as $\sigma_{r_i}^2$ while the magnitude of uncertainty on the plane orthogonal to \mathbf{v}_{r_i} is denoted as $\sigma_{\phi_i}^2$. Note that the orientation of $\sigma_{\phi_i}^2$ is not specified in order to simplify the calculations, as further explained in Section IV-B.

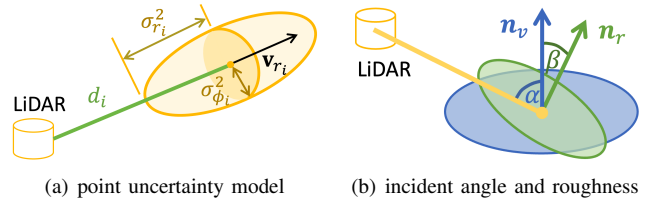


Fig. 1. Illustrations of geometric relationship for the point uncertainty model.

A. Uncertainty Factors

1) *Range and bearing*: Similar to [2], [5], but with a different calculation approach, we assume that the magnitude of the range and bearing uncertainty to each laser beam are both constant factors, σ_d^2 and σ_ω^2 , respectively. In a 3D space, the σ_d^2 extends in the \mathbf{v}_{r_i} direction, while σ_ω^2 is isotropic in the orthogonal plane of the \mathbf{v}_{r_i} .

2) *Incident Angle*: As illustrated in Figure 1(b), the incident angle α between the laser beam and the surface normal plays a pivotal role in the range sensing [6], [7]. As α increases from 0 to $\pi/2$, the range uncertainty also rises. We adopt the uncertainty model caused by the incident angle σ_{in_i} along the ray direction from [6], [7]:

$$\sigma_{in_i} = d_i \sigma_\omega \tan \alpha_i \quad (3)$$

where d_i is the distance from \mathbf{p}_i to the LiDAR.

3) *Roughness*: We account for the impact of surface roughness on the point in a simpler and faster manner compared to [3]. To achieve this, we estimate the normal of the target surface by analyzing two different neighborhood scales. As shown in Figure 1(b), the angle β formed between these two normals serves as a metric for quantifying the uncertainty introduced by the roughness:

$$\sigma_o = \eta \sin \beta, \beta = \arccos(\mathbf{n}_r, \mathbf{n}_v) \quad (4)$$

where η is a preset value, \mathbf{n}_r and \mathbf{n}_v are normals estimated under different neighborhood scales.

B. Fast Uncertainty Calculation

Assuming the roughness is isotropic in 3D space, we integrate the aforementioned three factors to compute the covariance of the point \mathbf{p}_i as follows:

$$\mathbf{A}_{\mathbf{p}_i} = \sigma_{r_i}^2 \mathbf{v}_{r_i} \mathbf{v}_{r_i}^T + \sigma_{\phi_i}^2 (\mathbf{I}_{3 \times 3} - \mathbf{v}_{r_i} \mathbf{v}_{r_i}^T) + \sigma_o^2 \mathbf{I}_{3 \times 3} \quad (5)$$

where $\sigma_{r_i}^2 = \sigma_d^2 + \sigma_{in_i}^2$, $\sigma_{\phi_i} = d_i \sigma_\omega$, and \mathbf{v}_{r_i} is the unit vector in ray direction. The first term captures the uncertainty arising from the range measurement and incident angle. We use the projection operator $(\mathbf{I} - \mathbf{v}_r \mathbf{v}_r^T)$ from [7] to isotropically project bearing uncertainty onto the 2D plane orthogonal to \mathbf{v}_{r_i} . Note that considering only the uncertainty of range and bearing, our method aligns with [2], [4], [5], which is proved in Appendix I-A. We present a more comprehensive point uncertainty model accompanied by an efficient method that offers greater geometric interpretability.

V. LUFA: LOCAL UNCERTAINTY FAST APPROXIMATION

LiDAR SLAM systems maintain statistics of localized map regions, such as mean position, eigenvalues, and eigenvectors, which are crucial for state estimation. Deriving Jacobian matrices for these statistics is vital for uncertainty propagation using linear Gaussian principles [1]. As outlined in BALM [4] and VoxelMap [5], computing these Jacobian matrices involves traversing all points in \mathcal{P} , resulting in $\mathcal{O}(n)$ time complexity, which can impact real-time performance.

To relieve this computation burden, we propose an incremental update approach for computing these statistics in $\mathcal{O}(1)$ time complexity when adding a new point. This ensures real-time performance, ideal for SLAM applications.

A. Incremental Center and Covariance

Using Welford's formula [8], we incrementally update the center \mathbf{m} and covariance \mathbf{S} of \mathcal{P} as follows:

$$\mathbf{m}_k = \frac{(k-1)}{k} \mathbf{m}_{k-1} + \frac{1}{k} \mathbf{p}_k \quad (6)$$

$$\mathbf{S}_k = \mathbf{S}_{k-1} + \frac{k-1}{k} (\mathbf{p}_k - \mathbf{m}_{k-1})(\mathbf{p}_k - \mathbf{m}_{k-1})^T \quad (7)$$

where \mathbf{m}_k and \mathbf{S}_k are updated from \mathbf{m}_{k-1} and \mathbf{S}_{k-1} , \mathbf{p}_k is the newly added point. Define $\mathbf{D} = (\mathbf{p}_k - \mathbf{m}_{k-1})(\mathbf{p}_k - \mathbf{m}_{k-1})^T$ and combine (1), the normalized covariance \mathbf{A} can be updated incrementally as follows:

$$\mathbf{A}_k = \frac{k-1}{k} \mathbf{A}_{k-1} + \frac{k-1}{k^2} \mathbf{D} \quad (8)$$

The updated covariance \mathbf{A}_k decomposes into two terms: the former scales the existing point cloud's covariance, while the latter captures the contribution from the new point \mathbf{p}_k .

B. Incremental Jacobian of Eigenvalues

Deducing from (8), the partial derivative of the j -th eigenvalue λ_j w.r.t. the point \mathbf{p}_i , $i < k$, can be updated incrementally as follows:

$$\begin{aligned} \frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_i} &= \frac{k-1}{k} \frac{\partial \lambda_{j,k-1}}{\partial \mathbf{p}_i} \\ &- \frac{d_u}{k^2 \cos \theta_j} (\cos \varphi_{j,k-1} \mathbf{v}_{j,k-1}^T + \cos \varphi_{j,k} \mathbf{v}_{j,k-1}^T) \end{aligned} \quad (9)$$

where d_u is the distance between \mathbf{p}_k and \mathbf{m}_{k-1} , θ_j is the angle between $\mathbf{v}_{j,k}$ and $\mathbf{v}_{j,k-1}$. Define $\mathbf{v}_u = (\mathbf{p}_k - \mathbf{m}_{k-1})$, then $\varphi_{j,k}$ and $\varphi_{j,k-1}$ are the angles from the direction of \mathbf{v}_u to the j -th eigenvector $\mathbf{v}_{j,k}$ derived from k points and the j -th eigenvector $\mathbf{v}_{j,k-1}$ derived from $k-1$ points, respectively. The proof of (9) is provided in Appendix I-B.

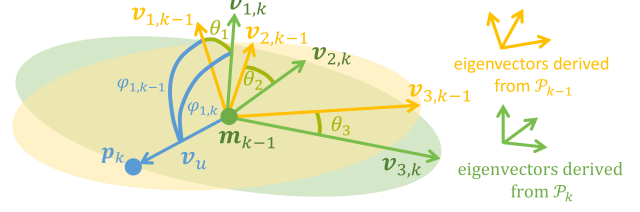


Fig. 2. An illustration of geometric relationship for the \mathbf{m}_{k-1} local frame.

Figure 2 visually depicts the geometric relationships involved in (9). To clarify the geometric meaning, let's consider \mathbf{v}_1 , the normal of a plane, as an example, which determines the direction for calculating the point-to-plane distance. Suppose we have sampled a sufficiently large number of points from the plane and \mathbf{p}_k lies on it, then $\theta_1 \approx 0$ and $\varphi_{1,k-1} = \varphi_{1,k} \approx \pi/2$. Consequently, the second term of $\partial \lambda_{1,k} / \partial \mathbf{p}_i$ is approximately equal to 0.

Furthermore, the second term (incremental terms) of (9) are identical for the first $k-1$ points. This characteristic allows for their efficient computation without the need to traverse \mathcal{P} . We take the magnitude of the second term of (9) to determine the updated form of the Jacobian matrix of the eigenvalues. If the magnitude is smaller than a threshold, we approximately update the Jacobian of eigenvalues as follows:

$$\mathbf{J}_{\lambda_j, \mathbf{p}_i, k} = \frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_i} \approx \frac{k-1}{k} \frac{\partial \lambda_{j,k-1}}{\partial \mathbf{p}_i}, i < k. \quad (10)$$

Otherwise, the Jacobian is updated as in BALM [4]:

$$\frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_i} = \frac{2}{k} (\mathbf{p}_i - \mathbf{m}_k)^T \mathbf{v}_{j,k} \mathbf{v}_{j,k}^T. \quad (11)$$

Considering that \mathbf{p}_k is independent of \mathbf{A}_{k-1} , the Jacobian matrix of eigenvalue w.r.t. \mathbf{p}_k can be simplified as:

$$\frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_k} = \frac{d_u (k-1)}{k^2 \cos \theta_j} (\cos \varphi_{j,k-1} \mathbf{v}_{j,k-1}^T + \cos \varphi_{j,k} \mathbf{v}_{j,k-1}^T), \quad (12)$$

which is proved in Appendix I-B.

C. Incremental Jacobian of Eigenvectors

According to BALM [4], the Jacobian matrix of the eigenvector \mathbf{v}_j w.r.t. \mathbf{p}_i can be represented as $\mathbf{J}_{\mathbf{v}_j, \mathbf{p}_i} = \partial \mathbf{v}_j / \partial \mathbf{p}_i = \mathbf{V} \mathbf{C}$, where \mathbf{V} is the eigenmatrix as shown in (2). For the elements in m -th row and n -th column in \mathbf{C} :

$$\mathbf{C}_{m,n}^{\mathbf{p}_i} = \begin{cases} \frac{(\mathbf{p}_i - \mathbf{m}_k)^T}{k(\lambda_n - \lambda_m)} (\mathbf{v}_m \mathbf{v}_n^T + \mathbf{v}_n \mathbf{v}_m^T), & m \neq n \\ \mathbf{0}, & m = n. \end{cases} \quad (13)$$

And (13) can be further simplified as $\mathbf{C}_{m,n}^{\mathbf{p}_i} = [\mathbf{C}_{m,n}^{x_i} \quad \mathbf{C}_{m,n}^{y_i} \quad \mathbf{C}_{m,n}^{z_i}] \in \mathbb{R}^{1 \times 3}$, $m, n \in \{1, 2, 3\}$, where the subscripts x_i , y_i , and z_i are elements of \mathbf{p}_i . Since diagonal

elements in \mathbf{C} are equal to 0, the rest of this section only specifies the form of the off-diagonal elements.

Analogous to the Jacobian matrix of the eigenvalues, the above matrix \mathbf{C} w.r.t \mathbf{p}_i , $i < k$, can be incrementally updated, which is also divided into scale and increment parts:

$$\begin{aligned} (\mathbf{C}_{m,n}^{\mathbf{p}_i})_k &= [(\mathbf{C}_{m,n}^{x_i})_k \quad (\mathbf{C}_{m,n}^{y_i})_k \quad (\mathbf{C}_{m,n}^{z_i})_k]_{1 \times 3} \\ &= \mathbf{W}_{m,n}^{\mathbf{p}_i} (\mathbf{C}_{m,n}^{\mathbf{p}_i})_{k-1} \\ &\quad - \frac{d_u}{k^2(\lambda_n - \lambda_m)_k} (\cos \varphi_m \mathbf{v}_n^T + \cos \varphi_n \mathbf{v}_m^T)_k \end{aligned} \quad (14)$$

where φ_m and φ_n are angles from \mathbf{v}_u to the m -th eigenvector \mathbf{v}_m and the n -th eigenvector \mathbf{v}_n , respectively. We specify the form of $\mathbf{W}_{m,n}^{\mathbf{p}_i}$ and prove (14) in Appendix I-C.

With the assumption that a sufficiently large number of points are already sampled from the surface, the second term of (14) is approximately equal to $\mathbf{0}$. Take \mathbf{v}_1 , i.e. normal \mathbf{n} , of the surface as an example, the partial derivative of the updated \mathbf{n} w.r.t. \mathbf{p}_i can be approximately updated as follows:

$$\begin{aligned} \mathbf{J}_{\mathbf{n},\mathbf{p}_i,k} &= \mathbf{V}_k (\mathbf{C}_{\mathbf{n},\mathbf{p}_i})_k \approx \mathbf{V}_k \mathbf{W} (\mathbf{C}_{\mathbf{n},\mathbf{p}_i})_{k-1} \\ &= \mathbf{V}_k \mathbf{W} \mathbf{V}_{k-1}^T \mathbf{V}_{k-1} (\mathbf{C}_{\mathbf{n},\mathbf{p}_i})_{k-1} \\ &= \underbrace{\mathbf{V}_k \mathbf{W} \mathbf{V}_{k-1}^T}_{\mathbf{Q}} \mathbf{J}_{\mathbf{n},\mathbf{p}_i,k-1} \end{aligned} \quad (15)$$

where we omit some of the subscripts to simplify the representation.

Considering that \mathbf{p}_k is independent of \mathbf{A}_{k-1} , the matrix \mathbf{C} w.r.t. \mathbf{p}_k can be simplified as:

$$\begin{aligned} (\mathbf{C}_{m,n}^{\mathbf{p}_k})_k &= [(\mathbf{C}_{m,n}^{x_k})_k \quad (\mathbf{C}_{m,n}^{y_k})_k \quad (\mathbf{C}_{m,n}^{z_k})_k]_{1 \times 3} \\ &= \frac{d_u(k-1)}{k^2(\lambda_n - \lambda_m)_k} (\cos \varphi_m \mathbf{v}_n^T + \cos \varphi_n \mathbf{v}_m^T)_k. \end{aligned} \quad (16)$$

We prove (16) in Appendix I-C and define $\mathbf{J}_{\mathbf{v}_j,\mathbf{p}_i,k} = \partial \mathbf{v}_j,k / \partial \mathbf{p}_i$ to simplify the representation.

D. Incremental Update of Covariance

Using the above incremental Jacobian matrix of the eigenvalues (10) and the eigenvectors (15), their covariance can be updated in an approximate form.

We simply divide the covariance of the eigenvalues corresponding to specified points into scale and increment parts as follows:

$$\begin{aligned} \mathbf{A}_{\lambda_j,k} &= \sum_i^{k-1} \mathbf{J}_{\lambda_j,\mathbf{p}_i,k} \mathbf{A}_{\mathbf{p}_i} \mathbf{J}_{\lambda_j,\mathbf{p}_i,k}^T + \mathbf{J}_{\lambda_j,\mathbf{p}_k,k} \mathbf{A}_{\mathbf{p}_k} \mathbf{J}_{\lambda_j,\mathbf{p}_k,k}^T \\ &= \frac{(k-1)^2}{k^2} \mathbf{A}_{\lambda_j,k-1} + \mathbf{J}_{\lambda_j,\mathbf{p}_k,k} \mathbf{A}_{\mathbf{p}_k} \mathbf{J}_{\lambda_j,\mathbf{p}_k,k}^T \end{aligned} \quad (17)$$

Similarly, the covariance of the eigenvector can be updated as follows:

$$\mathbf{A}_{\mathbf{v}_j,k} = \mathbf{Q} \mathbf{A}_{\mathbf{v}_j,k-1} \mathbf{Q}^T + \mathbf{J}_{\mathbf{v}_j,\mathbf{p}_k,k} \mathbf{A}_{\mathbf{p}_k} \mathbf{J}_{\mathbf{v}_j,\mathbf{p}_k,k}^T \quad (18)$$

where we omit some of the subscripts to simplify the representation.

This fast covariance approximation method can be seamlessly integrated with other statistical metrics that share similar scaling properties. Derived from (6), we get the Jacobian

matrix of the center \mathbf{m}_k w.r.t \mathbf{p}_i as $\mathbf{J}_{\mathbf{m},\mathbf{p}_i,k} = \frac{\partial \mathbf{m}_k}{\partial \mathbf{p}_i} = \frac{1}{k} \mathbf{I}_{3 \times 3}$ easily. Furthermore, $\mathbf{J}_{\mathbf{m},\mathbf{p}_i}$ can be incrementally updated as follows:

$$\mathbf{J}_{\mathbf{m},\mathbf{p}_i,k} = \frac{k-1}{k} \mathbf{J}_{\mathbf{m},\mathbf{p}_i,k-1}. \quad (19)$$

Combining (18) and (19), the covariance of the normal \mathbf{n} and the center \mathbf{m} can be fast approximated as follows:

$$\begin{aligned} \mathbf{A}_{\mathbf{n},\mathbf{m},k} &= \begin{bmatrix} \mathbf{Q} \mathbf{A}_{\mathbf{n},k-1} \mathbf{Q}^T & \frac{k-1}{k} \mathbf{Q} \mathbf{A}_{\mathbf{nm},k-1} \\ \frac{k-1}{k} \mathbf{A}_{\mathbf{mn},k-1} \mathbf{Q}^T & (\frac{k-1}{k})^2 \mathbf{A}_{\mathbf{m},k-1} \end{bmatrix} \\ &\quad + \begin{bmatrix} \mathbf{J}_{\mathbf{n},\mathbf{p}_k,k} \mathbf{A}_{\mathbf{p}_k} \mathbf{J}_{\mathbf{n},\mathbf{p}_k,k}^T & \frac{1}{k} \mathbf{J}_{\mathbf{n},\mathbf{p}_k,k} \mathbf{A}_{\mathbf{p}_k} \\ \frac{1}{k} \mathbf{A}_{\mathbf{p}_k} \mathbf{J}_{\mathbf{n},\mathbf{p}_k,k}^T & \frac{1}{k^2} \mathbf{A}_{\mathbf{p}_k} \end{bmatrix} \end{aligned} \quad (20)$$

In contrast to methods like VoxelMap [5], (20) offers a significant computational advantage by achieving a constant time complexity of $\mathcal{O}(1)$. This translates to faster execution, making it well-suited for real-time applications.

VI. INTEGRATION WITH LIO

In this section, we integrate the proposed point uncertainty model along with LUFA into our prior work [9], namely LOG-LIO2. The system state \mathbf{x} is as follows:

$$\mathbf{x} = [\mathcal{W} \mathbf{R}_{\mathcal{I}}^T \quad \mathcal{W} \mathbf{p}_{\mathcal{I}}^T \quad \mathcal{W} \mathbf{v}_{\mathcal{I}}^T \quad \mathcal{b}_{\omega}^T \quad \mathcal{b}_a^T \quad \mathcal{W} \mathbf{g}^T] \quad (21)$$

where $\mathcal{W} \mathbf{R}_{\mathcal{I}}^T$, $\mathcal{W} \mathbf{p}_{\mathcal{I}}^T$, and $\mathcal{W} \mathbf{v}_{\mathcal{I}}^T$ are the orientation, position, and velocity of IMU in the world frame. \mathcal{b}_{ω}^T and \mathcal{b}_a^T are gyroscope and accelerometer bias respectively. $\mathcal{W} \mathbf{g}^T$ is the known gravity vector in the world frame.

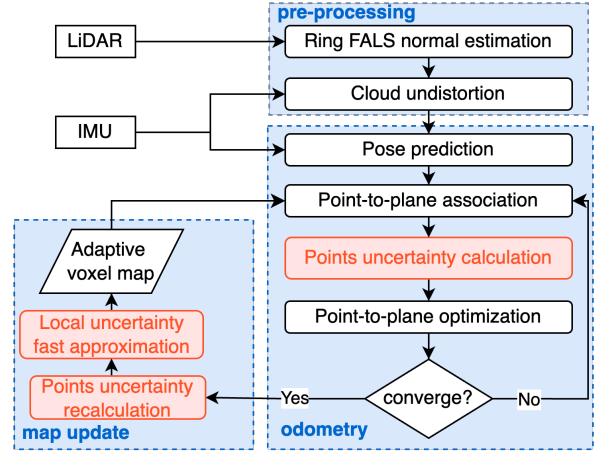


Fig. 3. System overview of LOG-LIO2 with enhancements marked in red.

The system structure is depicted in Figure 3. It comprises three modules: pre-processing, odometry, and map update. We employ adaptive voxelization from VoxelMap [5] for map management, leveraging its efficient data association and incorporating the computation of within-voxel covariance.

A. Pre-processing

For a point \mathbf{p}_i within the new input scan, we first estimate its normal \mathbf{n}_{r_i} using Ring FALS [9]. Subsequently, point cloud undistortion is performed to compensate for LiDAR motion based on IMU measurements [10].

B. Odometry

By incorporating the IMU measurements from the previous scan as a prediction $\hat{\mathbf{x}}_t$ for the current scan, we transform each point into the world frame. Each point is then assigned to the voxel it resides in if the points within that voxel satisfy planar geometry (indicated by the minimum eigenvalue below a threshold). Suppose \mathbf{p}_i associates with the l -th voxel v_l , we compute the point-to-plane distance as:

$$\mathbf{z}_i = \mathbf{n}_{v_l}^T (\mathbf{p}_i - \mathbf{m}_{v_l}). \quad (22)$$

where \mathbf{m}_{v_l} and \mathbf{n}_{v_l} denote the center and normal of v_l , respectively. Notably, \mathbf{n}_{v_l} differs from \mathbf{n}_{r_i} in that it incorporates geometric information from all map points within v_l , making it more suitable for the incident angle calculation (3), whereas \mathbf{n}_{r_i} focuses on the local geometry of individual points in the scan. Both \mathbf{n}_{v_l} and \mathbf{n}_{r_i} contribute to roughness estimation (4).

We denote the propagated state and covariance by $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{P}}_t$ respectively. By incorporating the prior distribution and stacking all the point-to-plane associations, we obtain the maximum a-posterior estimate (MAP) as follows [10]:

$$\min_{\hat{\mathbf{x}}_t} \left(\|\mathbf{x}_t \ominus \hat{\mathbf{x}}_t\|_{\hat{\mathbf{P}}_t^{-1}}^2 + \sum_{i \in \text{plane}} \|\mathbf{z}_i^\kappa + \mathbf{H}_i^\kappa \hat{\mathbf{x}}_t^\kappa\|_{\mathbf{A}_{\mathbf{p}_i, \mathbf{n}_{v_l}, \mathbf{m}_{v_l}}^{-1}}^2 \right) \quad (23)$$

where \ominus computes the difference between \mathbf{x}_t and $\hat{\mathbf{x}}_t$ in the local tangent space of \mathbf{x}_t , $\hat{\mathbf{x}}_t^\kappa$ is the error of the κ -th iterate update at time t , \mathbf{H}_i^κ is the Jacobian matrix w.r.t. $\hat{\mathbf{x}}_t^\kappa$. The point-to-plane distance is weighted by $\mathbf{A}_{\mathbf{p}_i, \mathbf{n}_{v_l}, \mathbf{m}_{v_l}}^{-1}$, which incorporates point, normal, and center uncertainties. We initialize the point-wise uncertainty in the LiDAR frame as detailed in Section IV and then transform it to the world frame. Normal and center uncertainties are estimated by LUFA using map points in the corresponding voxel. We employ iEKF [10] to optimize the system state.

C. Map update

After optimization, each point is assigned to the corresponding voxel with the updated state. A crucial aspect of this module lies in propagating the uncertainty from the scan points to the voxel center, as well as its eigenvalues and eigenvectors. The uncertainty of points is first recalculated based on the updated state. Apart from increment magnitude checks (see Section V), we employ LUFA considering two additional factors:

1) *Map stability*: To strike a balance between initial map instability and computational efficiency, we employ LUFA after points within a voxel exceed a predefined threshold n_{min} .

2) *Error accumulation*: The fast approximation introduces small errors with each iteration. To alleviate the accumulation of these errors, we limit the continuous application of LUFA to a maximum of $n_{ct} = 100$ iterations.

Beyond the above conditions, the rigorous update forms (11) and (13) are employed. Additionally, once the point count reaches n_{max} , we fix the voxel update using rigorous forms to ensure the accuracy of the local geometric information. This adaptive strategy effectively balances stability and efficiency during map updates.

VII. EXPERIMENTAL RESULTS

A. Experimental Settings

The experiment focuses on the following two research questions:

- Can LUFA approximate the uncertainty computed in rigorous form?
- Can LOG-LIO2 improve accuracy and efficiency by incorporating our point uncertainty model and LUFA?

We first validate LUFA in a simulation environment and then further test the performance of LOG-LIO2 on real-world datasets. Our workstation runs with Ubuntu 18.04, equipped with an Intel Core Xeon(R) Gold 6248R 3.00GHz processor and 32GB RAM.

B. LUFA Experiments

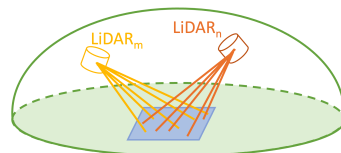


Fig. 4. Illustration of the simulation environment.

To assess the accuracy and efficiency of LUFA, we benchmark it against BALM [4]. Figure 4 illustrates the validation environment, simulating a $20m \times 20m$ plane with 20 LiDARs, each LiDAR captures 50 points on the plane. All LiDARs are positioned on the same side of the plane and remain within a 100m radius of its center. The center and normal of the plane, the poses of the LiDARs, and the coordinates of the sampled points on the plane are all randomly generated. To mimic real-world scenarios, we corrupt all these parameters with Gaussian noise, accounting for factors such as the plane's unevenness, inaccuracies in the LiDAR poses, and measurement errors. The point uncertainty is calculated as detailed in Section IV. We set $n_{min} = 200$ to define when LUFA is triggered.

1) *Accuracy*: Figure 5 illustrates the covariance of the λ_j and the covariance trace of \mathbf{v}_1 computed by LUFA and BALM. As the covariance of the eigenvectors involves matrices, we compare the trace of the covariance matrices in our experiments, which provides a measure of the overall magnitude of the covariance. The results indicate that the covariance computed by LUFA is close to that of BALM. This closeness in covariance values demonstrates the effectiveness of LUFA in approximating the uncertainty propagation while maintaining a reasonable level of accuracy.

2) *Efficiency*: We further plot the processing time w.r.t the number of points for better evaluation in Figure 6. As expected, BALM exhibits linear time complexity $\mathcal{O}(n)$, resulting in processing time increases proportionally with the number of points. In contrast, the processing time of LUFA remains nearly constant, adhering to $\mathcal{O}(1)$ time complexity. The observed peaks in LUFA's processing time occur at intervals of 100 points, which stems from our imposed limitation on the maximum number of consecutive LUFA

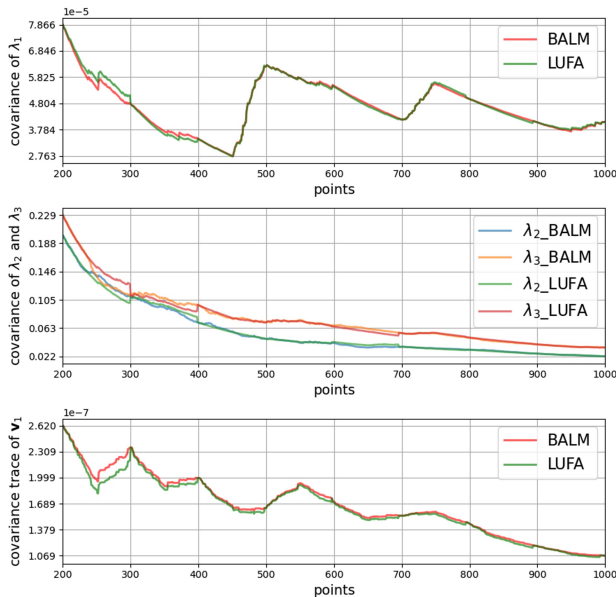


Fig. 5. Comparison of covariance calculated by LUFA and BALM.

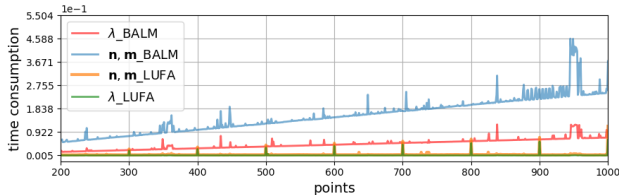


Fig. 6. Time consumption for computing the covariance. \mathbf{n} , \mathbf{m} represent the covariance of the normal and the center.

executions. Utilizing the incremental center (6) and covariance (7) ensures that even at peak processing times, LUFA remains computationally more efficient than BALM.

By comparing the results obtained using LUFA with those from BALM, we demonstrate that LUFA achieves comparable accuracy while significantly improving computational efficiency.

C. LIO Experiments

To evaluate the efficacy of LOG-LIO2 in real-world scenarios, we employ the M2DGR dataset [12]. This dataset collects data on a ground platform equipped with Velodyne-32 LiDAR and presents significant challenges for our previous method, LOG-LIO. To isolate the impact of specific components within LOG-LIO2, we introduce LOG2-i. LOG2-i differs from LOG-LIO2 in its map uncertainty update mechanism, employing the formulation outlined in BALM instead of the LUFA approach. We compare the performance of LOG-LIO2 against two leading LIO methods: FAST-LIO2 [10] and PV-LIO¹. PV-LIO is the reimplementation of VoxelMap [5] integrating IMU, showing promising results in practice. Both PV-LIO and VoxelMap use BALM’s formulation for uncertainty propagation.

¹<https://github.com/HViktorTsoi/PV-LIO>

To ensure a fair comparison, LOG-LIO2, LOG2-i, and PV-LIO use identical parameters across all dataset sequences: maximum voxel size of 1.6m and maximum octree layer of 3, resulting in a minimum voxel size of 0.2m. FAST-LIO2 and LOG-LIO are run with a map resolution of 0.4m using ikd-tree. Additionally, we set n_{min} and n_{max} to 200 and 1000 respectively, defining the thresholds for when LUFA is triggered and when it is no longer necessary. Due to the instability in the RTK signal, the first and last 100 seconds of sequences *street_07* and *street_10* are excluded from the evaluation. Note that all the LIO systems above perform scan-to-map registration by minimizing point-to-plane distances and loop closure was disabled for all experiments.

1) *Accuracy Evaluation*: Table I reports the root-mean-square-error (RMSE) of absolute trajectory error (ATE). LOG-LIO2 and LOG2-i stand out as the most accurate LIO systems, exhibiting comparable performance, with LOG-LIO and PV-LIO following behind. The advantages of LOG-LIO2, LOG2-i, and PV-LIO stem from a combination of an adaptive voxel map and uncertainty-weighted point-to-plane distance for scan-to-map registration. This approach offers a more accurate geometric representation, leading to minimal registration errors. Conversely, LOG-LIO and FAST-LIO2 depend on a fixed-scale tree structure for map management and implement isotropic noise-weighted point-to-plane distance. Notably, LOG-LIO distinguishes itself from FAST-LIO2 by incrementally maintaining normal and point distribution within map nodes, thereby enabling the capture of geometric complexities.

TABLE I
THE TRANSLATION RMSE(M) RESULTS OF POSE ESTIMATION
COMPARISON ON THE M2DGR DATASET

| | LOG-LIO2 | LOG2-i | LOG-LIO | PV-LIO | FAST-LIO2 |
|-----------|--------------|--------------|--------------|--------------|--------------|
| walk_01 | 0.131 | 0.132 | <u>0.117</u> | 0.135 | 0.112 |
| door_01 | <u>0.264</u> | 0.262 | 0.266 | 0.262 | 0.271 |
| door_02 | 0.197 | 0.197 | <u>0.196</u> | 0.194 | 0.200 |
| street_01 | 0.343 | 0.303 | <u>0.291</u> | 0.439 | 0.272 |
| street_02 | <u>2.625</u> | 2.541 | 3.252 | 3.540 | 2.754 |
| street_03 | 0.104 | 0.104 | 0.092 | <u>0.093</u> | 0.106 |
| street_04 | 1.032 | 1.079 | <u>0.697</u> | 1.081 | 0.552 |
| street_05 | 0.426 | <u>0.370</u> | 0.306 | 0.543 | 0.377 |
| street_06 | <u>0.355</u> | 0.338 | <u>0.355</u> | 0.494 | 0.434 |
| street_07 | <u>0.358</u> | 0.339 | 0.422 | 0.651 | 3.512 |
| street_08 | 0.166 | 0.156 | <u>0.140</u> | 0.138 | 0.170 |
| street_09 | <u>1.822</u> | 1.861 | 2.380 | 1.678 | 3.648 |
| street_10 | 0.314 | 0.369 | <u>0.349</u> | 0.464 | 0.956 |
| mean | <u>0.626</u> | 0.619 | 0.682 | 0.747 | 1.028 |

The best and second-best results are bolded and underlined respectively.

LOG2-i and LOG-LIO2 outperform PV-LIO in most sequences due to our comprehensive point uncertainty model. PV-LIO’s model suffers from limitations: constant uncertainty in the ray direction and neglecting surface geometry. In contrast, our model considers incident angle and surface roughness, providing a more precise environmental representation. This advantage is evident in large outdoor environments like the *street* sequences. In smaller indoor spaces like *door* sequences, all systems achieve comparable performance due to lower point uncertainty and well-defined

structural planes. Figure 7 compares the trajectories in the *street_07* sequence, highlighting the performance difference.

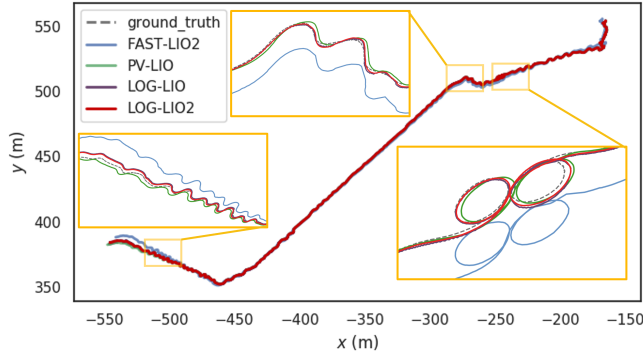


Fig. 7. Localization estimates in sequence *street_07* of the M2DGR dataset.

2) *Efficiency Evaluation*: Table II details the processing times of LOG-LIO2, LOG2-i, and PV-LIO for the *street* sequence, highlighting map update and runtime behavior across different thread configurations. Note that Ring FALS introduces an additional 6ms overhead within the pre-processing module of LOG-LIO2 and LOG2-i. Despite this overhead, comparing the average time consumption with 4 threads, LOG2-i exhibits a slightly shorter duration than PV-LIO, mainly benefiting from the fast uncertainty calculation (IV-B), increment center, and covariance (V-A). LOG-LIO2 stands out for its efficiency with LUFA, significantly reducing map update times compared to PV-LIO, operating nearly half the time of PV-LIO with a single thread and two-thirds the time with four threads.

TABLE II

THE AVERAGE TIME CONSUMPTION(MS) OF STREET SEQUENCE IN THE EXPERIMENTS

| thread(s) | LOG-LIO2 | | LOG2-i | | PV-LIO | |
|------------|----------|--------------|--------|--------------|--------|-------|
| | 1 | 4 | 1 | 4 | 1 | 4 |
| map update | 19.13 | 9.29 | 28.67 | 11.92 | 37.98 | 14.56 |
| total | 59.20 | 50.77 | 68.68 | <u>52.83</u> | 80.27 | 59.47 |

The best and second-best results are bolded and underlined respectively.

VIII. CONCLUSION

This paper presents a comprehensive point uncertainty model alongside a fast calculation method utilizing the projection operator. This model incorporates not only the range and bearing uncertainty of LiDAR but also the uncertainty arising from incident angle and surface roughness. Furthermore, we derive the incremental Jacobian matrices of the eigenvalues and eigenvectors, enabling the application of the local uncertainty fast approximation (LUFA). The accuracy and efficiency of our formulations are first demonstrated through simulation experiments, benchmarking against the rigorous form derived by BALM. Subsequently, we integrate all aforementioned methods into the LIO system LOG-LIO2.

Ablation experiments conducted on a public dataset validate the accuracy and efficiency of LOG-LIO2 compared to state-of-the-art LIO systems.

APPENDIX I

A. Consistency of Projection Operator with \mathbb{S}^2 Perturbation

In [2], [4], [5], the point uncertainty model considering range and bearing is derived from the perturbation of the true range d_i^{gt} and true bearing ω_i^{gt} as:

$$\mathbf{p}_i^{\text{gt}} = d_i^{\text{gt}} \omega_i^{\text{gt}} = (d_i + \sigma_{d_i}) (\omega_i \boxplus_{\mathbb{S}^2} \sigma_{\omega_i}) \quad (24)$$

where d_i and ω_i is range and bearing measurements respectively, \boxplus -operation encapsulated in \mathbb{S}^2 [11], $\sigma_{d_i} \sim \mathcal{N}(0, \Sigma_{d_i})$ and $\sigma_{\omega_i} \sim \mathcal{N}(\mathbf{0}_{2 \times 1}, \Sigma_{\omega_i})$ are range and bearing noise, respectively. The Jacobian matrix of \mathbf{p}_i w.r.t. d_i and σ_{ω_i} can be further obtained as $\mathbf{J}_{d_i, \omega_i} = [\omega_i \quad -d_i [\omega_i]_{\times} \mathbf{N}(\omega_i)]$, where $[\]_{\times}$ denotes the skew-symmetric matrix mapping the cross product, and $\mathbf{N}(\omega_i) = [\mathbf{N}_1 \ \mathbf{N}_2] \in \mathbb{R}^{3 \times 2}$ is the orthogonal basis of the tangent plane at ω_i . Then the covariance of the point leads to:

$$\mathbf{A}_{\omega_i} = \mathbf{J}_{d_i, \omega_i} \begin{bmatrix} \sigma_{d_i}^2 & \mathbf{0}_{1 \times 2} \\ \mathbf{0}_{2 \times 1} & \Sigma_{\omega_i}^2 \end{bmatrix} \mathbf{J}_{d_i, \omega_i}^T \quad (25)$$

Despite constructing $\mathbf{N}(\omega_i)$ arbitrarily in the tangent plane of ω_i is feasible, this approach incurs a computational cost. We show that utilization of the projection operator simplifies the computation while maintaining consistency:

$$\begin{aligned} \mathbf{A}_{\omega_i} &= \sigma_{d_i}^2 \omega_i \omega_i^T + d_i^2 \sigma_{\omega}^2 [\omega_i]_{\times} \mathbf{N}(\omega_i) \mathbf{I}_{2 \times 2} \mathbf{N}(\omega_i)^T [\omega_i]_{\times}^T \\ &= \sigma_{d_i}^2 \omega_i \omega_i^T + d_i^2 \sigma_{\omega}^2 [\omega_i]_{\times} (-[\omega_i]_{\times}) \\ &= \sigma_{d_i}^2 \omega_i \omega_i^T + d_i^2 \sigma_{\omega}^2 (\mathbf{I}_{3 \times 3} - \omega_i \omega_i^T). \end{aligned} \quad (26)$$

Therefore, by eliminating the need for constructing $\mathbf{N}(\omega_i)$, our approach results in a more efficient method.

B. Proof of The Incremental Jacobian of Eigenvalue

Similar to BALM [4], in this section, eigenvectors are viewed as constant. Multiplying (8) by $\mathbf{v}_{j,k-1}^T$ on the left and $\mathbf{v}_{j,k}$ on the right, and combine with (2), we get:

$$\begin{aligned} \lambda_{j,k} \mathbf{v}_{j,k-1}^T \mathbf{v}_{j,k} &= \frac{k-1}{k} (\lambda_{j,k-1} \mathbf{v}_{j,k-1}^T \mathbf{v}_{j,k} + \frac{1}{k} \mathbf{v}_{j,k-1}^T \mathbf{D} \mathbf{v}_{j,k}) \\ \lambda_{j,k} &= \frac{k-1}{k} \lambda_{j,k-1} + \frac{k-1}{k^2 \cos \theta_j} \mathbf{v}_{j,k-1}^T \mathbf{D} \mathbf{v}_{j,k}, \end{aligned} \quad (27)$$

where θ_j is the angle between $\mathbf{v}_{j,k}$ and $\mathbf{v}_{j,k-1}$.

Then the partial derivative of the updated eigenvalue $\lambda_{j,k}$ w.r.t. \mathbf{p}_i is given by:

$$\frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_i} = \frac{k-1}{k} \frac{\partial \lambda_{j,k-1}}{\partial \mathbf{p}_i} + \frac{k-1}{k^2 \cos \theta_j} \frac{\partial \mathbf{v}_{j,k-1}^T \mathbf{D} \mathbf{v}_{j,k}}{\partial \mathbf{p}_i}. \quad (28)$$

Given the value of $\partial \lambda_{j,k-1} / \partial \mathbf{p}_i$ in the last update, only the second term of the above equation needs to be solved.

For $i = 1, \dots, k-1$, the second term of (28) is:

$$\begin{aligned} & \frac{k-1}{k^2 \cos \theta_j} \frac{\partial \mathbf{v}_{j,k-1}^T \mathbf{D} \mathbf{v}_{j,k}}{\partial \mathbf{p}_i} \\ &= \frac{k-1}{k^2 \cos \theta_j} \left[-\frac{\mathbf{v}_u^T \mathbf{v}_{j,k-1} \mathbf{v}_{j,k}^T}{(k-1)} - \frac{\mathbf{v}_u^T \mathbf{v}_{j,k} \mathbf{v}_{j,k-1}^T}{(k-1)} \right] \\ &= -\frac{d_u}{k^2 \cos \theta_j} (\cos \varphi_{j,k-1} \mathbf{v}_{j,k}^T + \cos \varphi_{j,k} \mathbf{v}_{j,k-1}^T) \end{aligned} \quad (29)$$

where d_u is the distance between \mathbf{p}_k and \mathbf{m}_{k-1} . $\varphi_{j,k}$ and $\varphi_{j,k-1}$ are the angles from the direction of \mathbf{v}_u to the j -th eigenvector $\mathbf{v}_{j,k}$ derived from k points and the j -th eigenvector $\mathbf{v}_{j,k-1}$ derived from $k-1$ points, respectively.

For \mathbf{p}_k , which is independent of $\lambda_{j,k-1}$, (28) simplifies to:

$$\begin{aligned} \frac{\partial \lambda_{j,k}}{\partial \mathbf{p}_k} &= \frac{k-1}{k^2 \cos \theta_j} \frac{\partial \mathbf{v}_{j,k-1}^T \mathbf{D} \mathbf{v}_{j,k}}{\partial \mathbf{p}_k} \\ &= \frac{k-1}{k^2 \cos \theta_j} (\mathbf{v}_u^T \mathbf{v}_{j,k-1} \mathbf{v}_{j,k}^T + \mathbf{v}_u^T \mathbf{v}_{j,k} \mathbf{v}_{j,k-1}^T) \\ &= \frac{d_u(k-1)}{k^2 \cos \theta_j} (\cos \varphi_{j,k-1} \mathbf{v}_{j,k}^T + \cos \varphi_{j,k} \mathbf{v}_{j,k-1}^T). \end{aligned} \quad (30)$$

C. Proof of The Incremental Jacobian of Eigenvectors

We first revisit the derivations of BALM [4] from (31) to (34). Reformulating (2), we get:

$$\mathbf{\Lambda} = \mathbf{V}^T \mathbf{A} \mathbf{V}; \mathbf{V} \mathbf{\Lambda} = \mathbf{A} \mathbf{V}; \mathbf{\Lambda} \mathbf{V}^T = \mathbf{V}^T \mathbf{A}. \quad (31)$$

Utilizing the orthogonality property $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, we can further obtain:

$$\mathbf{V}^T \frac{\partial \mathbf{V}}{\partial q_i} + \left(\frac{\partial \mathbf{V}}{\partial q_i} \right)^T \mathbf{V} = \mathbf{0}. \quad (32)$$

Denoting an element of \mathbf{p}_i by q_i and combining (31), the derivative of $\mathbf{\Lambda}_k$ w.r.t. q_i leads to:

$$\begin{aligned} \frac{\partial \mathbf{\Lambda}_k}{\partial q_i} &= \mathbf{V}_k^T \frac{\partial \mathbf{A}_k}{\partial q_i} \mathbf{V}_k + \left(\frac{\partial \mathbf{V}_k}{\partial q_i} \right)^T \mathbf{A}_k \mathbf{V}_k + \mathbf{V}_k^T \mathbf{A}_k \frac{\partial \mathbf{V}_k}{\partial q_i} \\ &= \mathbf{V}_k^T \frac{\partial \mathbf{A}_k}{\partial q_i} \mathbf{V}_k + \underbrace{\mathbf{\Lambda}_k \mathbf{V}_k^T \frac{\partial \mathbf{V}_k}{\partial q_i}}_{(\mathbf{C}^q)_k} + \underbrace{\left(\frac{\partial \mathbf{V}_k}{\partial q_i} \right)^T \mathbf{V}_k \mathbf{\Lambda}_k}_{(\mathbf{C}^q)_k^T}. \end{aligned} \quad (33)$$

Given that $\mathbf{\Lambda}$ is diagonal, the off-diagonal elements in (33) can be combined with (32) to yield:

$$0 = \mathbf{v}_{m,k}^T \frac{\partial \mathbf{A}_k}{\partial q} \mathbf{v}_{n,k} + \lambda_{m,k} (\mathbf{C}_{m,n}^q)_k - (\mathbf{C}_{m,n}^q)_k \lambda_{n,k}.$$

Here, $(\mathbf{C}_{m,n}^q)_k$ represents the element at the m -th row and n -th column of $(\mathbf{C}^q)_k$, defined as:

$$(\mathbf{C}_{m,n}^q)_k = \begin{cases} \frac{1}{(\lambda_n - \lambda_m)_k} \mathbf{v}_{m,k}^T \frac{\partial \mathbf{A}_k}{\partial q} \mathbf{v}_{n,k}, & m \neq n \\ 0, & m = n \end{cases} \quad (34)$$

To derive the incremental form of the matrix \mathbf{C} as described above, we derive the partial derivatives in (34) using (8):

$$\frac{\partial \mathbf{A}_k}{\partial q} = \frac{k-1}{k} \frac{\partial \mathbf{A}_{k-1}}{\partial q} + \frac{k-1}{k^2} \frac{\partial \mathbf{D}}{\partial q}. \quad (35)$$

The specific form of $(\mathbf{C}_{m,n}^q)_{k-1}$, $m \neq n$, derived from BALM is given by:

$$(\mathbf{C}_{m,n}^q)_{k-1} = \frac{1}{(\lambda_n - \lambda_m)_{k-1}} \mathbf{v}_{m,k-1}^T \frac{\partial \mathbf{A}_{k-1}}{\partial q} \mathbf{v}_{n,k-1}. \quad (36)$$

Similar to (29), for $j = 1, \dots, k-1$, we can further obtain:

$$\mathbf{v}_{m,k}^T \frac{\partial \mathbf{D}}{\partial \mathbf{p}_j} \mathbf{v}_{n,k} = -\frac{d_u}{k-1} (\cos \varphi_m \mathbf{v}_n^T + \cos \varphi_n \mathbf{v}_m^T)_k \quad (37)$$

where φ_m and φ_n denote angles from the direction of \mathbf{v}_u to \mathbf{v}_m and \mathbf{v}_n , respectively.

Substituting (35), (36) and (37) into (34) yields:

$$\begin{aligned} (\mathbf{C}_{m,n}^{\mathbf{p}_i})_k &= [(\mathbf{C}_{m,n}^{x_i})_k \quad (\mathbf{C}_{m,n}^{y_i})_k \quad (\mathbf{C}_{m,n}^{z_i})_k]_{1 \times 3} \\ &= \overbrace{\frac{(k-1)(\lambda_n - \lambda_m)_{k-1}}{k(\lambda_n - \lambda_m)_k} \cos \theta_m \cos \theta_n (\mathbf{C}_{m,n}^{\mathbf{p}_i})_{k-1}}^{\mathbf{W}_{m,n}^{\mathbf{p}_i}} \\ &\quad - \frac{d_u}{k^2(\lambda_n - \lambda_m)_k} (\cos \varphi_m \mathbf{v}_n^T + \cos \varphi_n \mathbf{v}_m^T)_k \end{aligned} \quad (38)$$

where θ_m denotes the angle between $\mathbf{v}_{m,k}$ and $\mathbf{v}_{m,k-1}$.

For \mathbf{p}_k , which is independent of \mathbf{A}_{k-1} , (34) simplifies to:

$$\begin{aligned} (\mathbf{C}_{m,n}^{\mathbf{p}_k})_k &= [(\mathbf{C}_{m,n}^{x_k})_k \quad (\mathbf{C}_{m,n}^{y_k})_k \quad (\mathbf{C}_{m,n}^{z_k})_k]_{1 \times 3} \\ &= \frac{k-1}{k^2(\lambda_n - \lambda_m)_k} \mathbf{v}_{m,k}^T \frac{\partial \mathbf{D}}{\partial \mathbf{p}} \mathbf{v}_{n,k} \\ &= \frac{d_u(k-1)}{k^2(\lambda_n - \lambda_m)_k} (\cos \varphi_m \mathbf{v}_n^T + \cos \varphi_n \mathbf{v}_m^T)_k. \end{aligned} \quad (39)$$

REFERENCES

- [1] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.
- [2] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, 2021.
- [3] B. Jiang and S. Shen, "A lidar-inertial odometry with principled uncertainty modeling," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 292–13 299.
- [4] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.
- [5] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8518–8525, 2022.
- [6] T. Tasdizen and R. Whitaker, "Cramer-rao bounds for nonparametric surface reconstruction from range data," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings*. IEEE, 2003, pp. 70–77.
- [7] K.-H. Bae, D. Belton, and D. D. Lichti, "A closed-form expression of the positional uncertainty for 3d point clouds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 4, pp. 577–590, 2008.
- [8] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.
- [9] K. Huang, J. Zhao, Z. Zhu, C. Ye, and T. Feng, "Log-lid: A lidar-inertial odometry with efficient local geometric information estimation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 459–466, 2023.
- [10] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lid2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.
- [11] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," *arXiv preprint arXiv:2102.03804*, 2021.
- [12] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, "M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2266–2273, 2021.