2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11246498

Adaptive Sliding Window Optimization for Multi-Modal LiDAR Inertial Odometry and Mapping

Guodong Han[1,2], Wei Li[1,3*], *Member, IEEE*, Yu Hu[1,3*], *Member, IEEE*

***Abstract*— Fixed-Lag smoothing is widely employed as a backend in localization tasks. Generally, increasing the window length leads to better accuracy, but demands more computational resources. Therefore, determining an appropriate window length and whether a fixed length should be maintained throughout the localization process are worth studying. Assuming independent and identically distributed noise based on the distance-independent characteristic of LiDAR ranging errors, we propose an uncertainty-based adaptive sliding window (ASW) strategy. Through mathematical derivation, the reference uncertainty is affected by the LiDAR feature distribution of each frame. Consequently, we develop a multi-modal LiDAR inertial odometry and mapping framework based on ASW, which integrates mechanical and solid-state LiDAR to enhance odometry accuracy and mapping density. By designing a joint matching module, our approach leverages the strengths of distinct scanning patterns. Additionally, we incorporate loop closure detection in the mapping process to minimize cumulative drift. Extensive experiments conducted on both public and self-collected datasets demonstrate the effectiveness of our method. Compared to the state-of-the-art method, our approach improves the average accuracy by 10.3%. We also provide an open-source implementation for further studies. https://github.com/wowhhhhgd/ASW-LIOM.**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is an important technology in the field of autonomous navigation. Compared to cameras, Light Detection and Ranging (LiDAR) can provide more accurate and reliable ranging information, while being less sensitive to lighting conditions. There are two types of LiDAR systems: mechanical LiDAR and solid-state LiDAR, which have different scanning patterns. Mechanical LiDAR makes use of a rotating scanning mechanism to generate parallel scan lines, achieving a 360-degree field of view (FoV). But the main weakness is that the vertical angular resolution is positively correlated with price. On the other hand, solid-state LiDAR utilizes optoelectronic devices and phase control techniques instead of moving components to emit and receive laser pulses. This makes solid-state LiDAR gain the advantages of lightweight and low cost, but along with the disadvantage of narrow FoV. Due to the irregular scanning pattern of solid-state LiDAR, it can provide point supplementation for the vertical angular resolution of mechanical LiDAR, which leads to the potential for further improving the accuracy of multi-modal LiDAR odometry and mapping.

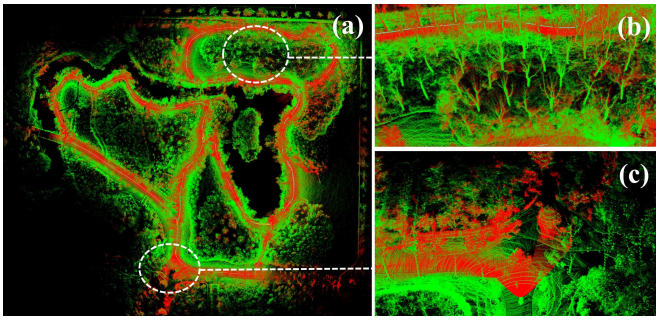


Fig. 1: Mapping results of the proposed method on Botanic Garden dataset sequence 1008-01. (a) Mapping result. (b)(c) Details of the result. The red and blue points are from solid-state and mechanical LiDAR respectively.

Sliding window optimization is widely used in LiDAR SLAM [1]–[4]. Fixed-Lag smoothing (FLS) improves consistency over filters by re-estimating a limited window of past states using new measurements. As the window length increases, more historical observations are taken into account, which typically leads to higher computational costs and more accurate positioning results. Based on this, how to select an appropriate window length, as well as if it should be consistent throughout the entire positioning process, remains to be studied. Most existing methods set the window length to be fixed during the entire SLAM process. In fact, if there are sufficient effective LiDAR features or small estimation variances currently, it is no need for a long time window to achieve the expected accuracy. Adaptive adjustment of the optimization window length can be guided by monitoring the uncertainty, as reflected in the covariance, of the state estimates. From a mathematical perspective, the uncertainty metric can serve as a predictor of localization accuracy. By modeling sensor noise, explicit uncertainty of the sensor-derived constraints in SLAM front-end can be calculated. This enables an evaluation of whether incorporating observations at a given time step reduces or increases the overall localization uncertainty during optimization. Because LiDAR ranging errors are independent of distance, constraints obtained from LiDAR observations fulfill the conditions for the independent and identically distributed (i.i.d.) assumption. Based on mathematical derivation, we introduce a quantitative method for evaluating the effectiveness of LiDAR feature matching constraints. Building upon the results of this evaluation, we propose an adaptive sliding window (ASW)

This work was supported by Beijing Natural Science Foundation (L243008), and in part by National Natural Science Foundation of China under Grant No. 62003323 and No. 62176250.

[1]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
[2]School of Info. Sci. and Tech., ShanghaiTech University, China.
[3]University of Chinese Academy of Sciences, Beijing, 100049, China.
*Correspondence: Wei Li, liwei2019@ict.ac.cn, Yu Hu, huyu@ict.ac.cn.

strategy, aiming to improve real-time performance while maintaining accuracy. Finally, a SLAM algorithm using ASW and leveraging complementary scanning pattern of mechanical and solid-state LiDAR is implemented. Mapping result is shown in Fig. 1. Our contributions can be concluded as follows:

1) A backend optimization method with an adaptive sliding window strategy is proposed, which adjusts the window length according to a quantitative evaluation criterion derived from the current LiDAR feature distribution to reduce unnecessary computations.
2) A multi-modal LiDAR inertial odometry and mapping framework based on ASW is proposed and open sourced (ASW-LIOM), which jointly matches feature points of mechanical and solid-state LiDAR to enhance the accuracy of odometry and the density of mapping. Additionally, the loop closure detection is incorporated into the framework to further reduce drift.
3) Experiments are conducted in both public dataset and real-world environment. The results have demonstrated that our approach improves the average accuracy by 10.3%, compared to the state-of-the-art.

II. Related Work

A. SLAM Using Mechanical and Solid-State LiDARs

LOAM [5] is proposed to extract feature points from the raw point cloud. It performs frame-to-frame matching with feature points for rough odometry, followed by frame-to-map matching for precise odometry. Subsequently, more algorithms based on LOAM have been proposed [6]–[10]. A tightly-coupled LiDAR inertial odometry algorithm by utilizing inertial measurement unit (IMU) to enhance the robustness of system is proposed in LIO-SAM [7]. However, LOAM-based SLAM algorithms with mechanical LiDARs may have drift odometry, due to inadequate features in the vertical direction, under challenging environments without large planes or long edges.

Solid-state LiDARs are used widely due to their low price and lightweight. Loam livox [11] proposes a new feature extraction method for solid-state LiDARs. On this basis, FAST-LIO [12] fuses LiDAR feature points with IMU using a tightly-coupled iterated extended Kalman filter. FAST-LIO2 [13] directly registers raw points to the map without extracting features using *ikd-Tree*. LiLi-OM [2] also presents a tightly-coupled LiDAR-inertial odometry and mapping scheme, and a modified feature extraction method for solid-state LiDARs is proposed. However, due to the small FoV of solid-state LiDARs, most of these algorithms may fail in some scenarios, such as close to a wall in doors.

To solve the problem of insufficient points and the small FoV, some multiple-LiDAR SLAM algorithms are proposed [14]–[16]. M-LOAM [14] achieves robust and simultaneous extrinsic calibration, odometry, and mapping using multiple mechanical LiDARs. While GM-Livox [15] proposes a tightly-coupled localization framework using multiple solid-state LiDARs, IMU, encoder, and GNSS measurements. However, most of them only employee the same type of LiDARs. The complementary strengths of the two types of LiDAR have not been fully utilized. MA-LIO [17] offers a filter-based system that focused on the temporal and spatial discrepancies inherent with different types of LiDARs. But its filter-based structure can lead to reduced accuracy when outlier observation comes. A tightly coupled multi-modal multi-LiDAR-inertial odometry and mapping framework with sliding window optimization is proposed in MMLOAM [1], but loop closure detection is not considered. Therefore, the accurate and efficient multi-modal LiDAR SLAM algorithm still needs further investigation.

B. Sliding Window Optimization in LiDAR SLAM

The backend optimization in SLAM can be categorized into optimization-based methods and filter-based methods. Compared to filter-based methods, optimization-based methods can acquire more accurate and robust odometry by integrating historical constraints. Sliding window optimization can remain within a bounded computation complexity and do not continue to grow over time. LiLi-OM [2] proposes a hierarchical keyframe-based sliding window optimization, which fuses the feature-map matching coefficients and the IMU constraints. MILIOM [18] uses the sliding window to construct a local map to improve the robustness of matching and adds LiDAR and IMU residuals in optimization. MM-LOAM [1] introduces a sliding window optimization that includes constraints from multi-modal LiDARs.

However, the sliding window optimization mentioned above generally set the window length to a fixed value. Poddar et al. [19] introduces an adaptive lag-length selection method for Kalman filter-based smoothers, where the length is set to vary with the system model and noise parameters. But it has not been validated in practical positioning applications. Ye et al. [20] proposes an adaptive lag smoother to implement the tightly-coupled GNSS/IMU positioning system, where the lag-length is determined by the availability of satellites in the FoV. This demonstrates the feasibility of adaptive sliding window in reducing computation time, but how to quantify the effectiveness of LiDAR features and set an appropriate window length remains to be studied.

III. Method

A. System Overview

It is assumed that the timestamp of each sensor is aligned, and the extrinsic parameters of the sensors are calibrated. The mechanical and solid-state LiDAR frame will transform to IMU frame $(\cdot)^I$ using the extrinsic parameters. The world frame is denoted as $(\cdot)^W$ and the body frame is $(\cdot)^B$. We assume the IMU frame $(\cdot)^I$ coincides with the robot body frame $(\cdot)^B$ for convenience.

An overview of the proposed method is shown in Fig. 2. First, IMU pre-integration is performed, which can be used to remove the distortion of fused raw point cloud. Then feature fusion and joint matching is conducted. The adaptive sliding window is constructed to estimate states, and loop closure

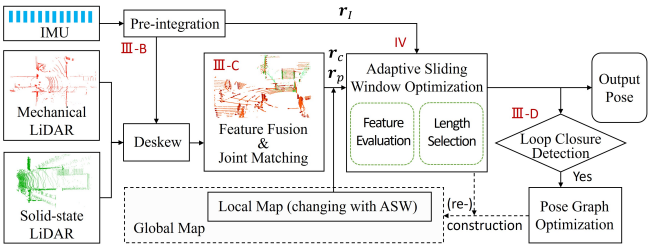


Fig. 2: Frame work of ASW-LIOM

detection is perform to correct the global pose. The optimal state $\mathbf{x}_i$ at time i can be written as:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{R}_i^T & \mathbf{p}_i^T & \mathbf{v}_i^T & \mathbf{b}_i^{aT} & \mathbf{b}_i^{gT} \end{bmatrix}^T \tag{1}$$

where $\mathbf{R}_i \in SO(3)$ is the rotation matrix, $\mathbf{p}_i \in \mathbb{R}^3$ is the position vector, $\mathbf{v}_i \in \mathbb{R}^3$ is the speed vector, $\mathbf{b}_i^a \in \mathbb{R}^3$ and $\mathbf{b}_i^g \in \mathbb{R}^3$ are the IMU bias. The sliding window contains k optimal states can be denoted as $\mathbf{X} = \begin{bmatrix} \mathbf{x}_i^T & \mathbf{x}_{i+1}^T & \dots & \mathbf{x}_{i+k-1}^T \end{bmatrix}^T$. These states can be acquired through solving the Maximum A Posteriori (MAP) optimization problem as follows:

$$\min_{\mathbf{X}} \Big\{ \|\mathbf{r}_p\|^2 + \sum_{j=i+1}^{i+k-1} \|\mathbf{r}_{I_j}\|^2 + \sum_{j=i}^{i+k-1} \|\mathbf{r}_{c_j}\|^2 + \sum_{j=i}^{i+k-1} \|\mathbf{r}_{p_j}\|^2 \tag{2}$$

where $\mathbf{r}_p$ is the prior factor marginalized by Schur-complement [21], $\mathbf{r}_I$ is the residual of IMU pre-integration, $\mathbf{r}_c$ and $\mathbf{r}_p$ are the residuals of relative LiDAR point to line and point to plane constraints respectively.

B. IMU Pre-integration and Deskew

The measurements of angular velocity $\hat{\boldsymbol{\omega}}_i$ and the acceleration $\hat{\mathbf{a}}_i$ of IMU at time i is denoted as:

$$\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i + \mathbf{b}_i^g + \mathbf{n}_i^g \tag{3}$$

$$\hat{\mathbf{a}}_i = \mathbf{R}_i^{\mathbf{BW}}(\mathbf{a}_i - \gamma) + \mathbf{b}_i^{\mathbf{a}} + \mathbf{n}_i^{\mathbf{a}} \tag{4}$$

where $\hat{\boldsymbol{\omega}}_i$ and $\hat{\mathbf{a}}_i$ are the raw IMU measurements. $\mathbf{b}_i$ and $\mathbf{n}_i$ present varying bias and white noise respectively. $\mathbf{R}_i^{BW}$ is the rotation matrix from W frame to B frame. γ is the constant gravity vector. The relative motion between time i and j can be obtained by applying the IMU pre-integration technique proposed in [22], which can be presented as:

$$\Delta\mathbf{v}_{ij} = \mathbf{R}_i^T\big(\mathbf{v}_j - \mathbf{v}_i - \gamma\Delta\mathbf{t}_{ij}\big) \tag{5}$$

$$\Delta\mathbf{p}_{ij} = \mathbf{R}_i^T\big(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta\mathbf{t}_{ij} - \frac{1}{2}\gamma\Delta\mathbf{t}_{ij}^2\big) \tag{6}$$

$$\Delta\mathbf{R}_{ij} = \mathbf{R}_i^T\mathbf{R}_j \tag{7}$$

where $\mathbf{t}_{ij}$ is the interval between time i and j. $\Delta\hat{\mathbf{v}}_{ij}$, $\Delta\hat{\mathbf{p}}_{ij}$ and $\Delta\hat{\mathbf{R}}_{ij}$ can be acquired by directly integrate the raw IMU measurements, then the residual of IMU pre-integration $\mathbf{r}_I$ can be constructed. For each LiDAR point $\mathfrak{p}_t$, $t \in [i, j]$, pre-integration measurements can be utilized to project it to B frame at time i through linear interpolation, to reduce the motion distortion of the raw LiDAR points, thus completing the deskew process.

C. Feature Extraction and Joint Matching

Inspired by LOAM [5] and Loam livox [11], the feature points are extracted from measurements according to their curvatures. The edge features are selected with high curvature and the plane features are with low curvature. The LiDAR scan from mechanical and solid-state LiDAR at time i are denoted as P_i^M and P_i^S. For mechanical LiDAR, the plane and edge feature points sets are extracted from P_i^M, denoted as $\mathcal{P}_i^M$ and $\mathcal{E}_i^M$ respectively. Similarly, the feature points sets extracted from P_i^S are denoted as $\mathcal{P}_i^S$ and $\mathcal{E}_i^S$. The feature points sets from two types LiDAR is fused, denoted as $\mathcal{P}_i$ and $\mathcal{E}_i$, where $\mathcal{P}_i = \{\mathcal{P}_i^M, \mathcal{P}_i^S\}$, and $\mathcal{E}_i = \{\mathcal{E}_i^M, \mathcal{E}_i^S\}$. Point supplementation for the vertical angular resolution of mechanical LiDAR provided by solid-state LiDAR is shown in Fig. 3.

A local map with feature points is adaptively constructed as the target of joint matching for $\mathcal{P}_i$ and $\mathcal{E}_i$ using corresponding global poses, denoted as $\mathcal{P}_{local} = \{\mathcal{P}_{i-l}, \dots, \mathcal{P}_{i-1}\}$, $\mathcal{E}_{local} = \{\mathcal{E}_{i-l}, \dots, \mathcal{E}_{i-1}\}$, where l is set according to the length of adaptive sliding window. Since feature points from multi-modal LiDAR are fused, both global and local maps can present more geometric information, so that it is easier and more accurate to find correspondences of plane and edge. For each plane point at the current window, the three closest points from $\mathcal{P}_{local}$ are selected as corresponding plane points. The corresponding plane points may come from different types of LiDAR, as shown in Fig. 4, which makes the matched plane potentially closer to the target point. Eventually, the residual of point to line can be calculated by:

$$\mathbf{r}_{c_{(i,\alpha)}} = \frac{\big|(\mathfrak{p}_{i,\alpha}^W - \mathfrak{p}_v^W) \times (\mathfrak{p}_{i,\alpha}^W - \mathfrak{p}_w^W)\big|}{\big|\mathfrak{p}_v^W - \mathfrak{p}_w^W\big|} \tag{8}$$

where $\mathfrak{p}_{i,\alpha}^W \in \mathcal{E}_i$, and $\mathfrak{p}_v^W, \mathfrak{p}_w^W \in \mathcal{E}_{local}$ are two nearest points of $\mathfrak{p}_{i,\alpha}^W$. The residual of point to plane can be calculated by:

$$\mathbf{r}_{p_{(i,\beta)}} = \frac{\big|(\mathfrak{p}_{i,\beta}^W - \mathfrak{p}_u^W) \cdot \vec{n}\big|}{\big|\vec{n}\big|} \tag{9}$$

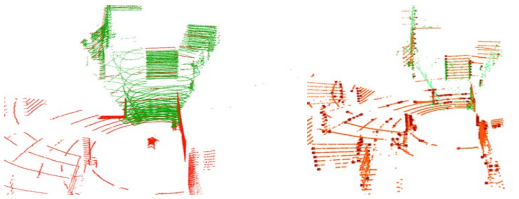
$$\vec{n} = (\mathfrak{p}_u^W - \mathfrak{p}_v^W) \times (\mathfrak{p}_u^W - \mathfrak{p}_w^W) \tag{10}$$

where $\mathfrak{p}_{i,\beta}^W \in \mathcal{P}_k$, and $\mathfrak{p}_u^W, \mathfrak{p}_v^W, \mathfrak{p}_w^W \in \mathcal{P}_{local}$ are three nearest points of $\mathfrak{p}_{i,\beta}^W$.

D. Loop Closure Detection

We save state $\mathbf{T}_i = \{\mathbf{R}_i|\mathbf{p}_i\}$ and the feature point cloud $\mathcal{P}_i$, $\mathcal{E}_i$ corresponding to the state in the global map, once ASW-LIOM outputs an optimal pose, and add the state into the pose graph as illustrated in Fig. 5. When the new state $\mathbf{T}_{i+1}$ is solved, we first search the pose graph and find the previous state that is closet to $\mathbf{T}_{i+1}$ in euclidean space. Notablely, the closet state may not be $\mathbf{T}_i$ if loop closure is detected. Suppose we find the closet state at previous time j, denoted as $\mathbf{T}_j$, then the relative transformation $\Delta\mathbf{T}_{j,i+1}$ between $\mathbf{T}_j$ and $\mathbf{T}_{i+1}$ can be computed as:

$$\Delta\mathbf{T}_{j,i+1} = \mathbf{T}_j^T\mathbf{T}_{i+1} \tag{11}$$



(a) Raw points (b) Feature points

Fig. 3: The left of the figure is the raw points, and the green and red point clouds represent data obtained from solid-state LiDAR and mechanical LiDAR. The right is the feature points extracted from the raw points.

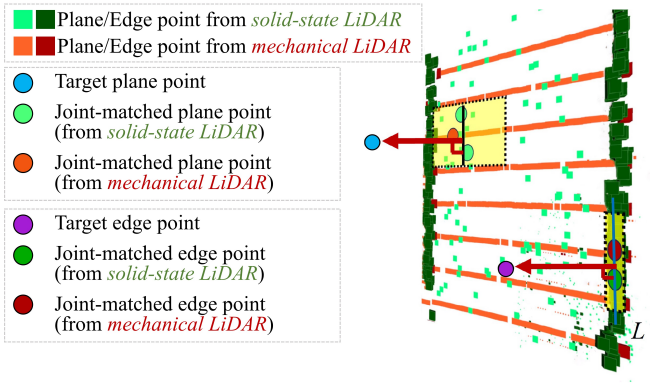


Fig. 4: Joint matching diagram. The green and red points represent feature points from solid-state LiDAR ahd mechanical LiDAR. The yellow box presents corresponding plane and edge.

A sub-map $\mathbb{F}_{sub} = \{\mathbb{F}_{j-l}, \ldots, \mathbb{F}_{j+l}\}$ is constructed using $\{\mathbf{T}_{j-l}, \ldots, \mathbf{T}_{j+l}\}$, where $\mathbb{F}_j = \{\mathcal{P}_i, \mathcal{E}_i\}$, $l = \frac{k}{2}$, k is the length of sliding window. The scan match between $\mathbb{F}_{i+1}$ and $\mathbb{F}_{sub}$ is performed to obtain the observation measurement $\Delta\hat{\mathbf{T}}_{j,i+1}$. The error between $\Delta\mathbf{T}_{j,i+1}$ and $\Delta\hat{\mathbf{T}}_{j,i+1}$ should be zero. Eventually, the loop closure constraint is added to the pose graph optimization, and all states will be updated after the pose graph optimization.

IV. Adaptive Sliding Window

A. The Quantitative Evaluation Method of Constraints Derived from LiDAR Features

Since our matching optimization is mostly based on LiDAR features, the geometric distribution of the feature points will affect the accuracy. In our previous work [23], a quantitative evaluation method of constraints derived from LiDAR features is introduced. According to equation (2), the cost function of state $\mathbf{x}_k$ at time k can be expressed as a quadratic in the form of matrix:

$$\mathbf{f}(\mathbf{x}_k) = \mathbf{r}(\mathbf{x}_k)^T \Sigma^{-1} \mathbf{r}(\mathbf{x}_k) \tag{12}$$

The residual of IMU and the residual of LiDAR are independent, so the effects of IMU and LiDAR on the state $\mathbf{x}_k$ can be discussed separately. The residual function $\mathbf{r}(\mathbf{x}_k)$

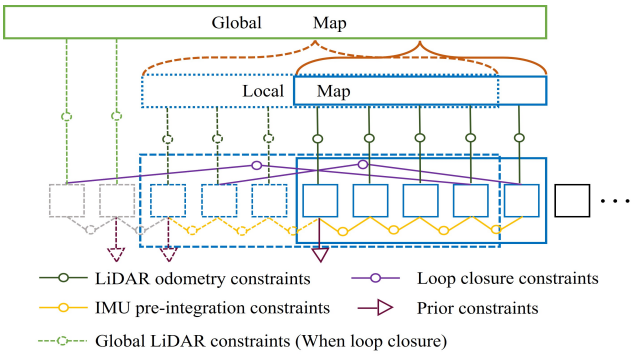


Fig. 5: Adaptive sliding window optimization. This factor graph presents the length of sliding window changes from 7 to 5.

contains the residuals of relative LiDAR point to line $\mathbf{r}_c$ and point to plane $\mathbf{r}_p$. Denoted the position of state $\mathbf{x}_k$ as $\mathbf{p}_k$, where $\mathbf{p}_k = [\mathbf{p}_k^x \quad \mathbf{p}_k^y \quad \mathbf{p}_k^z]$, the iterative step $\Delta\mathbf{p}_k$ can be computed as:

$$\Delta\mathbf{p}_k = \left(\mathbf{J}^T\Sigma^{-1}\mathbf{J}\right)^{-1}\mathbf{J}^T\Sigma^{-1}\mathbf{r}(\mathbf{x}_k) \tag{13}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{r}(\mathbf{x_k})$ with respect to $\mathbf{p}_k$, and Σ presents the weight associated with sensor noise model and is often simplified as an identify matrix. In particular,

$$\mathbf{J} = \frac{\partial \mathbf{r}(\mathbf{x_k})}{\partial \mathbf{p_k}} = \begin{bmatrix} \frac{\partial \mathbf{r}_1}{\partial \mathbf{p}_k} \\ \frac{\partial \mathbf{r}_2}{\partial \mathbf{p}_k} \\ \vdots \\ \frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{p}_k} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{r}_1}{\partial \mathbf{p}_k^x} & \frac{\partial \mathbf{r}_1}{\partial \mathbf{p}_k^y} & \frac{\partial \mathbf{r}_1}{\partial \mathbf{p}_k^z} \\ \frac{\partial \mathbf{r}_2}{\partial \mathbf{p}_k^x} & \frac{\partial \mathbf{r}_2}{\partial \mathbf{p}_k^y} & \frac{\partial \mathbf{r}_2}{\partial \mathbf{p}_k^z} \\ \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{p}_k^x} & \frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{p}_k^y} & \frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{p}_k^z} \end{bmatrix} \tag{14}$$

where $\mathbf{r}_i$ presents the residual of point to line or to plane. For a feature point $\mathfrak{p}_i$ at the current time k, it will be mapped to the local map through the optimized pose, which we denote as $\mathfrak{p}_i^{'}$. The transformation form $\mathfrak{p}_i^{'}$ to $\mathfrak{p}_i$ can be expressed as:

$$\mathfrak{p}_i^{'} = \mathbf{R}_k\mathfrak{p}_i + \mathbf{p}_k \tag{15}$$

then, we have

$$\mathfrak{p}_i = \mathbf{R}_k^{-1}(\mathfrak{p}_i^{'} - \mathbf{p}_k) \tag{16}$$

$$\begin{aligned} \frac{\partial \mathbf{r}_i}{\partial \mathbf{p_k}} &= \frac{\partial \mathbf{r}_i}{\partial \mathfrak{p}_i} \cdot \frac{\partial \mathfrak{p}_i}{\partial \mathbf{p_k}} \\ &= \vec{n}_i \cdot \frac{\partial(\mathbf{R}_k^{-1}(\mathfrak{p}_i^{'} - \mathbf{p}_k))}{\partial[\mathbf{p}_k^x \quad \mathbf{p}_k^y \quad \mathbf{p}_k^z]} \\ &= \vec{n}_i \cdot \mathbf{R}_k^{-1} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \end{aligned} \tag{17}$$

where $\vec{n}$ is the normal vector of the line or plane.

At the result of the iterative optimization, the error covariance matrix for $\Delta\mathbf{p}_k$ can be computed as:

$$\mathbf{cov}(\Delta\mathbf{p}_k) = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\sigma_L^2 \tag{18}$$

According to equation (14), the matrix $\mathbf{J}^T\mathbf{J}$ is a 3×3 symmetric matrix and depends on the relative geometry of

the LiDAR feature. It is assumed that the components of $\mathbf{r}(\mathbf{x}_k)$ are independent and identically distributed. Thus, the covariance of $\mathbf{r}(\mathbf{x}_k)$ is defined as a scalar multiple σ_L^2 of the identify. The matrix $\left(\mathbf{J}^T\mathbf{J}\right)^{-1}$ quantifies how the residual errors of the LiDAR are translated into components of the covariance of $\Delta\mathbf{p}_k$. Then we design the evaluation function $\mathbf{E}(\cdot)$ formulated as:

$$\mathbf{E}(\mathbf{r}) = \sqrt{tr\left[\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\right]} \tag{19}$$

where $tr(\cdot)$ is the trace of the matrix. Significantly, a necessary condition of the evaluation is that the noise associated with ranging measurements is independent of the distance. If the value of $\mathbf{E}(\mathbf{r})$ is small, it means that there are adequate features in the current sliding window. Since the Ceres solver [24] is used to solve the optimization problem, the Jacobian matrix of $\mathbf{r}(\mathbf{x}_k)$ with respect to $\mathbf{p}$ can be acquired directly.

B. The Strategy of Changing The Length of Window

As Fig. 5 and equation (2) describe, there are prior constraints, IMU constraints and LiDAR odometry constraints in the sliding window optimization. In this part, the strategy of changing the length of window based on these constraints and the value of equation (19) is discussed. If we do not want the length of window to change frequently, the average value of $\mathbf{E}(\mathbf{r})$ can be calculated, which is denoted as $\bar{\mathbf{E}}(\mathbf{r})$. The longest and shortest window length are denoted as h and l respectively. There are three situations; one situation is that the value of $\bar{\mathbf{E}}(\mathbf{r})$ is higher than a threshold max, which means that there are no adequate feature points in the current window, the length of the window can be increased to add more constraints to get a better optimization result; thus set the length of the sliding window as h. The second situation is that the value of $\bar{\mathbf{E}}(\mathbf{r})$ is smaller than a threshold min, which means that there are adequate feature points, the length of the current sliding window can be cut down to reduce the unnecessary computation. Since the similar optimization result can be acquired with the shorten length of sliding window, we set the length of sliding window as l. The last one situation is that the value of $\bar{\mathbf{E}}(\mathbf{r})$ is between the hyperparameter min and max. We designed a linear mapping functions based on the variable $\bar{\mathbf{E}}(\mathbf{r})$ to calculate the sliding window length, which is formulated as:

$$k^* = \begin{cases} l, & \text{if } \bar{\mathbf{E}}(\mathbf{r}) < min \\ l + \lfloor \dfrac{\bar{\mathbf{E}}(\mathbf{r}) - min}{max - min}(h-l) \rfloor, & \text{if } min < \bar{\mathbf{E}}(\mathbf{r}) < max \\ h, & \text{if } \bar{\mathbf{E}}(\mathbf{r}) > max \end{cases} \tag{20}$$

We also developed a logarithmic mapping function to compare, which is formulated as:

$$k^* = \begin{cases} l, & \text{if } \bar{\mathbf{E}}(\mathbf{r}) < min \\ \lfloor a \cdot e^{\bar{\mathbf{E}}(\mathbf{r})} + c \rfloor, & \text{if } min < \bar{\mathbf{E}}(\mathbf{r}) < max \\ h, & \text{if } \bar{\mathbf{E}}(\mathbf{r}) > max \end{cases} \tag{21}$$

where $a = \frac{h-l}{e^{max} - e^{min}}$, $c = l - \left(\frac{h-l}{e^{max} - e^{min}}\right) \cdot e^{min}$, $\lfloor \cdot \rfloor$ denotes the floor operation. The details of the algorithm are described in Algorithm 1. Fig. 5 shows a possible scenario that the length of sliding window changes from 7 to 5 after performing the adaptive sliding window algorithm.

Algorithm 1 Adaptive Sliding Window Algorithm

Input: The current length of window k_0, feature points frames, hyperparameters min and max

Output: The new length of sliding window k^*, optimal states $\mathbf{X} = \begin{bmatrix}\mathbf{x}_i^T & \mathbf{x}_{i+1}^T & \dots & \mathbf{x}_{i+k_0-1}^T\end{bmatrix}^T$

```
Initialize the optimization problem 𝕃
Construct the local feature points map P_local and E_local
for each j ∈ [i, i + k_0 − 1] do
    Add point-to-line constraint r_{c_j} to 𝕃
    Add point-to-plane constraint r_{p_j} to 𝕃
    if j > i then
        Add IMU constraint r_{I_j} to 𝕃
    end if
end for
Add prior constraint r_p to 𝕃
Conduct the sliding window optimization, and get the k_0 optimal states X = [x_i^T  x_{i+1}^T  ...  x_{i+k_0-1}^T]^T
Compute the Jacobian matrix of J respect to p_{i+k_0-1}
Calculate the value of Ē(r) and k*
if k* <= k_0 then
    Compute the number of states that require marginalization m = k_0 − k* + 1
    Marginalize the m states using Schur-complement, and update prior constraint r_p
else
    reset prior constraint r_p
end if
return k* and X = [x_i^T  x_{i+1}^T  ...  x_{i+k_0-1}^T]^T
```

V. Experiments

A. Dateset and Implementation Details

The public datasets used in experiments are TIERS LiDARs dataset [25] and Botanic Garden Dataset [26]. For TIERS LiDARs dataset, the Velodyne VLP-16, Livox Horizon and the IMU provided by Livox Horizon is used in our experiments. Botanic Garden Dataset is a high-quality robot navigation dataset, which is collected in a luxuriant botanic garden. The Velodyne VLP-16, Livox AVIA and the IMU provided by Livox AVIA is used. Notablely, the sensor timestamp has been synchronized in both datasets. For the forest01, indoor01 and indoor02 sequences in TIRES LiDARs dataset, the LIVOX-ROS-Driver provided by the factory does not use the ros system timestamp. Although the messages are recorded at the same time, the timestamp formats of the Livox LiDAR and Velodyne LiDAR are different. So there is a time offset between Livox LiDAR and Velodyne LiDAR. When receiving messages in the code, the time offset was calculated and added to each frame of Livox message. This will convert the timestamp format of Livox lidar to be consistent with Velodyne LiDAR. The specifications of dataset is shown in Table.I.

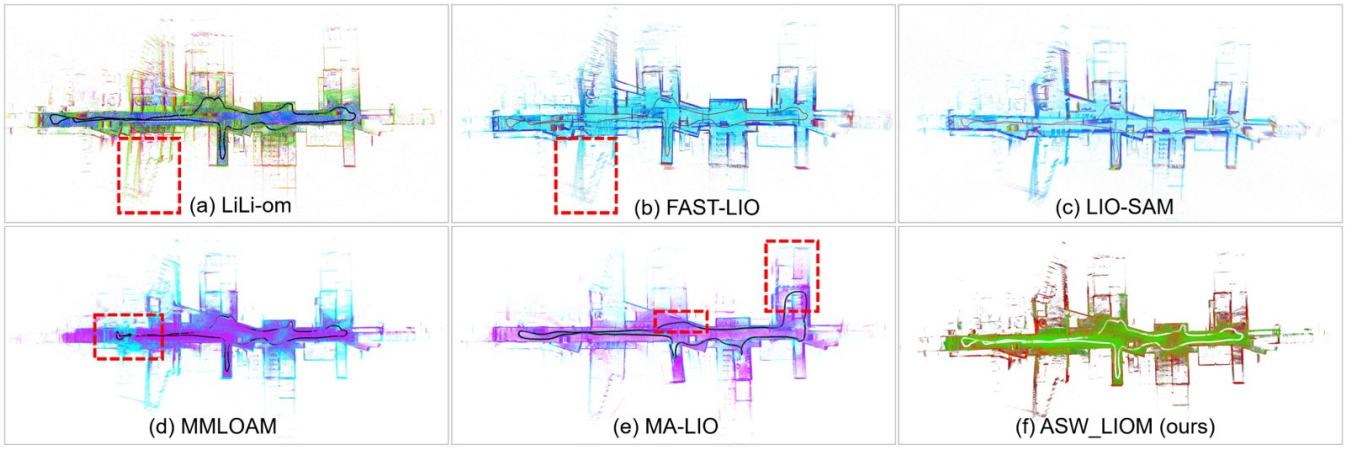


Fig. 6: Visualization results of our method and comparative methods on TIRES dataset sequence indoor11

TABLE I: SPECIFICATIONS OF DATASETS

Dataset	LiDAR Information	IMU Information	Type
TIRES	Velodyne VLP-16 Livox Horizon	Livox built-in	Indoor
Botanic Garden	Velodyne VLP-16 Livox AVIA	Livox built-in	Outdoor
Ours	RoboSense Ruby Lite Livox Horizon	Livox built-in	Outdoor

Our algorithm is implemented in C++ and executed in Ubuntu18.04 using the robot operating system(ROS) [27] and deployed on a 16GB RAM desktop computer with an Intel(R) Core(TM) i7-14700HX CPU. The nonlinear optimization problem in equation (2) is solved using the Ceres Solver [24]. The hyperparameters are $min = 0.001$, $max = 0.1$ based on our experimental experience. The evo odometry evaluation tool [28] is used to align trajectories and conduct comparisons based on the root mean square error (RMSE) of the absolute trajectory error (ATE) metric. The RMSE of ATE is defined by:

$$\text{RMSE(ATE)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{T}_{G_i}^{-1}\hat{\mathbf{T}}_i - \mathbf{I}_{4\times4}\|_F^2} \tag{22}$$

Where $\mathbf{T}_{G_i}$ and $\hat{\mathbf{T}}_i$ denote the pose of ground truth and the estimated pose at time i, respectively.

B. Evaluation on Public Dataset

In order to evaluate the performance of the proposed algorithm, we compared our method with the state-of-the-art LiDAR SLAM algorithms, including LiLi-om [2], FAST-LIO [12], LIO-SAM [7], MA-LIO [17] and MMLOAM [1]. We note that LiLi-om, LIO-SAM, and FAST-LIO are designed for single LiDAR configuration, so we conducted two experiments for each algorithm: one using a mechanical LiDAR and the other using a solid-state LiDAR. It is clearly shown in Table.II that our system consistently achieves better performance compared to existing methods on most sequences. Fig. 6. shows the map of different algorithms on indoor11, the comparison algorithms have shown some drift.

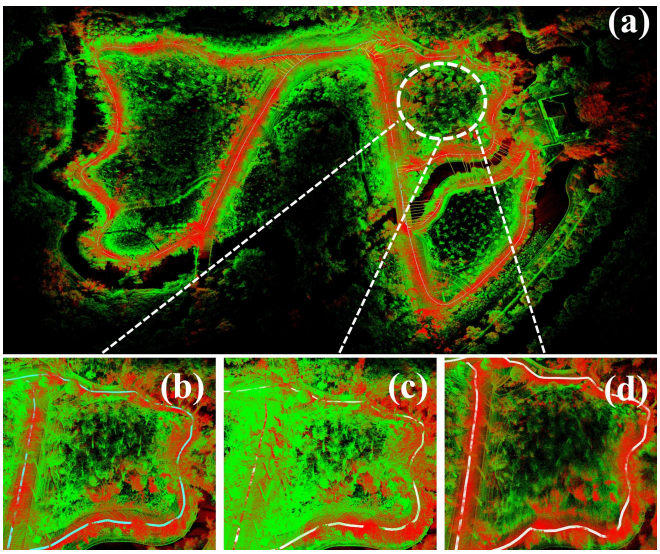


Fig. 7: The comparison of different multiple-LiDAR methods. (a) Mapping results on Botanic Garden dataset sequence 1006-03 of our method. (b) Details of our method. (c) Details of MMLOAM. (d) Details of MA-LIO.

Compared to the algorithms only using mechanical LiDAR, our method performs better by combining the advantages of two types of LiDARs. For indoor02 sequence, though MMLOAM achieves the best performance, the result of our method is very close to the best result, since the scenario of dataset indoor02 is a room, which can provide adequate feature points continuously. Our method will maintain the minimum window length at this scenario. For indoor01 and indoor02 sequences, the algorithms only using solid-state LiDAR failed. The main reason is that the FoV of solid-state LiDAR is too small to get enough effective feature points. Compared to the second best of result, our method improve the average accuracy by 10.3%. The comparison of different multiple-LiDAR methods on sequence 1006-3 of Botanic Garden are shown in Fig. 7. The details of our method is more clear than other methods.

TABLE II: ATE(unit:m) OF DIFFERENT METHODS ON SEQUENCES

Dataset	Sequence	Distance	LiLi-om(S/M)	FAST-LIO(S/M)	LIO-SAM(M)	MMLOAM	MA-LIO	Ours_no_loop	Ours
TIERS	forest01	21.46m	0.231/0.159	0.102/0.128	0.182	0.094	0.083	0.095	**0.081**
	indoor01	33.89m	F/0.507	F/0.429	0.403	0.418	0.431	0.339	**0.320**
	indoor02	70.58m	F/0.453	F/0.403	0.515	**0.303**	0.580	0.317	0.304
	indoor09	174.52m	0.762/0.568	0.473/0.925	1.029	0.591	0.575	0.461	**0.417**
	indoor11	271.33m	2.954/1.368	4.442/1.535	0.632	0.910	0.627	0.545	**0.541**
Botanic Garden	1005-05	566.84m	3.512/3.418	3.253/2.904	2.985	2.612	2.814	2.630	**2.458**
	1006-03	686.23m	5.437/4.845	5.645/4.215	3.841	3.460	3.791	3.421	**3.174**
	1008-01	855.92m	5.267/4.225	5.482/4.627	4.348	3.689	3.890	3.673	**3.340**

[1] S presents solid-state LiDAR, M presents mechanical LiDAR.
[2] The best results are presented in **bold** and the second-best are in underline.
[3] F denotes that the method has failed.

TABLE III: ATE (unit:m) AND RUN TIME (unit:s) OF DIFFERENT SLIDING WINDOW (SW) STRATEGIES

Metric	SW strategies	Window length	indoor09	indoor11	1005-05	1006-03	1008-01
ATE/Run Time	Fixed Length	5	0.472/0.074	0.550/0.084	2.546/0.064	3.401/0.064	3.451/0.075
		10	0.428/0.122	0.544/0.138	2.487/0.145	3.185/0.134	3.401/0.141
		15	**0.415**/0.181	0.538/0.194	**2.454**/0.184	3.171/0.195	3.338/0.188
	Random Length	5-15	0.580/0.089	0.550/0.952	2.845/0.074	3.650/0.078	4.012/0.081
		5-30	0.472/0.114	0.780/0.095	3.024/0.068	3.847/0.069	4.129/0.095
	Adaptive Length	5-15 (Linear)	0.419/0.118	0.544/0.124	2.466/0.109	3.181/0.119	3.348/0.138
		5-15 (Log)	0.417/0.120	0.541/0.131	2.458/0.115	3.174/0.124	3.340/0.141
		5-30 (Linear)	0.416/0.125	0.539/0.146	2.457/0.121	3.171/0.154	3.338/0.170
		5-30 (Log)	**0.415**/0.128	**0.536**/0.152	2.457/0.137	**3.169**/0.165	**3.336**/0.178

[1] The best results are presented in **bold** and the second-best are in underline.

TABLE IV: AVERAGE RUN TIME OF DIFFERENT MODULES

Dataset	Module	Time(ms)
Botanic Garden	Data preprocess	6.73
	Feature extraction	24.03
	Length determination	0.052
	ASW optimization	96.43

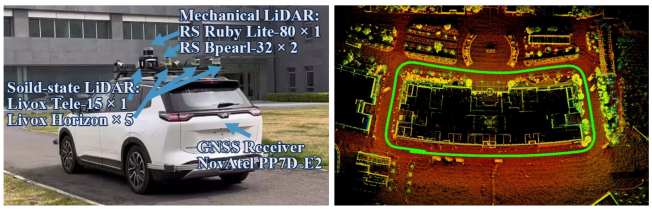
We also conduct an ablation study to assess the effectiveness of loop closure detection. As illustrates in Table.II, the accuracy has decreased without loop closure detection. Since the distance of Botanic Garden is longer than TIRES, and there more loop closure scenes, the accuracy improvement is more significant by adding loop closure module.

C. Evaluation for Adaptive Sliding Window

To evaluate the effectiveness of adaptive sliding window in backend optimization, the experiments are conducted which use different sliding window strategies, including the fixed length of sliding window, random length of sliding window and our proposed adaptive sliding window strategy with linear and logarithmic mapping function. In addition to calculating RMSE, we also recorded the time spent on backend optimization. The results are shown in Table.III. For the strategy of fixed length, it can be observed that the accuracy improves as the window length increases. Correspondingly, the required time also increases. Compared to the maximum fixed window length, the adaptive window length can achieve the closed or even higher accuracy in less time. For the random length, the accuracy does not follow a consistent trend. Since the length is changed randomly, it may shorten the window length in areas with fewer feature points, where actually needs a longer length. Drifts will be introduced in this case. Therefore, the performance of random window length also has randomness. Under the same setting of length, we can see that logarithmic mapping function can achieve higher accuracy than linear mapping function with less increase in time consumption. Because the window length variation of logarithmic mapping is not as large as that of linear mapping. Although setting the window length between 5-30 can achieve higher accuracy, it also comes with additional costs. Taking sequence 1008-01 as an example, when using logarithmic mapping, the accuracy of window length ranging from 5-30 increased by 1.2% compared to window length ranging from 5-15, but the time consumption increased by 26%. Therefore, considering the limited computing resources, we used a window length range of 5-15 with logarithmic mapping function when conducting other experiments. We also conduct an experiment to analyze run time of different modules. The result is shown in Table.IV. Compared to other modules, the time spent by length determination is almost negligible, which means applying adaptive sliding window algorithm does not bring additional computation cost.

D. Self-Collected Dataset Experiments

We have conducted experiments in a real-world environment, as depicted in Fig. 8. The experimental platform is a sport-utility vehicle equipped with three mechanical LiDARs (one RS Ruby Lite-80 and two RS Bpearl-32), six solid-state LiDARs (one Livox Tele-15 and five Livox Horizon) and a GNSS receiver (NovAtel PP7D-E2). The Livox Horizon and the RoboSense Ruby Lite are used in our experiments. The experiment was carried out with the vehicle that completed a full lap around a building, covering a total distance of



(a) Experimental platform (b) Mapping result in real world

Fig. 8: Mapping result of ASW-LIOM in real-world environment

340.225 meters. To achieve precise temporal synchronization of the multi-modal sensor system, we employ a Pulse Per Second (PPS) hard-triggering mechanism. The PPS signal that distributed to all sensors via a dedicated synchronization hub, is obtained from the GNSS receiver NovAtel PP7D-E2. Upon receiving the rising edge of the PPS pulse, each sensor initiates its data acquisition process. To quantify localization accuracy, the ground truth trajectory is derived from post-processed results of the NovAtel PP7D-E2. The average error of our method is 0.4%. This result demonstrates the effectiveness of our method in real-world scenarios, with the error remaining at a level suitable for practical applications in autonomous driving tasks.

VI. CONCLUSIONS

In this work, we propose an adaptive sliding window strategy that dynamically adjusts the window length based on the distribution of LiDAR features, which can reduce computational cost while preserving the expected accuracy. To exploit different scanning patterns for improved feature distribution at each time step, we present a multi-modal LiDAR inertial odometry and mapping framework that combines mechanical and solid-state LiDARs. Furthermore, Loop closure detection is added to reduce cumulative errors. The results of experiments on public and self-collected datasets confirm the accuracy of our approach. For future work, we plan to add more sensors, like camera sensor, to achieve more robust and semantically enriched mapping outcomes.

REFERENCES

[1] L. Qingqing, Y. Xianjia, J. P. Queralta, and T. Westerlund, "Robust multi-modal multi-lidar-inertial odometry and mapping for indoor environments," *arXiv preprint arXiv:2303.02684*, 2023.

[2] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.

[3] Z. Wang, X. Liu, L. Yang, and F. Gao, "Sw-lio: A sliding window based tightly coupled lidar-inertial odometry," *IEEE Robotics and Automation Letters*, 2023.

[4] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3144–3150.

[5] J. Zhang, S. Singh, *et al.*, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[6] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[7] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[8] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam: Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4390–4396.

[9] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "Mulls: Versatile lidar slam via multi-metric linear least square," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 633–11 640.

[10] Z. Wang, L. Yang, F. Gao, and L. Wang, "Fevo-loam: Feature extraction and vertical optimized lidar odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 086–12 093, 2022.

[11] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 3126–3131.

[12] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[13] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[14] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust odometry and mapping for multi-lidar systems with online extrinsic calibration," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 351–371, 2021.

[15] 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 Journal*, vol. 22, no. 4, pp. 3626–3636, 2022.

[16] S. Das, N. Mahabadi, M. Fallon, and S. Chatterjee, "M-lio: Multi-lidar, multi-imu odometry with sensor dropout tolerance," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–7.

[17] M. Jung, S. Jung, and A. Kim, "Asynchronous multiple lidar-inertial odometry using point-wise inter-lidar uncertainty propagation," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4211–4218, 2023.

[18] T.-M. Nguyen, S. Yuan, M. Cao, L. Yang, T. H. Nguyen, and L. Xie, "Miliom: Tightly coupled multi-input lidar-inertia odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5573–5580, 2021.

[19] S. Poddar and J. L. Crassidis, "Adaptive lag smoother for state estimation," *Sensors*, vol. 22, no. 14, p. 5310, 2022.

[20] C. Ye, W. Li, and Y. Hu, "A tightly-coupled gnss rtk/ins positioning algorithm based on adaptive lag smoother," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–8.

[21] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of field robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[22] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[23] W. Li, Y. Hu, Y. Han, and X. Li, "Kfs-lio: Key-feature selection for lightweight lidar inertial odometry," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5042–5048.

[24] S. Agarwal, K. Mierle, and T. C. S. Team. (2023, 10) Ceres Solver. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[25] L. Qingqing, Y. Xianjia, J. P. Queralta, and T. Westerlund, "Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3837–3844.

[26] Y. Liu, Y. Fu, M. Qin, Y. Xu, B. Xu, F. Chen, B. Goossens, P. Z. Sun, H. Yu, C. Liu, L. Chen, W. Tao, and H. Zhao, "Botanicgarden: A high-quality dataset for robot navigation in unstructured natural environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2798–2805, 2024.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[28] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.