2DLIW-SLAM:2D LiDAR-Inertial-Wheel Odometry with Real-Time Loop Closure

Bin Zhang[1,*] Zexin Peng[1,*] Bi Zeng[1,†] and Junjie Lu[1] ‡ §

[1]Guangdong University of Technology, Guangzhou, Guangdong, People's Republic of China

E-mail: `zomb2000@163.com`

E-mail: `zb9215@gdut.edu.cn`

April 2024

Abstract.   Due to budgetary constraints, indoor navigation typically employs 2D LiDAR rather than 3D LiDAR. However, the utilization of 2D LiDAR in Simultaneous Localization And Mapping (SLAM) frequently encounters challenges related to motion degeneracy, particularly in geometrically similar environments. To address this problem, this paper proposes a robust, accurate, and multi-sensor-fused 2D LiDAR SLAM system specifically designed for indoor mobile robots. To commence, the original LiDAR data undergoes meticulous processing through point and line extraction. Leveraging the distinctive characteristics of indoor environments, line-line constraints are established to complement other sensor data effectively, thereby augmenting the overall robustness and precision of the system. Concurrently, a tightly-coupled front-end is created, integrating data from the 2D LiDAR, IMU, and wheel odometry, thus enabling real-time state estimation. Building upon this solid foundation, a novel global feature point matching-based loop closure detection algorithm is proposed. This algorithm proves highly effective in mitigating front-end accumulated errors and ultimately constructs a globally consistent map. The experimental results indicate that our system fully meets real-time requirements. When compared to Cartographer, our system not only exhibits lower trajectory errors but also demonstrates stronger robustness, particularly in degeneracy problem.

Keywords: SLAM,2D LiDAR, IMU, wheel odometry, tightly coupled odometry, loop detection, optimization.

Submitted to:

---

‡ * Both authors contributed equally to the paper.

§ † Corresponding author.

1. Introduction

SLAM plays a crucial role in facilitating autonomous localization and map construction for robots in unknown environments, thereby providing substantial support for robot autonomy and navigation [1]. As SLAM continues to advance, its real-life applications have garnered increasing interest for enhancing the quality of life and fostering a smarter society. Particularly in indoor environments, where GPS signals often prove unreliable for robot localization, SLAM technology proves highly suitable for efficient positioning.

At present, mainstream indoor mobile robots predominantly rely on 2D LiDAR for localization and mapping, such as floor-cleaning robots and restaurant service robots. However, a limitation of these robots lies in their loose coupling with other sensors. For instance, Cartographer [2] fails to fully leverage the potential performance of each sensor, potentially resulting in motion degradation when the environmental geometric structure is relatively simple. Additionally, due to sensor constraints, 2D LiDAR SLAM typically assumes strict horizontal ground surfaces during modeling, effectively restricting the robot's motion to three degrees of freedom (3DoF). Although this assumption holds true in many indoor scenarios, it may not be applicable in complex environments.

In scenarios where six degrees of freedom (6DoF) robotic motion is applicable, sensor combinations involving 3D LiDAR and an Inertial Measurement Unit (IMU), such as FAST-LIO [3], LIO-SAM [4], have shown promise. While 3D LiDAR can provide 3D point cloud data, leading to higher accuracy in the optimization process, this comes at the cost of consuming more computational resources and requiring more expensive hardware compared to 2D LiDAR, as also mentioned in BLVI-SLAM [5]. As a result, many indoor robots are equipped solely with 2D LiDAR, which provides only 2D scanning data. Our aim is to extract geometric structures, such as corner points and straight lines, from 2D scan data for tracking and loop detection, similar to 3D LiDAR front-end processing. Additionally, the motion of most indoor mobile robots typically exhibits approximately 3DoF, with only minor variations occurring in a few axes, such as roll and pitch. This suggests the possibility of estimating 6DoF using a 2D LiDAR,

making it a viable primary sensor choice.

In recent times, propelled by advancements in Visual SLAM [6] and 3D LiDAR SLAM [7], multi-sensor fusion for motion and pose estimation has found extensive utility. Given that indoor robots generally move at slow speeds with frequent stops and starts, the performance of IMU under such conditions may not be ideal, whereas wheel odometry demonstrates superior performance, especially when the robot is stationary and there is no error accumulation. Moreover, as most indoor robots are wheeled robots, their motion state can be inferred to some extent through encoder information. However, the fusion of 2D LiDAR and wheel odometry remains relatively underexplored in this domain.

Considering these factors, relying solely on a combination of 2D LiDAR and IMU might not offer sufficient robustness, particularly in scenarios featuring significant dynamic interference and degraded environments. Such conditions can lead to erroneous motion state estimates, potentially resulting in system crashes. Thus, a more reasonable approach involves augmenting the sensor combination by integrating the robot's built-in wheel odometry, representing a cost-effective solution. Hence, the primary objective of this paper is to design a fast, stable, and cost-effective 2D LiDAR-inertial-wheel odometry. Furthermore, our proposed system incorporates real-time loop detection and pose graph optimization to create a comprehensive SLAM system, termed as 2DLIW-SLAM.

The primary contributions of this paper are as follows:

- Development of a novel point-line feature extraction method tailored for 2D LiDAR data, along with the construction of a corresponding relative pose transformation observation model. Upon that, we present a multi-sensor tightly-coupled odometry that seamlessly fuses data from 2D LiDAR, IMU, and wheel odometry.

- Introduction of a global feature-based loop detection algorithm that achieves high-precision matching while meeting real-time requirements.

- We incorporate additional constraints, such as ground constraints, and integrate pivotal components such as initialization, graph optimization, and probability grid map, among others, thus assembling a complete SLAM system.
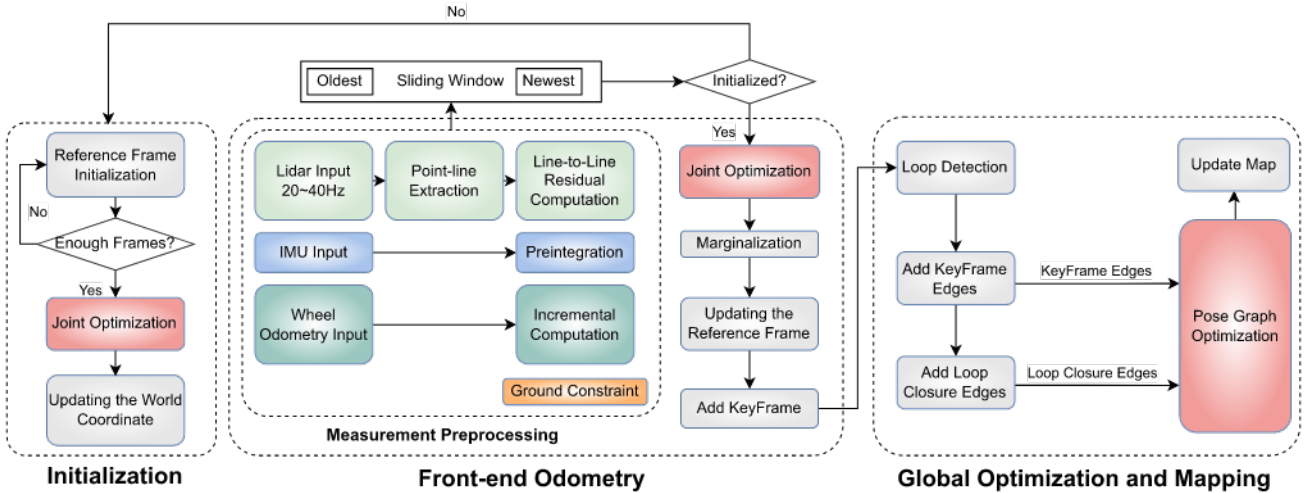
Figure 1: The system structure of 2DLIW-SLAM. The Initialization and Front-end Odometry will be elaborated in detail in Section IV. The Global Optimization and Mapping will be discussed comprehensively in Section V.

2. Related Work

LiDAR Odometry and Mapping. Before the optimization algorithm, SLAM heavily relied on filtering algorithms [8–10]. Gmapping [10] further enhanced filtering performance by reducing the number of particles and minimizing resampling frequency. Subsequently, graph optimization-based techniques began to gain prominence [2, 11, 12]. Notably, Cartographer [2] incorporates the concept of submaps and leverages 2D LiDAR for loop closure detection, thus effectively addressing long-term drift issues.

Subsequently, the primary research focus shifted towards 3D LiDAR SLAM. Zhang *et al.* [13] introduced LOAM, a method leveraging geometric features in the environment to establish LiDAR odometry based on point-line and point-plane constraints, significantly enhancing the accuracy of LiDAR odometry. Additionally, they introduced loop closure detection functionality, culminating in a comprehensive SLAM framework called LeGO-LOAM [14]. Following this, by integrating IMU data and applying ISAM2 [15] theory, they proposed LIO-SAM [4], resulting in further system performance enhancement.

In addition to the aforementioned approaches, Wei *et al.* [3,16] proposed the FAST-LIO and FAST-LIO2 series of methods, which effectively fuse 3D LiDAR and IMU data using the Iterated Error Kalman Filter (iEKF) and have achieved remarkable results.

Recent developments in the SLAM field have gravitated toward multi-sensor fusion, leading to the emergence of SLAM frameworks that integrate LiDAR, IMU, and visual data. Noteworthy examples include R2LIVE [17] and R3LIVE [18] proposed by Lin *et al.*, and LVIO-SAM [19] proposed by Zhong *et* *al..* Zhang *et al.* [20] have improved traditional point cloud registration and loop-detection methods, proposing a new scheme for multi-sensor SLAM systems. GLIO [21] tightly couples GNSS, LiDAR, and IMU integration for continuous and drift-free state estimation of intelligent vehicles in urban areas. Song *et al.* [22] utilized a dual neural network and a square-root cubature Kalman filter to mitigate sensor noise and varied maneuvers. These approaches further enhance localization accuracy and robustness by leveraging multiple sensor modalities.

Wheel Odometry in SLAM. [23] represents the first attempt to fuse LiDAR, IMU, and wheel odometry in a loosely coupled manner. Initially, specific states are computed using data from the LiDAR, IMU, and wheel odometry. Subsequently, these states are integrated into an Extended Kalman Filter (EKF), resulting in the final state estimation. EKF-LOAM [24] also adopts the EKF framework and incorporates a simple and lightweight adaptive covariance matrix based on the detected geometric feature count. The OpenLORIS-Scene [25] dataset provides wheel odometry data for experimentation. However, the incorporation of wheel odometry in SLAM is not yet widespread, and open-source implementations are relatively scarce. [25] also emphasizes that SLAM algorithms should not overlook the advantages and benefits offered by wheel odometry.

Loop Detection. Loop detection is a critical process in SLAM. Cartorgrapher [2] employs a search window-based technique to match the current LiDAR frame with the map for loop closure detection. Kim *et al.* [26] encode a LiDAR frame into rotation-invariant descriptors and accelerates the matching process using the KD-Tree method. Building upon

this algorithm, a descriptor with both rotation and translation invariance, called scancontext++ [27]. Xiang *et al.* [28] combined encoding and machine learning to propose FastLCD for handling point cloud data and achieving reliable and accurate loop closure detection. OverlapTransformer [29] uses a lightweight neural network to expose the range image representation of point clouds, enabling fast retrieval. BoW3D [30] presented a bag-of-words model applicable to 3D LiDAR data, which not only effectively identifies previously visited locations but also real-time corrects the complete pose.

3. Our System Overview

We consistently employ rotation vector θ to represent pose and p to denote the translation vector. The optimized state variables include the robot's position $p_k \in \mathbb{R}^3$ and orientation angles $\theta_k \in$ so(3) in the world coordinate system, velocity $v_k \in \mathbb{R}^3$, as well as acceleration bias $b_{\alpha_k} \in \mathbb{R}^3$ and gyroscope bias $b_{\omega_k} \in \mathbb{R}^3$:

$$X = [p_k, \theta_k, v_k, b_{\alpha_k}, b_{\omega_k}]^T \tag{1}$$

A general SLAM structure is adopted, comprising two main components: front-end odometry and back-end optimization, as depicted in figure 1.

The front-end odometry tightly couples three primary sensors: a 2D LiDAR, an IMU, and wheel odometry. The 2D LiDAR is used to extract line features and optimizes the current frame's pose by matching these features with those from the reference frame. The IMU adopts an IMU pre-integration observation model. On the other hand, the wheel odometry captures changes in motion over a period, with more detailed information provided later in this paper. The implementation of ground constraints ensures that the robot remains well-constrained even under slight perturbations, effectively reducing the robot's 6DoF to 3DoF. Additionally, the odometry periodically transmits keyframes to the back-end.

The back-end primarily consists of loop detection and pose graph optimization. Initially, it receives keyframes from the front-end and establishes relative pose constraints between them. Periodic loop closure detection is performed to establish loop closure constraints in the pose graph, which is then subjected to global optimization. Finally, keyframes will be employed to periodically create a two-dimensional probability grid map.

4. Front-end odometry

4.1. Extracting points and lines

The point-line process employed in this paper consists of two main steps: line extraction and corner extraction

based on the identified lines, as depicted in figure 2. The extracted lines are utilized for front-end state estimation, while the corners play a crucial role in back-end mapping for loop closure detection. The specific point-line extraction process is detailed as follows:

Initially, the continuity is determined based on the distance between neighboring points, so that a scan of points is divided into multiple continuous point sets $\{S_1, ..., S_n\}$.

Each point set S_i is processed independently. For every point P_i, the angle θ formed by the two vectors connecting P_i to its previous point P_{i-1} and next point P_{i+1} are computed.

$$l_i = P_i - P_{i-1}, l_j = P_{i+1} - P_i$$
$$\theta_i = \arccos \frac{l_i l_j}{\|l_j l_j\|}. \tag{2}$$

Due to the small jitter between neighboring points in a scan of points, simply using neighboring points will be more affected by noise and result in larger fluctuations. Therefore, when selecting neighboring points, points that are too close together are skipped. For each θ, we use Non-Maximum Suppression (NMS) to determine the largest angle in the retained neighborhood.

Starting from the initial point P_s of S_i, we use equation (2) to calculate θ formed by P_s, the current potential endpoint P_e^c and the next potential endpoint P_e^n. If θ exceeds a set threshold, they are considered to be in the same line. This process continues until one θ below the threshold is encountered or the endpoint is reached.

All points between P_s and the final endpoint P_e are treated as belonging to the same line, and a least squares fitting is applied to all points within this range.

$$\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix} A = 0 \tag{3}$$

where the line expression $A^T P = 0$ is derived, enabling the simultaneous calculation of the projection coordinates P_s and P_e for the two endpoints on this line. Throughout this paper, line segments are represented using the tuple (P_s, P_e).

Once a line segment is identified, the current point is designated as the new P_s, and the aforementioned process is reiterated until all potential endpoints are processed. To assess the continuity of a detected line segment with adjacent lines, the angle between the two lines is computed. If this angle surpasses a specific threshold, it is regarded as a stable corner point formed by the intersection of the two lines. Using the aforementioned algorithm, stable lines and corners can be effectively extracted. To facilitate rapid subsequent matching, a map is constructed for each scan frame.
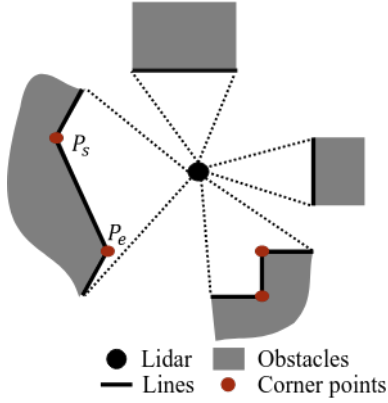
Figure 2: Illustration of line feature extraction. All points are processed using $A^T P = 0$ to obtain the linear equation, and the line segment range is defined by (P_s, P_e).
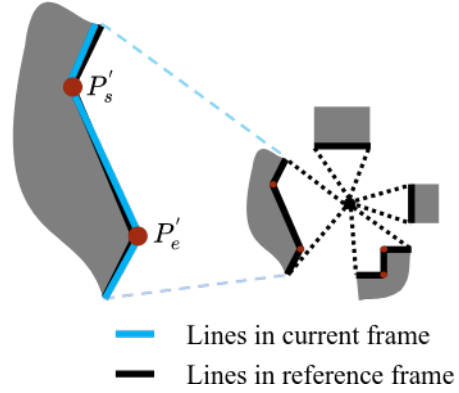

Figure 3: Illustration of line feature matching optimization between the current frame and the reference frame. The distances between $(P_s^{'}, P_e^{'})$ and line in the reference frame are utilized as observational constraints to optimize the pose of the current frame.

4.2. Line-to-Line Alignment Constraints

By leveraging wheel odometry information and past state estimation, we can derive the pose change during this period, represented as $T_{O_c}^{O_r}$. Furthermore, based on the relative pose T_L^O between the LiDAR and the wheel odometry, we calculate the pose change of the LiDAR between the current frame and the reference frame, denoted as $T_{L_c}^{L_r}$. We adopt a specific strategy to select reference frames from historical frames. After that, the extracted lines are compared with line data from the reference frame, which is depicted in figure 3.

Utilizing $T_{L_c}^{L_r}$, we effectively map the lines $l_c = (P_s, P_e)$ in the current scan to the reference frame L_r. In other words, we establish their corresponding positions and orientations in the reference frame.

$$l_c^{'} = (P_s^{'}, P_e^{'}) = (T_{L_c}^{L_r} P_s, T_{L_c}^{L_r} P_e). \qquad (4)$$

Upon identifying these lines, they undergo a threshold check: if the angle between any two lines exceeds a certain threshold, they are discarded from consideration. In cases where multiple lines meet the criteria, the line with the smallest angle is selected as the matched line l_r, thereby forming a line matching pair $l_{m_i} = (l_r, l_c)$. By systematically finding all possible line-line matching pairs between the current frame L_c and the reference frame L_r, a set of matching lines is obtained, denoted as $m = \{l_{m_i} = (l_r, l_c)\}$.

Given that individual lines generally do not strictly overlap, we compute the distance d_1 from the initial point $P_s^{'}$ to the target line l_r and d_2 from the terminal point $P_e^{'}$ to l_r [13]. Specifically, this constraint can be expressed as follows:

$$d_1 = \left\| (P_s^{l_r} - P_e) - \left(\frac{P_e^{l_r} - P_s^{l_r}}{\| P_e^{l_r} - P_s^{l_r} \|} \right)^T \right.$$
$$\left. \cdot (P_s^{l_r} - P_e) \frac{P_e^{l_r} - P_s^{l_r}}{\| P_e^{l_r} - P_s^{l_r} \|} \right\|. \qquad (5)$$

Likewise, d_2 is computed. With both d_1 and d_2 determined, the observation model can now be formulated as follows:

$$h_L(p_r, \theta_r, p_c, \theta_c) = d_1 + d_2 + n_l = 0 \qquad (6)$$

where it is assumed that n_l follows Gaussian noise with a distribution of $n_l \sim \mathcal{N}(\mu, \sigma_l^2)$. This observation model will be integrated into the joint optimization process in the following steps.

4.3. IMU Preintegration

Since the state at time t_{B_k} changes continuously in nonlinear optimization processes, IMU state propagation must be performed frequently, leading to a significant performance loss. To overcome this issue, we adopt the IMU preintegration [31]:

$$\alpha_{B_{k+1}}^{B_k} = \iint_{t_{B_k}}^{t_{B_{k+1}}} R_t^{B_k} (\hat{a}_t - b_{a_t} - n_a) \, dt^2$$

$$\beta_{B_{k+1}}^{B_k} = \int_{t_{B_k}}^{t_{B_{k+1}}} R_t^{B_k} (\hat{a}_t - b_{a_t} - n_a) \, dt \qquad (7)$$

$$\gamma_{B_{k+1}}^{B_k} = I + \int_{t_{B_k}}^{t_{B_{k+1}}} \gamma_t^{B_k} \lfloor \hat{\omega} - b_{\omega_t} - n_\omega \rfloor_\times \, dt.$$

As observed, this integrated part depends solely on the IMU bias values. Finally, the IMU observation model

can be derived as follows:

$$
\begin{bmatrix}
R_W^{B_k}(p_{B_{k+1}}^W - p_{B_k}^W + v_{B_K}^W \Delta t^2) - \alpha_{B_{k+1}}^{B_k} \\
R_W^{B_k}(v_{B_{k+1}}^W - v_{B_k}^W + g^W \Delta t) - \beta_{B_{k+1}}^{B_k} \\
\exp(\theta_{b_{k+1}}^w)^T \exp(\theta_{b_{k+1}}^w)\gamma_{B_{k+1}}^{B_k} \\
b_{a_{B_{k+1}}} - b_{a_{B_k}} \\
b_{\omega_{B_{k+1}}} - b_{\omega_{B_k}}
\end{bmatrix} + n_B
$$

$$
= h_B(p_{B_k}^W, \theta_{B_k}^W, v_{B_k}^W, p_{B_{k+1}}^W, \theta_{B_{k+1}}^W, v_{B_{k+1}}^W) + n_B. \tag{8}
$$

Equation (8) establishes the relationship between the position, orientation, and velocity of the robot in the current frame B_k and the neighboring frame B_{k+1}, along with the corresponding noise term n_B. The noise term n_B follows Gaussian noise, *i.e.*, $n_B \sim \mathcal{N}(\mu, \Sigma_B)$.

4.4. Observation of Wheel Odometry

We adopt the assumption that the data acquired directly from the wheel odometry represents the change in the chassis pose over a period, denoted as $\hat{T}_{O_{i+1}}^{O_i}$. This can be calculated using the values $p_{B_k}^W, \theta_{B_k}^W, p_{B_{k+1}}^W, \theta_{B_{k+1}}^W$.

Furthermore, the wheel odometry data can be transformed into translational magnitude d, directional change θ_d, and pose variation θ as follows:

$$
\hat{d}_{i+1}^i = \|\hat{p}_{i+1}^i\|
$$
$$
(\hat{\theta}_d)_{i+1}^i = atan\left(\frac{(\hat{p}_{i+1}^i)_y}{(\hat{p}_{i+1}^i)_x}\right) \tag{9}
$$
$$
\hat{\theta}_{i+1}^i = \|\log(\hat{R}_{i+1}^i)\|
$$

where $\hat{d}_{i+1}^i$, $(\hat{\theta}_d)_{i+1}^i$, $\hat{\theta}_{i+1}^i$, and $\hat{R}_{i+1}^i$ represent the magnitude of the wheel odometry displacement increment, the angle of displacement increment, the angle of pose change, and the rotation matrix, respectively.

By utilizing these equations, the observation model can be derived as follows:

$$
\begin{bmatrix}
\hat{d}_{O_{k+1}}^{O_k} - d_{O_{k+1}}^{O_k} \\
\hat{\theta}_{d_{O_{k+1}}}^{O_k} - \theta_{d_{O_{k+1}}}^{O_k} \\
\hat{\theta}_{O_{k+1}}^{O_k} - \theta_{O_{k+1}}^{O_k}
\end{bmatrix} + n_O
$$
$$
= h_O(p_{B_k}^W, \theta_{B_k}^W, p_{B_{k+1}}^W, \theta_{B_{k+1}}^W) + n_O. \tag{10}
$$

where n_O follows Gaussian noise, *i.e.*, $n_O \sim \mathcal{N}(\mu, \Sigma_O)$, with its covariance matrix computed simultaneously during the wheel odometry increment extraction process.

4.5. Ground Constraint

The wheel odometry mainly provides motion information in the x-y plane. To achieve a reliable estimation of the 6DoF state, it becomes essential to incorporate ground constraints to stabilize the system.

Given the IMU's pose in the world coordinate system, denoted as $p_{B_k}^W, \theta_{B_k}^W$, we compute the pose of the wheel odometry in the world coordinate system, represented as $p_{O_k}^W, \theta_{O_k}^W$, using the following equation:

$$
(p_{O_k}^W, \theta_{O_k}^W) = \log\left(\exp\left(p_{B_k}^W, \theta_{B_k}^W\right) T_O^B\right). \tag{11}
$$

using equation (11), we can derive the direction vector of the z-axis in the current wheel odometry frame and its value in the world coordinate system, expressed as:

$$
axis_z = \left(\exp(\theta_{O_k}^W)\right)_z
$$
$$
z = (p_{O_k}^W)_z \tag{12}
$$

where $(\cdot)$ denotes the third element of the vector. $axis_z$ and z represent the orientation and value of the z-axis in the current wheel odometry frame, respectively.

We establish an observation model as follows:

$$
\begin{bmatrix}
z \\
\|\arcsin(axis_z \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T)\|
\end{bmatrix} + n_g
$$
$$
= h_g(p_{B_k}^W, \theta_{B_k}^W) + n_g \tag{13}
$$

where n_g follows Gaussian distribution, i.e., $n_g \sim \mathcal{N}(0, \Sigma_g)$, with the actual covariance depending on the level of ground jitter. By employing the ground constraint, the robot's motion is constrained to 3DoF, which aligns better with real-world observations.

4.6. Joint Optimization

By applying Maximum Likelihood Estimation (MLE) to the observations presented in equation (6), equation (8), equation (10) and equation (13), we can effectively solve the following nonlinear problem:

$$
X^* = \arg\min_X \sum_{i=k-w}^{k} \sum_{j=1}^{m_i} \|h_{L_j}(p_r, \theta_r, p_i, \theta_i)\|_{\sigma_l^2}^2
$$
$$
+ \|h_B(p_{B_{k-1}}^W, \theta_{B_{k-1}}^W, v_{B_{k-1}}^W, p_{B_k}^W, \theta_{B_k}^W, v_{B_k}^W)\|_{\Sigma_B}
$$
$$
+ \|h_O(p_{B_{k-1}}^W, \theta_{B_{k-1}}^W, p_{B_k}^W, \theta_{B_k}^W)\|_{\Sigma_O}
$$
$$
+ \sum_{i=o}^{k} \|h_g(p_{B_k}^W, \theta_{B_k}^W)\|_{\Sigma_g}
$$

$$\tag{14}$$

where the subscript k designates the current frame, while $k-1$ corresponds to the previous frame and w denotes the sliding windows size.

In the front-end odometry tracking process, prior factors are introduced simultaneously to aid in the estimation.

4.7. Initialization

Before optimizing the position of the front-end odometry, we first perform an initialization of the

system. This process involves setting the reference frame and using the line features and corner point features of the LiDAR frame in the initialization window to obtain a better initial position. The specific steps involved are as follows:

(i) Accumulate n consecutive non-stationary LiDAR frames, and when a certain quantity criterion k is met, the system initiates the initialization check.

(ii) If each frame can establish a sufficient number of line-to-line correspondences with the first frame's LiDAR lines, the formal initialization process begins. Otherwise, all data is cleared, and the initialization preparation returns to Step i.

(iii) Perform joint optimization based on equation (14).

4.8. Construction of Reference Frames

(i) The system maintains two reference frames: the current reference frame and the backup reference frame. When the current reference frame is empty, the current frame becomes the reference frame.

(ii) Estimate the state of the current frame and map the line data of the current frame onto the current reference frame based on the estimation result. If the corresponding cell already contains line data, the cell remains unchanged; otherwise, the cell is updated.

(iii) When the accumulated data in the reference frame reaches half of the threshold, the current frame data is synchronously updated to both the current reference frame and the backup reference frame.

(iv) If the current reference frame reaches the threshold, it is deleted, and the backup reference frame takes its place as the current reference frame. A new empty backup reference frame is then created. Subsequently, repeat Steps i to iv.

5. Global Optimization and Mapping

5.1. Loop Detection

2D LiDAR scan describes the geometric structure of the environment, e.g., straight lines on walls, corners of rooms. The current common feature descriptors such as ORB [32] or SFIT [33] are difficult to characterize such 2D scans that lack texture features. Therefore, we have developed a method for matching global feature point descriptors that can characterize 2D scan data based on its geometric structure. We employ this method for loop detection, as depicted in 4. In our front-end processing pipeline, we extract corner features from the space between two line segments. These corner features will be employed for loop closure detection. Consequently, corresponding descriptor sets are generated for each keyframe, ensuring adequate information to address the issue of insufficient data in individual frame scans.

Based on the stability of corner points and their invariant relative relationships, the frame matching problem can be formulated as follows: Given two sets of points $M = P_1^M, \ldots, P_m^M$ and $N = P_1^N, \ldots, P_n^N$, where at least k points in set N satisfy the following relationship with set M:

$$P_i^M = T_N^M P_j^N \tag{15}$$

where T_N^M represents the relative pose between the coordinate systems of set M and set N.

For any point set M, upon selecting a point P_i^M from it, we can compute the distance d_{ij}^M and the vector $v = P_i^M - P_j^M$ to other points. Subsequently, the angle between v and $v_x = [1 \quad 0 \quad 0]^T$ is calculated and denoted as $a_{ij}^M \in (-\pi, \pi]$. Similar operations can be performed for the other point set N. When sets M and N have at least k points that satisfy equation (15), the following relationships hold for these k points:

$$\begin{aligned} d_{ij}^M &= d_{ij}^N \\ diff &= Norm(a_{ij}^M - a_{ij}^N) \\ Norm(a) = f(x) &= \begin{cases} a \\ a - \pi & \text{if } a > \pi \\ a + \pi & \text{if } x < -\pi \end{cases} \end{aligned} \tag{16}$$

During the algorithm's actual execution, slight noise and environmental changes may make strict adherence to the length consistency relationship and the angle difference consistency relationship in equation (16) challenging. Therefore, we discretize both the distance d and angle a relationships as follows:

$$\begin{aligned} d &= round(\frac{d}{d_{res}}) \\ a &= round(\frac{a}{a_{res}}) \end{aligned} \tag{17}$$

Where d_{res} represents the allowed maximum distance error, and a_{res} represents the allowed maximum angle error. For any point P_i^M in the point set M, the descriptor $des_i^M = (d_{ij}^M, a_{ij}^M, j)$ is obtained using the above method. We arrange all descriptors in the order of d_{ij}^M from smallest to largest into a list $DesM = [des_i^M]$, which represents the set of descriptors of point set M. In the matching process, we first find the matches d_{ij}^M and d_{ij}^N, and then perform the α_{ij} matching. When the element with the maximum angle difference exceeds the threshold t, the matching is considered established, and at least t pairs of matching points are obtained.

In principle, matching all descriptors from set $DesM$ with all descriptors from set $DesN$ can achieve the highest theoretical success rate. However, such
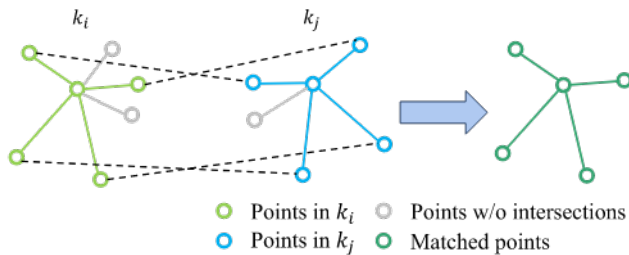
Figure 4: Illustration of loop closure matching. All points are corner feature points, and similarity matching is performed by comparing the distance and angle features between these corner points.

an approach leads to a considerably high algorithmic complexity.

To mitigate this complexity, adopting a randomized strategy proves to be an effective approach. Let the size of point set M be denoted as m, and the size of point set N as n, with their intersection size being c. By randomly selecting c points from point set M, the algorithm can successfully accomplish the matching task. If a success rate of no less than p is desired, let the corresponding number of random trials be denoted as γ. The ensuing relationship can be deduced as follows:

$$1 - (1 - \frac{c}{m})^{\gamma} > p \qquad (18)$$

Continued derivation yields the subsequent expression:

$$\gamma \geq \lceil \frac{\log(1-p)}{\log(1-\frac{c}{m})} \rceil. \qquad (19)$$

In practical operation, successful matching between two frames often results in a relatively substantial intersection c. In many instances, it holds true that $\gamma \ll m$, consequently leading to significant savings in terms of m-k matchings. To illustrate, when $p = 0.95$, $m = 20$, and $c = 10$, implementing the randomized algorithm facilitates a minimum of 75% reduction in matching time. This efficiency gain ensures adept matching outcomes while achieving the specified 95% success rate.

5.2. Pose Graph Opitimization

In the domain of global optimization, we employed a pose graph optimization method, akin to the implementation found in VINS [31].

By employing the aforementioned matching algorithm, we were able to obtain a minimum of k pairs of matching point pairs denoted as p_i^M, p_j^N. Assuming there are K matching point pairs, solving for the relative pose can be equivalently represented as addressing the ensuing nonlinear least squares problem:

$$p_n^M, \theta_n^M = \arg\min \sum_{i=1}^{K} \|p_i^M - \exp(p_N^M, \theta_N^M)P_j^N\|^2. \qquad (20)$$

5.3. Probability Grid Map

The generated map takes the form of a two-dimensional probability grid map. Its approach to updating probabilities closely mirrors that of Cartographer [2], entailing the calculation of hit and miss counts for each individual grid in order to adjust its occupancy probability. Furthermore, the integration of a Gaussian filtering kernel has been introduced to further optimize the grid map.

6. EXPERIMENT

6.1. Experimental Scenarios and Datasets

This paper utilizes the OpenLORIS-Scene [25] dataset, which comprises visual, IMU, wheel odometry, and 2D LiDAR data, tailored for typical indoor scenarios involving service robots. However, the dataset does not cover 3D LiDAR data, so we mainly compare the algorithm proposed in this paper with mainstream 2D LiDAR SLAM and visual SLAM. The dataset features four distinct indoor scenes: office, home, cafe, and corridor. Notably, the office scene simulates a confined space reminiscent of a small-scale office environment. In the home scene, the robot navigates through a typical household, covering the living room and various rooms. The cafe scene recreates the atmosphere of an operational coffee shop, replete with tables, chairs, and a diverse array of patrons, including pedestrians. Transparent glass elements are also introduced, introducing challenges to the precision of the LiDAR. Lastly, the corridor scene offers a narrow and elongated environment, primarily consisting of a lengthy hallway. However, the presence of transparent windows along the corridor significantly influences the localization accuracy of the 2D LiDAR. Table 1 provides an overview of trajectory lengths for each scene. It is pertinent to emphasize that all experiments were executed on an environment employing the AMD Ryzen 7 PRO 1700.

Table 1: Total trajectory length

Scenes	Office	Home	Cafe	Corridor
Length (m)	17.96	40.06	46.84	143.20

Table 2: Absolute pose error compared to previous works.

Method	Office	Home	Cafe	Corridor
Gmapping [10]	0.063	**0.111**	0.174	0.145
Cartographer [2]	0.096	0.114	0.176	0.129
ORB-SLAM3 (S-I) [34]	0.235	0.406	**0.164**	1.131
VINS-Fusion (S-W) [35]	0.323	0.570	0.401	0.504
2DLIW-SLAM w/o loop	0.064	0.140	0.192	0.131
2DLIW-SLAM	**0.060**	0.115	0.170	**0.126**

(a) Trajectories in the home scene   (b) Map constructed by 2D-LIW
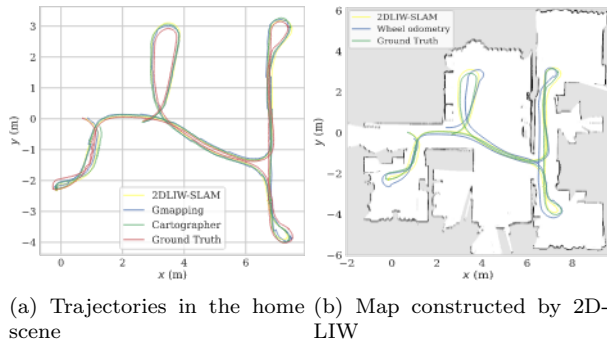
Figure 5: The performance of 2DLIW in home scene. 2DLIW can address the drift error in wheel odometry and achieve effective localization and mapping.

6.2. Pose Error

To evaluate the accuracy of the odometry, we employed the Root Mean Squared Error (RMSE) of Relative Pose Error (RPE) as the primary assessment metric. For the evaluation of localization precision, we utilized the RMSE of Absolute Pose Error (APE). These aforementioned metrics concurrently consider both translation and rotation errors in the evaluation process.

Table 3 presents the RMSE of RPE, with a pose interval set at 0.1 meters. In the office scene, 2DLIWO exhibits exceptional performance, with an impressively low RMSE of 0.0197, surpassing that of Cartographer and Gmapping and gets the lowest errors across all scenes. Furthermore, Cartographer [2] displayed superior tracking performance when contrasted with Gmapping [10].

Table 3: Relative pose error compared to other 2D LiDAR SLAM.

Method	Office	Home	Cafe	Corridor
Gmapping [10]	0.026	0.040	0.049	0.108
Cartographer [2]	0.027	0.027	0.040	0.027
2DLIWO	**0.020**	**0.023**	**0.032**	**0.026**

The RMSE of APE is depicted in table 2. In this experiment, we also tested the open-source visual systems ORB-SLAM3 [34] in Stereo-Inertial (S-I) mode and VINS-Fusion3 [35] in Stereo-Wheel (S-W) mode. Among the four scenes, 2DLIW-SLAM performs best in the office and corridor scenes, demonstrating high accuracy in localization. Particularly in the corridor scene, its APE reaches 0.126, significantly outperforming other algorithms. In the home scene, there are a number of walls, so they lack texture features and instead have a rich geometric structure. Therefore, SLAM based on 2D LiDAR achieves higher accuracy, with Gmapping [10] performing the best. Conversely, in the cafe scene, rich texture details enable the visual-IMU based ORB-SLAM3 [34] to achieve better localization accuracy. When faced with challenges such as texture ambiguity or dynamic objects, visual-wheel SLAM performs even worse than 2D LiDAR SLAM. This suggests that 2D LiDAR offers higher robustness in indoor robotics.

6.3. Loop Detection

Table 4: Global localization in Deutsches Museum Dataset.

Methods	Time (ms)	Translation Error (m)	Rotation Error (°)
AMCL [36]	661.15	0.09	0.78
Cartographer [2]	1021.29	**0.06**	**0.73**
Ours	**8.99**	0.07	0.74

In order to verify the effectiveness of loop detection, we conducted ablation experiments and listed the results in Table 2. The results indicate that after loop detection, optimizing the loop edges helps to improve the SLAM performance.

Furthermore, we isolated the loop matching module and combined it with the Iterative Closest Point (ICP) algorithm to compare global localization effectiveness with AMCL [36] and Cartographer [2] on the Deutsches Museum dataset [2], as depicted

Table 5: Relative Pose Error in Degeneracy Problem

Method	Max	Mean	Median	Min	Rmse	Sse	Std
Gmapping [10]	0.914	0.042	0.013	0.001	0.094	7.125	0.085
Cartographer [2]	0.511	0.018	0.009	0.001	0.042	1.673	0.038
2DLIW-SLAM	**0.312**	**0.012**	**0.006**	**0.002**	**0.027**	**0.851**	**0.024**
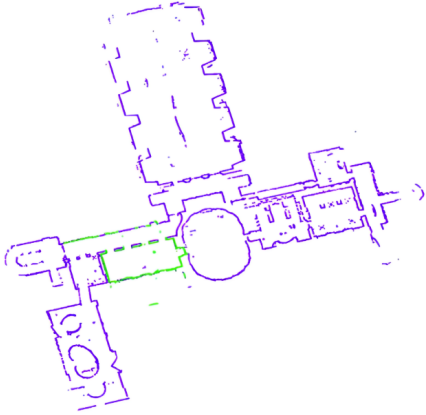
Figure 6: Illustration of successful global localization in Deutsches Museum.

Figure 7: Trajectories in degeneracy problem.
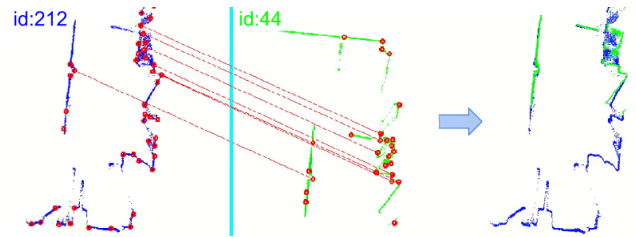
Figure 8: Illustration of successful loop closure matching in the office scene.

in figure 6. Four different times and locations were selected as the starting positions of the global localization algorithm. Subsequently, the performance of global localization was evaluated based on the average processing time for each frame and the accuracy of successful localization. Given the vastness of the dataset's scene, to ensure successful localization, we set the number of particles to 20,000 for AMCL, the number of search layers to 7 for Cartographer, the length of the search window edge to 7m, and the search angle to 30°. As shown in table 4, our matching optimization algorithm not only achieves high localization accuracy, but also exhibits two orders of magnitude lower time complexity compared to other methods.

Additionally, in table 6, we present the average values of translation error and rotation error for the matched frames in the Openloris dataset. These errors are computed from the difference between the ICP optimization of the matched frames and the positions obtained from front-end tracking.

6.4. Degeneracy Problem

Within the corridor scene, we imposed a LiDAR measurement range limit of 3 meters. This restriction aimed to simulate an extensive hallway scenario, characterized by similar geometric attributes. Such scenarios often lead to large errors because of the loss of observation information.
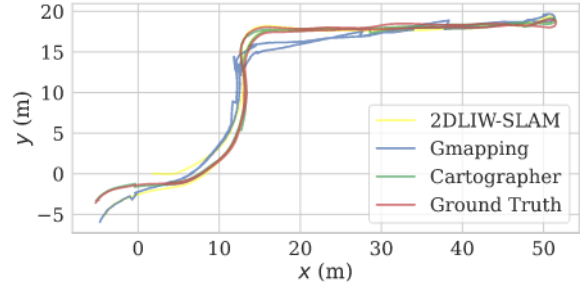
Table 5 provides an overview of RPE in degeneracy problem, and it's apparent that 2DLIW-SLAM exhibits the most robust tracking performance in this particular scenario and gets the lowest RMSE of 0.027. This can be attributed to the tightly-coupled front-end of 2DLIW-SLAM, which effectively integrates wheel odometry, significantly enhancing the resilience of the front-end odometry.

figure 7 visually depicts the trajectories of the three algorithms within this challenging scenario. Both 2DLIW-SLAM and Cartographer exhibit effective global localization. However, due to the constraints of the LiDAR's range (limited to 3 meters), loop closure detection through corner feature matching encounters difficulties, resulting in drift errors.

6.5. Performance of 2DLIW-SLAM

figure 5a elegantly portrays the culmination of 2DLIW-SLAM's positioning and mapping prowess in home scene. While wheel odometry substantively contributes to the positioning process, seamless cooperation among various sensors, coupled with adept loop closure detection, effectively counteracts wheel odometry

Table 6: Performance of Loop Detection

Scenes	Total KFs	Corners / KF	Fast Filter Ratio	Descriptor Match (us)	Frame-to-Frame (us)	Translation Error (m)	Rotation Error (°)
office	259	34.47	41.00%	1.98	394.90	0.09	0.37
home	1944	18.78	93.54%	0.19	68.70	0.03	0.44
cafe	1342	27.96	84.09%	0.54	146.74	0.08	0.75
corridor	2199	18.51	99.51%	0.03	39.65	0.42	0.30

Table 7: Performance of Tracking

Scenes	Num. of Line Extractions	Num. of Line Matches	Time (ms)
office	50.09	30.84	9.47
home	44.26	26.36	13.42
cafe	65.46	38.32	14.66
corridor	42.52	34.50	12.68

drift errors as figure 5b, culminating in remarkable positioning and mapping precision.

Table 7 demonstrates that 2DLIW-SLAM meets the real-time requirements for indoor mobile robot applications. By employing the line extraction approach and line-line matching algorithm presented in this paper, a substantial number of lines can be successfully extracted and matched in the majority of cases. When the sliding window size of 5 is used in the tracking process, the average frame rate can exceed 60 frames. Considering that indoor mobile robots typically operate within a frame rate range of 20 to 30 frames, these results satisfactorily meet real-time demands.

A concrete instance of loop closure detection employing this methodology is portrayed in figure 8, elegantly illustrating the algorithm's adeptness at identifying meaningful matching relationships between keyframes. Noteworthy is the office scenario, where considerable robot jitter results in substantial environmental shifts. Nevertheless, the algorithm reliably pinpoints loop closures.

Table 6 meticulously documents critical facets of back-end optimization. On one hand, the matching time between individual descriptors is short. Concurrently, frame-to-frame matching durations correspond proportionally to the feature point count. On the other end, the proportion of erroneously filtered matches in the rapid filtering phase inversely correlates with the feature point ratio, a correlation well-aligned with expectations. In practical testing, feature points typically range between 15 and 40 per frame. Consequently, this algorithm convincingly maintains its real-time execution capacity across a majority of scenarios.

7. Conclusion

In this paper, we propose a novel, low-cost multi-sensor SLAM system, named 2DLIW-SLAM, designed for indoor mobile robot applications. For the front-end odometry, we incorporate 2D LIDAR, IMU, and wheel odometry, achieving outstanding positioning accuracy through tightly coupled optimization. In the back-end mapping phase, we utilize 2D LiDAR to extract global feature points, implementing efficient loop closure detection. By combining the pose graph, we perform global BA, thereby enhancing the system's positioning accuracy and robustness. Overall, 2DLIW-SLAM meets the requirements for real-time performance and robustness. Also, there are still areas for improvement in the system. For instance, introducing a camera to capture richer visual information and considering the impact of dynamic obstacles on the system. Additionally, the modeling of odometry has not reached an optimal level. In subsequent work, we will make targeted improvements to further enhance the system's performance.

Conflict of interest

The authors declare that they have no conflicts of interest.

ORCID iDs

Bin Zhang https://orcid.org/0009-0006-2024-7248
Zexin Peng https://orcid.org/0009-0002-2749-2319

Bi Zeng https://orcid.org/0000-0001-8596-8333
Junjie Lu https://orcid.org/0009-0004-2506-7567

References

[1] Chinmay Kolhatkar and Kranti Wagle. Review of SLAM Algorithms for Indoor Mobile Robot with LIDAR and RGB-D Camera Technology. In Margarita N. Favorskaya, Saad Mekhilef, Rajendra Kumar Pandey, and Nitin Singh, editors, *Innovations in Electrical and Electronic Engineering*, Lecture Notes in Electrical Engineering, pages 397–409, Singapore, 2021. Springer.

[2] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, Stockholm, Sweden, May 2016. IEEE.

[3] Wei Xu and Fu Zhang. FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, April 2021.

[4] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, October 2020. ISSN: 2153-0866.

[5] Zhenbin Liu, Zengke Li, Ao Liu, Yaowen Sun, and Shiyi Jing. Fusion of binocular vision, 2D lidar and IMU for outdoor localization and indoor planar mapping. *Measurement Science and Technology*, 34(2):025203, November 2022.

[6] Fengyu Liu, Yi Cao, Xianghong Cheng, and Luhui Liu. A visual SLAM method assisted by IMU and deep learning in indoor dynamic blurred scenes. *Measurement Science and Technology*, 35(2):025105, November 2023.

[7] Guojian He, Yisha Liu, and Chengxiang Li. Tightly coupled laser-inertial pose estimation and map building based on B-spline curves. *Measurement Science and Technology*, 34(12):125130, September 2023.

[8] Bruno Steux and Oussama El Hamzaoui. tinySLAM: A SLAM algorithm in less than 200 lines C-language program. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 1975–1979, December 2010.

[9] Isaac Miller and Mark Campbell. Rao-Blackwellized Particle Filtering for Mapping Dynamic Environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3862–3869, April 2007. ISSN: 1050-4729.

[10] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, February 2007.

[11] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient Sparse Pose Adjustment for 2D mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29, October 2010. ISSN: 2153-0866.

[12] Hugh Durrant-Whyte, Nicholas Roy, and Pieter Abbeel. A Linear Approximation for Graph-Based Simultaneous Localization and Mapping. In *Robotics: Science and Systems VII*, pages 41–48. MIT Press, 2012.

[13] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, July 2014.

[14] Tixiao Shan and Brendan 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)*, pages 4758–4765, October 2018. ISSN: 2153-0866.

[15] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, May 2011. ISSN: 1050-4729.

[16] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. FAST-LIO2: Fast Direct LiDAR-Inertial Odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, August 2022.

[17] Jiarong Lin, Chunran Zheng, Wei Xu, and Fu Zhang. R ^2 LIVE: A Robust, Real-Time, LiDAR-Inertial-Visual Tightly-Coupled State Estimator and Mapping. *IEEE Robotics and Automation Letters*, 6(4):7469–7476, October 2021.

[18] Jiarong Lin and Fu Zhang. R3LIVE: A Robust, Real-time, RGB-colored, LiDAR-Inertial-Visual tightly-coupled state Estimation and mapping package. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10672–10678, May 2022.

[19] Xinliang Zhong, Yuehua Li, Shiqiang Zhu, Wenxuan Chen, Xiaoqian Li, and Jason Gu. LVIO-SAM: A Multisensor Fusion Odometry via Smoothing and Mapping. In *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 440–445, December 2021.

[20] Guangyi Zhang, Tao Zhang, and Chen Zhang. Accurate real-time SLAM based on two-step registration and multimodal loop detection. *Measurement Science and Technology*, 34(2):025201, November 2022.

[21] Xikun Liu, Weisong Wen, and Li-Ta Hsu. GLIO: Tightly-Coupled GNSS/LiDAR/IMU Integration for Continuous and Drift-Free State Estimation of Intelligent Vehicles in Urban Areas. *IEEE Transactions on Intelligent Vehicles*, 9(1):1412–1422.

[22] Rui Song, Yongchun Fang, and Haoqian Huang. Reliable Estimation of Automotive States Based on Optimized Neural Networks and Moving Horizon Estimator. *IEEE/ASME Transactions on Mechatronics*, 28(6):3238–3249, December 2023.

[23] Shaojiang Zhang, Yanning Guo, Qiang Zhu, and Zhiyuan Liu. Lidar-IMU and Wheel Odometer Based Autonomous Vehicle Localization System. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 4950–4955, June 2019. ISSN: 1948-9447.

[24] Gilmar P. Cruz Júnior, Adriano M. C. Rezende, Victor R. F. Miranda, Rafael Fernandes, Héctor Azpúrua, Armando A. Neto, Gustavo Pessin, and Gustavo M. Freitas. EKF-LOAM: An Adaptive Fusion of LiDAR SLAM With Wheel Odometry and Inertial Data for Confined Spaces With Few Geometric Features. *IEEE Transactions on Automation Science and Engineering*, 19(3):1458–1471, July 2022.

[25] Xuesong Shi, Dongjiang Li, Pengpeng Zhao, Qinbin Tian, Yuxin Tian, Qiwei Long, Chunhao Zhu, Jingwei Song, Fei Qiao, Le Song, Yangquan Guo, Zhigang Wang, Yimin Zhang, Baoxing Qin, Wei Yang, Fangshi Wang, Rosa H. M. Chan, and Qi She. Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3139–3145, May 2020. ISSN: 2577-087X.

[26] Giseop Kim and Ayoung Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*,

pages 4802–4809, October 2018. ISSN: 2153-0866.

[27] Giseop Kim, Sunwook Choi, and Ayoung Kim. Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments. *IEEE Transactions on Robotics*, 38(3):1856–1874, June 2022.

[28] Haodong Xiang, Wenzhong Shi, Wenzheng Fan, Pengxin Chen, Sheng Bao, and Mingyan Nie. FastLCD: A fast and compact loop closure detection approach using 3D point cloud for indoor mobile mapping. *International Journal of Applied Earth Observation and Geoinformation*, 102:102430, October 2021.

[29] Junyi Ma, Jun Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition. *IEEE Robotics and Automation Letters*, 7(3):6958–6965, July 2022.

[30] Yunge Cui, Xieyuanli Chen, Yinlong Zhang, Jiahua Dong, Qingxiao Wu, and Feng Zhu. BoW3D: Bag of Words for Real-Time Loop Closing in 3D LiDAR SLAM. *IEEE Robotics and Automation Letters*, 8(5):2828–2835, May 2023.

[31] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, August 2018.

[32] Fanyu Zeng, Wenchao Zeng, and Yan Gan. ORB-SLAM2 with 6DOF Motion. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 556–559, June 2018.

[33] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[34] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, December 2021.

[35] Kejian J. Wu, Chao X. Guo, Georgios Georgiou, and Stergios I. Roumeliotis. VINS on wheels. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162.

[36] Dieter Fox. KLD-sampling: Adaptive particle filters. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 713–720, Cambridge, MA, USA, January 2001. MIT Press.