

Article

A Tightly Coupled LiDAR-Inertial SLAM for Perceptually Degraded Scenes

Lin Yang ^{1,2,*} , Hongwei Ma ^{1,2}, Yan Wang ^{1,2}, Jing Xia ^{1,2} and Chuanwei Wang ^{1,2}

¹ School of Mechanical Engineering, Xi'an University of Science and Technology, Xi'an 710054, China; mahw@xust.edu.cn (H.M.); 16105301002@stu.xust.edu.cn (Y.W.); xiajing1984@xust.edu.cn (J.X.); wangchuanwei228@xust.edu.cn (C.W.)

² Shaanxi Key Laboratory of Mine Electromechanical Equipment Intelligent Monitoring, Xi'an 710054, China

* Correspondence: 17101016007@stu.xust.edu.cn

Abstract: Realizing robust six degrees of freedom (6DOF) state estimation and high-performance simultaneous localization and mapping (SLAM) for perceptually degraded scenes (such as underground tunnels, corridors, and roadways) is a challenge in robotics. To solve these problems, we propose a SLAM algorithm based on tightly coupled LiDAR-IMU fusion, which consists of two parts: front end iterative Kalman filtering and back end pose graph optimization. Firstly, on the front end, an iterative Kalman filter is established to construct a tightly coupled LiDAR-Inertial Odometry (LIO). The state propagation process for the a priori position and attitude of a robot, which uses predictions and observations, increases the accuracy of the attitude and enhances the system robustness. Second, on the back end, we deploy a keyframe selection strategy to meet the real-time requirements of large-scale scenes. Moreover, loop detection and ground constraints are added to the tightly coupled framework, thereby further improving the overall accuracy of the 6DOF state estimation. Finally, the performance of the algorithm is verified using a public dataset and the dataset we collected. The experimental results show that for perceptually degraded scenes, compared with existing LiDAR-SLAM algorithms, our proposed algorithm grants the robot higher accuracy, real-time performance and robustness, effectively reducing the cumulative error of the system and ensuring the global consistency of the constructed maps.

Keywords: perceptually degraded scenes; LiDAR; IMU; state estimation; SLAM



Citation: Yang, L.; Ma, H.; Wang, Y.; Xia, J.; Wang, C. A Tightly Coupled LiDAR-Inertial SLAM for Perceptually Degraded Scenes. *Sensors* **2022**, *22*, 3063. <https://doi.org/10.3390/s22083063>

Academic Editors: Luis Payá, Oscar Reinoso García and Helder Jesus Arajo

Received: 21 March 2022

Accepted: 14 April 2022

Published: 15 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

SLAM is the basis for autonomous navigation in robots [1], which use various sensors to conduct real-time 6DOF state estimation in 3D space; this state estimation is the key to achieving high-performance SLAM. The LiDAR-based approach can obtain accurate range information for a scene in 3D space in real time and is invariant to ambient lighting conditions. However, due to the problems of motion distortion [2], low-frequency updates, and sparse point clouds [3] in LiDAR measurements, pure LiDAR is unsuitable for robots dealing with aggressive movements or repetitive structures, such as tunnels or narrow corridors. The shortcomings of LiDAR can be compensated for by fusing IMU. Unlike LiDAR, IMU is not affected by drastic changes in structural features or the environment and can provide high-precision pose estimation with high frequency in a short time. However, due to noise and the bias of IMU sensors, error accumulation can drift over time. To overcome these drawbacks of independent sensors, the reliability of state estimation [4,5] can be improved by fusing multiple sensors together. Thus, the robustness of state estimation can be effectively improved through tightly coupled LIO. Many studies have shown that introducing loop detection in SLAM systems can effectively resolve pose drift caused by the accumulation of sensor errors [6,7]. After detecting the loop frame, the pose is continuously optimized to ensure the global consistency of the map, which greatly improves the positioning accuracy.

Despite the superior performance of 3D LiDAR, scenes characterized by single spatial structures, complex environments, uneven illumination [8], narrow spaces and other issues pose new challenges for LiDAR-SLAM: (1) The geometric features in an environment that are recognized by LiDAR (such as points, lines and surfaces) are usually manually defined by mathematical methods, and LiDAR-based solutions can easily degrade when robots operate in environments with no geometric features, repetitive environments and symmetrical structures. (2) The data measured by LiDAR show a high degree of similarity in one dimension, mainly because of the sidewalls in promenade environments, which can provide a lateral localization reference but not a vertical one. In this case, the feature matching of the sidewalls can easily lead to incorrect attitude constraint estimates. (3) Repeated laser features can lead to misleading constraints and attitude drift in the longitudinal direction such that the point cloud alignment converges to the wrong result and the localization performance deteriorates rapidly or fails completely. This severely affects overall SLAM performance and limits the ability of robots to operate autonomously in such environments. At the same time, as the mileage of a robot increases, the trajectory estimated by the odometer will gradually deviate from the real trajectory. To improve the positioning accuracy of the system and the global consistency of the map and suppress the accumulated error of the odometer, when the robot recognizes that the environment is near the pose of a historical frame, relative pose constraints are integrated into the cost function by calculating the relative relationship between the current frame and the loop frame to establish the solution for the overall pose of the system. The main difficulties of loop detection are as follows: (1) Perceived ambiguity. For example, scenes with very similar structures, such as corridors, tunnels and stairs, increase the difficulty of judgment. (2) The sparsity of the point clouds extracted by the LiDAR sensor itself limits the robustness and discrimination of the observation data. (3) The scale of the data can gradually increase the number of frames to be judged as the running time increases, which reduces the real-time performance of the map construction.

Robust 6DOF state estimation and high-performance SLAM can be used to enable robots to more effectively deal with these special perceptually degraded scenes. In this paper, we propose a tightly coupled LiDAR-SLAM fusion algorithm with two sensors, LiDAR and IMU, as input information sources. The framework of the system consists of two parts: front end iterative Kalman filtering and back end pose graph optimization, these are less dependent on external features, have a strong ability to adapt to the environment and are especially suitable for perceptually degraded scenes such as corridors and roadways. However, in perceptually degraded scenes, ensuring the tradeoff between the estimation accuracy, computational complexity and robustness of SLAM systems is still a major problem faced by tightly coupled LIO [9]. Therefore, research in this area is of great importance.

The main contributions of this paper are as follows:

- (1) We propose a tightly coupled LiDAR-IMU fusion SLAM algorithm that uses both LiDAR and IMU sensors as input information for perceptually degraded scenes. The results show that our algorithm improves the accuracy, real-time performance and robustness of robots working in such scenarios.
- (2) Our algorithm achieves robot state updates by establishing an error-state-based iterative Kalman filter at the front end and adds loop detection and ground constraints in the tightly coupled framework at the back end, which effectively reduces the cumulative error of the system and ensures the global consistency of the constructed maps by optimizing the relative poses between adjacent keyframes.
- (3) Our algorithm fuses different types of sensors in a tightly coupled framework to achieve robust 6DOF state estimation and high-performance SLAM for perceptually degraded scenes and outputs motion trajectories, a global map and state estimation in real time after global graph optimization.

The remainder of this paper is organized as follows: In Section 2, we discuss relevant related research. We outline the framework of the system and provide a detailed system

overview in Section 3. We describe the pre-processing of the sensor data in preparation for the subsequent work in Section 4. The construction of the front end of the tightly coupled LIO is described in Section 5. We conduct back end pose graph optimization in Section 6. The relevant experiments and results are analyzed in Section 7, followed by conclusions in Section 8.

2. Related Work

Of the many studies related to LiDAR-SLAM, we focus on work related to SLAM algorithms for 6DOF state estimation and multi-sensor fusion in robots. The different fusion frameworks of these algorithms are usually classified as loosely coupled and tightly coupled.

2.1. LiDAR Odometry and Mapping

LiDAR odometry is usually performed by using scan-matching methods such as ICP [10] and GICP [11] to calculate the relative positional transformation between two consecutive LiDAR frames. Feature-based matching methods have become a popular alternative to matching complete point clouds due to their computational efficiency. For example, in [12], a plane-based real-time LiDAR odometry alignment method was proposed. Assuming an operation in a structured environment, this method extracts planes from a point cloud and matches them by solving a least squares problem. In [13,14], LiDAR odometry was implemented by matching feature points with edge lines and planes using the LOAM algorithm and its improved algorithm, which achieved good results in various scenarios. However, the scanned point cloud is often skewed due to the internal rotation mechanism of the 3D LiDAR and the motion of the sensor. The use of LiDAR alone for attitude estimation is not ideal because using skewed point clouds or features for registration will eventually lead to large drifts. Therefore, LiDAR is often used in conjunction with other sensors, such as cameras or IMUs, for state estimation and SLAM. For example, in [15–19], IMU, LiDAR and GNSS were fused into an extended Kalman filter (EKF) in the optimization phase for robot state estimation.

2.2. Loosely Coupled LiDAR-Inertial Odometry

The loosely coupled fusion method usually processes the measurement of each sensor input separately and weights the measurement results to infer the motion state of the robot. LOAM [13] and LeGO-LOAM [14] are loosely coupled systems in which IMU measurements are typically used to mitigate laser degradation in featureless environments, and only the IMU-calculated pose is used as an initial estimate for LiDAR scan alignment and motion distortion correction, not as a constraint for global optimization. The EKF is a commonly used method for loosely coupled fusion. To improve the localization accuracy of autonomous robots, ref. [20] incorporated a wheel speedometer in the loosely coupled LIO framework and established the observation equations of the EKF using the velocity and distance measured by the wheel speedometer. Further, ref. [21] fused LiDAR, IMU, odometry and GNSS based on graph optimization, with the flexible handling of scan matching and joint IMU constraints as well as other observation constraints. In general, loosely coupled methods are computationally efficient [22] and relatively flexible in implementation. However, because these methods ignore the correction of the internal state of the IMU, they are easily disturbed by noise, and the loosely coupled system treats the measurement part as a black box and decouples the measurements of individual sensors; this decoupling leads to information loss [23].

2.3. Tightly Coupled LiDAR-Inertial Odometry

Unlike loosely coupled methods, tightly coupled methods typically fuse raw LiDAR feature points (rather than the scanning registration results) with IMU data, which yields a high robustness and accuracy, and can be divided into optimization-based and filter-based methods.

An optimization-based method [7] introduced IMU in the LIO-mapping system pre-integration to minimize the residuals of the IMU and LiDAR measurements; the proposed approach can correct long-term drift through loop detection to achieve good accuracy, but the real-time application of the algorithm is time-consuming. As one of the most advanced algorithms available, LIO-SAM [24] has undergone a large number of experimental tests and outperforms many LIO algorithms that only use LiDAR or are loosely coupled. The sliding window of the LiDAR keyframe is introduced to limit the computational complexity, and the factor graph is used for joint IMU and LiDAR constraint optimization to obtain higher accuracy. This fusion framework is based on a factor graph, which is suitable for multi-sensor fusion. The filter-based method often uses an EKF. However, it is susceptible to linearization errors, which can lead to poor performance and even divergent results [25]. The authors of [6] proposed LINS, which is a robot-centered LiDAR-inertial estimator that recursively corrects the robot state estimation for ground vehicles by using an error state iterative Kalman filter. Although this method has high accuracy and high computational efficiency, it also has drift problems in the long-term navigation process. To reduce the impact of estimation errors caused by linearization, R-LINS [26], FAST-LIO [27] and FAST-LIO2 [28] all use an iterative Kalman filter. Compared with R-LINS and FAST-LIO, FAST-LIO2 uses a new data structure ikd-Tree to save the map; this method can register the original points directly to the map without extracting features, which enables the algorithm to work faster and more accurately.

The most significant difference between our approach and other tightly coupled methods is the specific environment of perceptual degradation. We have chosen a filter-based optimization approach at the front end to improve the system's robustness, which often uses an extended Kalman Filter (EKF). However, the EKF can easily cause a system to be affected by linearization errors [29]. With regard to the scanning constraints observed by LiDAR, this disadvantage is particularly prominent. If the initial pose is incorrect and leads to the mismatch of extracted features, it is considered to be highly nonlinear. To eliminate the errors caused by incorrect matching, we propose an iterative Kalman filter [30], which can repeatedly find a better match in each iteration. In addition, we use an error state representation to ensure the validity of the linearization [31]. Therefore, our proposed approach constructs an iterative Kalman filter on the front end to fuse LiDAR and IMU measurements to achieve tightly coupled LIO, thus laying the groundwork for back end global optimization studies.

3. Overview

3.1. System Framework

Our system receives data from 3D LiDAR and IMU sensors. The goal of the system is for a robot to perform real-time 6DOF state estimation and build a globally consistent map. The system framework of our proposed algorithm, which is shown in Figure 1, is mainly composed of two parts: front end iterative Kalman filtering and back end pose graph optimization. First, pre-processing is conducted on the sensor measurements to compensate for the motion of the original point cloud by pre-integrating the IMU measurements between two consecutive LiDAR frames. Then, stable features are extracted from the corrected point cloud. On the front end (Section 5), the sensor data are pre-processed, the IMU prediction model and the LiDAR observation model are built, the iterative Kalman filter is constructed and the a priori positional attitude of the robot is propagated through a state propagation process, which consists of predictions and observations, to increase the accuracy of the updated pose. Then, the loop is iterated to obtain a tightly coupled LIO. On the back end (Section 6), the keyframe is used to determine if the current pose should be added to the pose graph optimization and loop detection is used to determine if the current position is revisited, which can be expressed as a maximum a posteriori (MAP) problem. In addition, loop detection and ground constraints are added to the tightly coupled framework to optimize the relative pose between adjacent keyframes, reduce cumulative drift, maintain global consistency and improve the positioning accuracy.

Finally, information such as the motion trajectory, global map and state estimation are output after pose graph optimization.

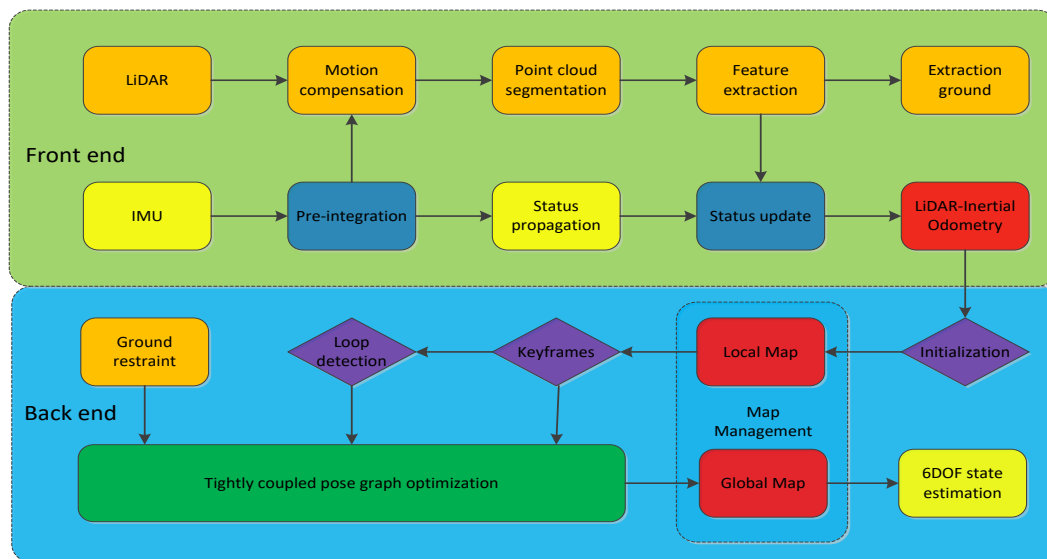


Figure 1. System framework.

3.2. Relevant Agreement

To clearly describe the ranging process, we define three reference coordinates. Additionally, we assume that the IMU and robot coordinates are consistent, the origin of the world coordinates is the origin of the robot coordinates when the coordinate is initialized, and the robot coordinates are consistent with the center of the LiDAR and IMU coordinates. Because the connection between the sensors is rigid, the external calibration between the LiDAR and IMU sensors is obtained by offline calibration. The z-axis of the world coordinate is defined as coinciding with the normal to the surface of the Earth's ellipsoid, with the other two axes lying in the plane orthogonal to the z-axis, which conforms to the right-hand rule. Table 1 lists the meanings of all variables.

Table 1. Meaning of the variables involved in this paper.

Symbol	Meaning	Symbol	Meaning
$p \in \mathbb{R}^3$	Position	p_A^B	Translation of A to B (matrix)
$v \in \mathbb{R}^3$	Velocity	q_A^B	Rotation of A to B (quaternion)
$\theta \in \mathbb{R}^3$	Euler angular pose	R_A^B	Rotation of A to B (matrix)
q	Quaternion pose	t_A^B	Translation of A to B (vector)
$T \in \mathbb{R}^4$	Transformation matrix	T_A^B	Transformation matrix from A to B
$\omega \in \mathbb{R}^3$	Angular velocity	w	World coordinates
$a \in \mathbb{R}^3$	Acceleration	l	LiDAR coordinates
$g \in \mathbb{R}^3$	Gravity	b	Robotics, IMU coordinates
$b_a, b_\omega \in \mathbb{R}^3$	Acceleration and angular velocity bias	b_k	kth frame of LiDAR under B
$n_a, n_\omega \in \mathbb{R}^3$	Acceleration and angular velocity noise	$(\bullet)^k$	State of the kth frame
P	3D point cloud	$(\bullet)_t$	State at moment t
M	Map	$(\bullet)$	Priori state

4. Sensor Data Pre-Processing

4.1. IMU Pre-Integration

The IMU pre-integration algorithm was proposed by Geneva [32]. To avoid continuous propagation of the initial attitude, the world coordinates are separated and only the state

change between two consecutive frames in the IMU coordinate is calculated. The position, speed and attitude of two consecutive frames, b_k and b_{k+1} , can be expressed as:

$$\begin{aligned} p_{b_{k+1}}^{b_k} &= R_w^{b_k} p_{b_{k+1}}^w = R_w^{b_k} \left(p_{b_k}^w + v_{b_k}^w \Delta t_k - \frac{1}{2} g^w \Delta t_k^2 \right) + \alpha_{b_{k+1}}^{b_k} \\ v_{b_{k+1}}^{b_k} &= R_w^{b_k} v_{b_{k+1}}^w = R_w^{b_k} \left(v_{b_k}^w - g^w \Delta t_k \right) + \beta_{b_{k+1}}^{b_k} \\ \gamma_{b_{k+1}}^{b_k} &= \left[q_{b_k}^w \right]^{-1} \otimes q_{b_{k+1}}^w \end{aligned} \quad (1)$$

where $\alpha_{b_{k+1}}^{b_k}$, $\beta_{b_{k+1}}^{b_k}$ and $\gamma_{b_{k+1}}^{b_k}$ are the real values of the IMU pre-integration. Therefore, the IMU pre-integration model can be simplified as:

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \iint_{t \in [t_k, t_{k+1}]} R \left(\gamma_t^{b_k} \right) (\hat{a}_t - b_{a_t} - n_a) dt^2 \\ \beta_{b_{k+1}}^{b_k} &= \int_{t \in [t_k, t_{k+1}]} R \left(\gamma_t^{b_k} \right) (\hat{a}_t - b_{a_t} - n_a) dt \\ \gamma_{b_{k+1}}^{b_k} &= \int_{t \in [t_k, t_{k+1}]} \left[\frac{1}{2} (\hat{\omega}_t - b_{\omega_t} - n_\omega) \right]_R \gamma_t^{b_k} dt \end{aligned} \quad (2)$$

4.2. Motion Compensation

Due to the rotational mechanism inside the LiDAR sensor, the LiDAR measurements are not synchronized and, therefore, the current frame point cloud is prone to distortion. The LiDAR moves at constant angular and linear velocities at a constant speed. Therefore, the timestamp information of each point can be obtained, a linear interpolation operation used to obtain the pose transformation matrix corresponding to each moment, and the motion compensation of the current frame point cloud can be achieved by using IMU measurements [13,14].

The current frame point cloud is divided into N subframes by P_k , where P_k^i is the i th subframe of the current laser frame, and $i \in \{1, 2, \dots, N\}$. Let the pose transformation matrix between two consecutive LiDAR frames in the time interval $[t_k, t]$ be T_k^L , which can be obtained by pre-integration of the IMU data and is not explained in detail here. Then, $T_{(k,i)}^L$ is the current i th subframe for the LiDAR attitude transformation in the time interval $[t_k, t_i]$, which can be calculated by the linear interpolation of:

$$T_{(k,i)}^L = \frac{t_i - t_k}{t - t_k} T_k^L \quad (3)$$

By performing the current subframe transformation, the motion compensated point cloud for P_k' is obtained as:

$$P_k' = \left\{ T_{(k,1)}^L P_k^1, T_{(k,2)}^L P_k^2, \dots, T_{(k,i)}^L P_k^i \right\}, i = 1, 2, \dots, N \quad (4)$$

4.3. Feature Extraction

The feature extraction and matching method in this paper is similar to the one introduced in LOAM [13]: the smoothness evaluation variables are first used to select edge points and plane points, and then the corresponding edge line points and plane points in other keyframes are selected and used to construct the residuals of the LiDAR observation model. When a new frame of LiDAR point cloud data is motion compensated, feature extraction is performed. Firstly, by calculating the local smoothness, edge and plane features can be extracted. In addition, the reflectance of a point is used as an additional determinant. If the reflectance of a point is different from the adjacent threshold, it is also considered to be another edge point. Second, the curvature of the points is used to calculate the plane smoothness as an indicator for extraction of the feature information of the current frame.

Let S be the set of consecutive points i returned by the LiDAR in the same scan. Then, the smoothness c of the local surface can be defined and evaluated as:

$$c = \frac{1}{|S| \cdot \|X_{(k,i)}^L\|} \left\| \sum_{j \in S, j \neq i} (X_{(k,i)}^L - X_{(k,j)}^L) \right\| \quad (5)$$

The feature points can be divided into two main categories: plane points and edge points. Then, the points in the scan are sorted according to the c value and the maximum c point is selected as the edge point, that is, the point on the sharp edge in 3D space, which varies greatly in size compared to the surrounding points and has a higher curvature and a higher smoothness. The minimum c point is selected as the plane point, that is, the point on a smooth plane in 3D space, which varies slightly in size compared to the surrounding points and has a lower curvature and a lower smoothness. The LiDAR scanning features of the extracted edge line and plane at time I are expressed as $\mathbb{F}_{ei}$ and $\mathbb{F}_{pi}$. Then, a set of all extracted features $\mathbb{F}_i$ in frame i is formed, which can be expressed as $\mathbb{F}_i = \{\mathbb{F}_{ei}, \mathbb{F}_{pi}\}$.

4.4. Extraction Ground

Inspired by [14], adding ground constraints can provide additional constraints on the z-axis displacements of keyframe nodes when the ground robot works in different spatial regions while reducing cumulative errors by using more constraint information to improve the pose estimation accuracy. The key to establishing ground constraints is to accurately extract the road planes; determining how to spend the shortest amount of time to complete the planar segmentation and choosing a robust estimation method with few iterations, a high resistance to noise and random sample consensus (RANSAC) are also essential processes [33]. According to the basic principle of RANSAC, a plane is obtained by selecting any three points from each frame of a point cloud. The most commonly used plane equation is $ax + by + cz = d$, where $a^2 + b^2 + c^2 = 1$, $d > 0$, (a, b, c) is the plane normal vector and d is the distance from the LiDAR sensor to the plane. These four parameters can determine a plane.

Because the robot may travel over uneven ground, it is not possible to construct pose maps with fixed planes as nodes as a constraint. To ensure that the constructed constraints are consistent with the actual situation, a subgraph-based ground extraction method is proposed. Based on the constructed local point cloud map, the true planar parameters are extracted as follows: $\pi_m = [n_a, n_b, n_c, d]$. To improve the efficiency of ground extraction, the current position is taken as the origin and the point cloud is searched for in a submap within the radius of the LiDAR measurement distance. Based on the 3D LiDAR attitude at the current moment, the ground parameters are converted to LiDAR coordinates using Equations (6) and (7):

$$[n'_a, n'_b, n'_c]^T = R_t \cdot [n_a, n_b, n_c]^T \quad (6)$$

$$d' = d - t_t \cdot [n'_a, n'_b, n'_c]^T \quad (7)$$

where $\pi'_m = [n'_a, n'_b, n'_c, d]$ denotes the coordinates of the ground in the LiDAR coordinate and $[R_t, t_t]$ is the positional attitude of the LiDAR at moment t .

The calculation of the error between the attitude node and the ground plane node [34] can be expressed as:

$$e_{i,m} = q(\pi'_m) - q(\pi_t) \quad (8)$$

$$q(\pi) = \left[\arctan\left(\frac{n_a}{n_b}\right), \arctan\left(\frac{n_c}{n_b}\right), d \right] \quad (9)$$

5. Front End: Tightly Coupled LiDAR-Inertial Odometry

Tightly coupled LIO uses IMU measurements and features extracted by LiDAR in two consecutive scans to estimate the relative positional transformation of the robot. To avoid

large linearization errors caused by growing uncertainty due to IMU measurements [35], we constructed a robot-centric iterative Kalman filter.

5.1. State Definitions

To facilitate the derivation of the iterative Kalman filter, we define the state $x_{b_{k+1}}^{b_k}$ from time k to $k + 1$ under the IMU coordinate, expressed as:

$$x_{b_{k+1}}^{b_k} := [p_{b_{k+1}}^{b_k}, v_{b_{k+1}}^{b_k}, q_{b_{k+1}}^{b_k}, b_a, b_g, g^{b_k}] \quad (10)$$

According to the basic principle of the ESKF [36], once the error state δx is solved, we add the error state to the a priori state $\bar{x}_{b_{k+1}}^{b_k}$ to obtain the a posteriori state $x_{b_{k+1}}^{b_k}$, which can be expressed as:

$$x_{b_{k+1}}^{b_k} = \bar{x}_{b_{k+1}}^{b_k} \oplus \delta x = \begin{bmatrix} \bar{p}_{b_{k+1}}^{b_k} + \delta p \\ \bar{v}_{b_{k+1}}^{b_k} + \delta v \\ \bar{q}_{b_{k+1}}^{b_k} \otimes \exp(\delta \theta) \\ \bar{b}_a + \delta b_a \\ \bar{b}_g + \delta b_g \\ \bar{g}^{b_k} + \delta g \end{bmatrix} \quad (11)$$

where $\otimes$ is the product of quaternions, and $\exp(\cdot)$ represents the Euler angles to quaternions.

5.2. Definition of the Motion Equation

The equation of motion over linearized continuous time is defined as:

$$\delta \hat{x}(t) = F_t \delta x(t) + G_t w \quad (12)$$

where w is the Gaussian noise vector $w = [n_a^T, n_g^T, n_{b_a}^T, n_{b_g}^T]^T$, F_t is the error state transfer matrix at moment t and G_t is the noise Jacobi matrix at moment t .

$$F_t = \begin{bmatrix} 0 & I_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -R_t^{b_k} [\hat{a}_t]_{\times} & -R_t^{b_k} & 0 & -I_3 \\ 0 & 0 & -[\hat{\omega}_t]_{\times} & 0 & -I_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

$$G_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -R_t^{b_k} & 0 & 0 & 0 \\ 0 & -I_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & I_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (14)$$

where $[\cdot]_{\times} \in \mathbb{R}^{3 \times 3}$ represents the conversion of a 3D vector to a skew-symmetric matrix. I_3 is the unit matrix. $\hat{a}_t$ is the true value of the acceleration at moment t and $\hat{\omega}_t$ is the true value of the angular velocity at moment t .

Therefore, the equation of motion linearized in continuous time by Equation (12) is discretized [37] to obtain the mean value of the prior state via Equation (15) and the variance via Equation (16), respectively.

$$\delta x_{i+1} = (I + F_i \Delta t) \delta x_i \quad (15)$$

$$P_{i+1}^{b_k} = (I + F_i \delta t) P_i^{b_k} (I + F_i \delta t)^T + (V_i \delta t) Q (V_i \delta t)^T \quad (16)$$

where P is the covariance matrix of δx . Q is the covariance matrix of w , which is obtained offline by sensor calibration.

5.3. Definition of the Observation Equation

The observation model is mainly obtained by matching the features extracted by the LiDAR to the point cloud and is defined as:

$$\delta \hat{y}(t) = f(\bar{x}_{b_{k+1}}^{b_k} \oplus \delta x) (J_k M_k J_k^T)^{-1} \quad (17)$$

where J_k is the measurement noise of $f(\cdot)$, and M_k is the covariance matrix of the measurement noise. $f(\cdot)$ represents a stacked residual vector computed from point-to-edge pairs or point-to-plane pairs.

Given the error term in $x_{b_{k+1}}^{b_k}$, $f(\cdot)$ corresponds to $\mathbf{p}_i^{l_{k+1}}$ and the i th characteristic point in the LiDAR coordinate $k+1$ frames, and the observation equation can be described as

$$f_i(\mathbf{x}_{b_{k+1}}^{b_k}) = \begin{cases} \frac{\left| \left(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k} \right) \times \left(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_b^{l_k} \right) \right|}{\left| \mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k} \right|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_e \\ \frac{\left| \left(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k} \right)^T \left(\left(\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k} \right) \times \left(\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k} \right) \right) \right|}{\left| \left(\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k} \right) \times \left(\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k} \right) \right|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_p \end{cases} \quad (18)$$

$$\hat{\mathbf{p}}_i^{l_k} = \mathbf{R}_l^{b^T} \left(\mathbf{R}_{b_{k+1}}^{b_k} \left(\mathbf{R}_l^b \mathbf{p}_i^{l_{k+1}} + \mathbf{p}_l^b \right) + \mathbf{p}_{b_{k+1}}^{b_k} - \mathbf{p}_l^b \right) \quad (19)$$

where $\hat{\mathbf{p}}_i^{l_k}$ is the pose transformation of the point $\mathbf{p}_i^{l_{k+1}}$ from the $k+1$ th frame to the k th frame. For edge points, the distance is between point $\mathbf{p}_i^{l_{k+1}}$ and the line formed by $\mathbf{p}_a^{l_k} \mathbf{p}_b^{l_k}$. For planar points, the distance is between point $\mathbf{p}_i^{l_{k+1}}$ and the plane formed by $\mathbf{p}_a^{l_k} \mathbf{p}_b^{l_k} \mathbf{p}_c^{l_k}$.

5.4. Status Update

In this paper, an iterative extended Kalman filter (IEKF) is used and the state update can be abstracted as an optimization problem. Because the error state is composed of the deviation of the prior state and the residual of the observation model, the error state update problem can be turned into a joint optimization problem of solving for the minimum of the deviation of the prior state and the minimum of the deviation of the observation model relative to the pose solution, which can be expressed as:

$$r_S(x) = \min \|\delta x\|_{P(k)^{-1}} + \left\| f\left(\bar{x}_{b_{k+1}}^{b_k} \oplus \delta x\right) \right\|_{(J_k M_k J_k^T)^{-1}} \quad (20)$$

where $\|\cdot\|$ denotes the M-norm.

Thus, the Kalman gain equation for the ESKF iterative observation can be expressed as Equation (21), and the mean value of the posterior state can be expressed as Equation (22):

$$K_{k,j} = P_k H_{k,j}^T \left(H_{k,j} P_k H_{k,j}^T + J_{k,j} M_k J_{k,j}^T \right)^{-1} \quad (21)$$

$$\delta x_{j+1} = \delta x_j + K_{k,j} \left(H_{k,j} \delta x_j - f\left(\bar{x}_{b_{k+1}}^{b_k} \oplus \delta x_{j+1}\right) \right) \quad (22)$$

where $H_{k,j}$ is the Jacobi matrix of $f(\bar{x}_{b_{k+1}}^{b_k} \oplus \delta x_{j+1})$ and Δx_j represents the correction vector at the j th iteration. Note that in each iteration, new matching edges and matching surfaces are found to further minimize the metric error, so $H_{k,j}$, $J_{k,j}$ and $K_{k,j}$ are recomputed.

When $f(\mathbf{x}_{b_{k+1}}^{b_k})$ is below a certain threshold, P_k is updated using Equation (23), which can be obtained after the n th iteration, and the variance of the posterior state can be expressed as:

$$P_{k+1} = (I - K_{k,n}H_{k,n})P_k(I - K_{k,n}H_{k,n})^T + K_{k,n}M_kK_{k,n}^T \quad (23)$$

By iterative updates, the exact posterior state $x_{b_{k+1}}^{b_k}$ can be obtained, and this information can be used as the prior state at the next moment, which is in turn set as the prior state at the next moment.

5.5. State Composition

The positional information was calculated earlier under the robot-centered reference coordinate. Therefore, to obtain global coordinate information, state synthesis is needed, and the synthesis formula can be expressed as:

$$x_w^{b_{k+1}} = \begin{bmatrix} p_w^{b_{k+1}} \\ q_w^{b_{k+1}} \end{bmatrix} = \begin{bmatrix} R_{b_k}^{b_{k+1}}(p_w^{b_k} - p_{b_{k+1}}^{b_k}) \\ q_{b_k}^{b_{k+1}} \otimes q_w^{b_k} \end{bmatrix} \quad (24)$$

6. Back End: Pose Graph Optimization

6.1. Initialization

LiDAR and IMU sensors are used, where the acceleration deviation b_{a_0} of the IMU and the deviation b_{ω_0} of the gyroscope can be obtained by offline calibration. The external calibration matrix R_l^b, p_l^b between the LiDAR and the IMU can be obtained by offline calibration. The initial roll and pitch of the global poses are obtained by unbiased acceleration measurements before the motion. The initial local gravity is obtained from the initial roll and pitch by converting the gravity vector represented in the navigation frame to the current local frame.

6.2. Keyframe Selection

To keep a limited number of poses for estimation, on the back end, we deploy a keyframe selection strategy. The selection of keyframes can greatly improve the computational efficiency and ensure that the algorithm can run in real time on larger environmental maps.

When selecting keyframes, matching errors and redundant keyframes must be reduced to decrease the computational effort, while sparse keyframes increase the uncertainty of interframe observations and degrade the map quality. Usually, the first point cloud is used as the keyframe, and the keyframe selection criteria based on the real-time point cloud are based on the relative positional change between consecutive frames. Let the pose of the $k-1$ th keyframe be T_{k-1} and the pose of the k th keyframe be T_k [38]. The relative positional change of consecutive keyframes can be obtained using Equation (25):

$$\Delta T_{k-1,k} = T_{k-1}^{-1} T_k = \begin{pmatrix} \Delta \mathbf{r} & \Delta s \\ 0 & 1 \end{pmatrix} \quad (25)$$

where the rotation of the change between two consecutive keyframes can be expressed as ΔR , the translation of the change between two consecutive keyframes can be expressed as Δs and the time of the transformation between two consecutive keyframes can be expressed as Δt .

$$\Delta R = \arccos\left(\frac{\text{trace}(\Delta \mathbf{r}) - 1}{2}\right) \quad (26)$$

$$\|\Delta s\| = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \quad (27)$$

When any of the above three criteria exceed the set threshold, the current k th frame of the point cloud is selected as a keyframe. The other LiDAR frames between the two

keyframes are discarded. Adding keyframes in this way achieves not only a balance between map density and memory consumption but also a uniform distribution in space, which is suitable for real-time nonlinear optimization.

6.3. Mapping

When a point cloud frame is detected as a keyframe, the environment map of the current frame needs to be updated, and the main method is to convert the real-time point cloud information of the keyframe to the world coordinates by coordinate transformation [39]. When $k + 1$ frames are selected as sub-keyframes, continuous sub-keyframe optimization is performed using ICP and the relative change in pose between sub-keyframes is T_{k+1}^L . By the relative change of the k th sub-keyframe to the world coordinates as T_k^W , the relative change of the $k+1$ th sub-keyframe to the world coordinate can be obtained as $T_{k+1}^W = T_k^W T_{k+1}^L$. Finally, the point cloud coordinates of the $k + 1$ sub-keyframes are converted into the world coordinates, and the converted sub-keyframes are merged into a voxel mapping M_i . Because two types of features are extracted in the previous feature extraction step, M_i consists of two sub-voxel maps, that is, the edge feature voxel map set M_i^e and the planar feature voxel map set M_i^p . The interrelationship between the LiDAR frames and the voxel maps is as follows:

$$M_i = \{M_i^e, M_i^p\} \quad (28)$$

where $M_i^e = \mathcal{F}_i^e \cup \mathcal{F}_{i-1}^e \cup \dots \cup \mathcal{F}_{i-n}^e$, $M_i^p = \mathcal{F}_i^p \cup \mathcal{F}_{i-1}^p \cup \dots \cup \mathcal{F}_{i-n}^p$. $\mathcal{F}_i^e$ and $\mathcal{F}_i^p$ are the edge line and plane features, respectively, extracted by the desired transformation sub-keyframe that are down-sampled to eliminate duplicate features that fall in the same pixel cell, while the point cloud of the current sub-keyframe is created as a local map and updated by an octree structure.

6.4. Loop Detection

To reduce the cumulative error of the sensor, we add loop detection. When the robot detects all historical keyframes during operation, pose graph optimization is combined with all relevant positions and poses in the historical keyframes. When performing loop detection, the current real-time keyframe is first compared with the historical keyframe, and then the candidate loop frame is selected. The following conditions are met: (1) the index of the current keyframe is larger than the index of the history keyframe; (2) the difference between the track distance of the current keyframe and the history keyframe is larger than the set threshold; and (3) the relative translation distance between the current keyframe and the history keyframe is smaller than the set threshold. Finally, the candidate loop frame with the highest score is selected by comparing the candidate loop frames and the currently registered keyframes. The keyframes and candidate loop frames are added as nodes to the SLAM pose graph optimization, and the edge constraints are the relative poses obtained from point cloud registration.

The loop detection algorithm [40] is described in Algorithm 1. Suppose the current frame point cloud is represented as P_m in the B coordinates and its relative pose change to the world coordinates is T_m . If the keyframe database contains the keyframe pose set D_T and the point cloud set D_P , then a kd tree search with radius r is performed. Therefore, the problem to be solved is to show that the loop has been detected if we can find T'_m in the keyframe pose set D_T such that the sum of the root mean square error of the point cloud P_m of the current frame after the relative change T_{ICP} with the nearest point in D_P is minimized. The keyframe pose D_T registers the keyframe point cloud D_P to the local point cloud map M constructed in the world coordinate space.

Algorithm 1 Loop Detection Algorithm.Input: P_m and T_m from the current frame point cloudOutput: T_{ICP}

```

1: if  $(T_m, P_m)$  is Keyframe() then
2:   if  $D_T \neq \emptyset$  or  $D_P \neq \emptyset$  then
3:      $T'_m \leftarrow \text{KDtree.RadiusSearch}(T_m, D_T, r)$ ;
4:      $M \leftarrow \text{registerPointCloud}(D_T, D_P)$ ;
5:     if  $T'_m \neq 0$  then
6:        $T_{ICP} \leftarrow \text{ComputeICP}(P_m, M, T_m)$ ;
7:     end if
8:   end if
9:    $T'_m \leftarrow 0$ 
10:   $D_T = D_T \cup T_m$  and  $D_P = D_P \cup P_m$ 
11: end if

```

Only the current pose of the point cloud in the world coordinates for the current frame is obtained, and T_{ICP} and M are fed to the pose graph optimization for state correction. The state correction can be expressed as a modification of Equation (29) by solving the following cost function:

$$r_M(\hat{z}_{(j,m)}, \chi) = \bar{P}_j^w - \bar{P}_m^w = R_{b_k}^w (R_l^b \bar{P}_j^l + p_l^b) + p_{b_k}^w - \bar{P}_m^w \quad (29)$$

where $\bar{P}_m^w \in M$ is the closest point to $\bar{P}_j^w$ in the global map. By solving the modified cost function, the current state can be used for repositioning in the global map.

6.5. Pose Graph Optimization Construction

To effectively solve the large-scale simultaneous localization and map building problem, the method of cluster adjustment (BA) is often used; that is, the map optimization uses sensor poses and spatial points. However, as the trajectory of the robot increases, the BA method decreases the computational efficiency of the system as the size of the map continues to increase. To address this issue, the pose graph optimization only uses the trajectory as a constraint for the pose estimation and no longer optimizes the pose estimation of the feature points. Based on bitmap optimization theory, the keyframe pose, ground constraint and loop detection constraint are calculated to construct a multi-constraint-based cost function for bitmap optimization.

The pose graph optimization problem can be efficiently solved by standard optimization methods such as the Gauss–Newton or Levenberg–Marquardt (LM) algorithms [41]. This problem has also been integrated into the Generalized Graph Optimization (G2O) of the Optimization Library [42]. Our goal is to estimate the 6DOF self-motion of the robot and simultaneously build a globally consistent map. Assuming that these measurement errors conform to a zero-mean Gaussian distribution, solving the maximum a posteriori (MAP) problem is equivalent to minimizing the negative log-likelihood, which can be written in the form of the squared Marxian distance:

$$\hat{X}_n = \text{argmin} \|r_S(x)\| + \sum \|e_{i,m}\| + \sum \|r_M(\hat{z}_{(j,m)}, \chi)\| \quad (30)$$

where $r_S(x)$ denotes the deviation of the error state estimated by LIO, $e_{i,m}$ denotes the error between the attitude node and the ground plane node and $r_M(\hat{z}_{(j,m)}, \chi)$ denotes the deviation of the current state of the loop detection in the global map.

7. Tests and Analyses

We conducted a series of quantitative and qualitative analysis experiments on the performance of the proposed tightly coupled LiDAR-IMU fusion SLAM algorithm and compared the results with those of other state-of-the-art LiDAR-SLAM methods. All

methods were tested under the same conditions. The hardware platform was an inspection robot with sensors and an on-board computer, as shown in Figure 2. The sensor consisted of a LiDAR (Velodyne VLP-16) with a sampling frequency of 10 Hz and an IMU (hipnuc-CH110) with a sampling frequency of 200 Hz. The on-board computer was an Intel Core i7 with a 2.7 GHz clock, eight cores and 16 G of RAM. All algorithms were implemented in C++ and executed on an Ubuntu 18.04 system using the medoic version of ROS.



Figure 2. Wheeled inspection robot.

7.1. Implementation and Datasets

Since accurate true values are not available for comparison for perceptually degraded scenes, in order to validate the performance of our proposed method (including accuracy, cumulative error and robustness) while ensuring the experimental soundness, we first used public dataset 1 and public dataset 2 for quantitative analysis. Note that all algorithms were tested without the loop module. To verify whether our proposed method causes the accuracy of the SLAM system to decrease as the speed of motion changes, we used the public dataset 1, which is a six-set fast-to-slow dataset comparing rotation and translation errors, whose true values are provided by the indoor high-precision motion capture instrument. To verify whether our proposed method increases the cumulative error of the system as it operates for a long time, we used public dataset 2, which is the 2011_09_30_drive_0028 sequence of the KITTI datasets, whose true values are provided by outdoor high-precision GPS. Subsequently, to verify whether our proposed method is robust under three different real perceptual degradation environments, we recorded three self-picked datasets with our own experimental platform, and the self-picked datasets were qualitatively analyzed for real promenade environments under three perceptual degradation scenes. Furthermore, we compared the method with the a_loam, lego_loam and lio_sam methods. We verified the excellent robustness of our proposed algorithm through horizontal comparison between different algorithms and vertical comparison between different scenes. In addition, we designed ablation experiments to verify the effects of adding ground constraints and loop detection constraints on the overall optimization results. Finally, a runtime analysis was performed for the main modules.

7.2. Quantitative Analysis

7.2.1. Tests at Different Movement Speeds

Table 2 shows the root mean square error (RMSE) results for different speeds and methods. LOAM [1] was considered as the baseline. LIO represents the local window optimized odometry method, LIO-raw is the result obtained by removing motion compensation on top of LIO and LIO-no-ext is the result obtained by removing online external parameter

estimation on top of LIO. The “ours” column represents the proposed LIO generated by the iterative Kalman filter. The best results are shown in bold.

Table 2. Rotation and translation errors for comparing true values in the following six fast-to-slow datasets.

Error	Dataset	LOAM	LIO-Raw	LIO-No-Ext	LIO	Ours
Root mean square error of translation (m)	Fast_1	0.4469	0.2464	0.0957	0.0949	0.0700
	Fast_2	0.2023	0.4346	0.1210	0.0755	0.0683
	Med_1	0.1740	0.1413	0.1677	0.1002	0.0577
	Med_2	0.1010	0.2460	0.3032	0.1308	0.0647
	Slow_1	0.0606	0.1014	0.0838	0.0725	0.0248
	Slow_2	0.0666	0.1016	0.0868	0.1024	0.0310
Root mean square error of rotation (rad)	Fast_1	0.1104	0.1123	0.0547	0.0545	0.0283
	Fast_2	0.0763	0.1063	0.0784	0.0581	0.0277
	Med_1	0.0724	0.0620	0.0596	0.0570	0.0198
	Med_2	0.0617	0.0886	0.0900	0.0557	0.0287
	Slow_1	0.0558	0.0672	0.0572	0.0581	0.0309
	Slow_2	0.0614	0.0584	0.0571	0.0533	0.0284

The results show that our proposed algorithm always provided more accurate translational and rotational states for both fast and slow motions. The table also shows that, with motion compensation and online external reference estimation, LIO provided a better performance, especially in the case of fast motion, probably because more IMU excitation had to be generated. The proposed method, in contrast, was not affected by the speed of motion and performed better at any speed of motion relative to LIO.

7.2.2. Deviation Test with Long Time Change

To evaluate how the error varied over time, we used the 2011_09_30_drive_0028 sequence from the KITTI datasets, with the four different algorithms producing roughly the same motion trajectory but with subtle differences in some areas. To explore the differences between the algorithms, the EVO tool was used for quantitative analysis. The final motion trajectories of the different methods are shown in Figure 3 and the trajectory error and the true value produced by the four different methods are shown in Figure 4.

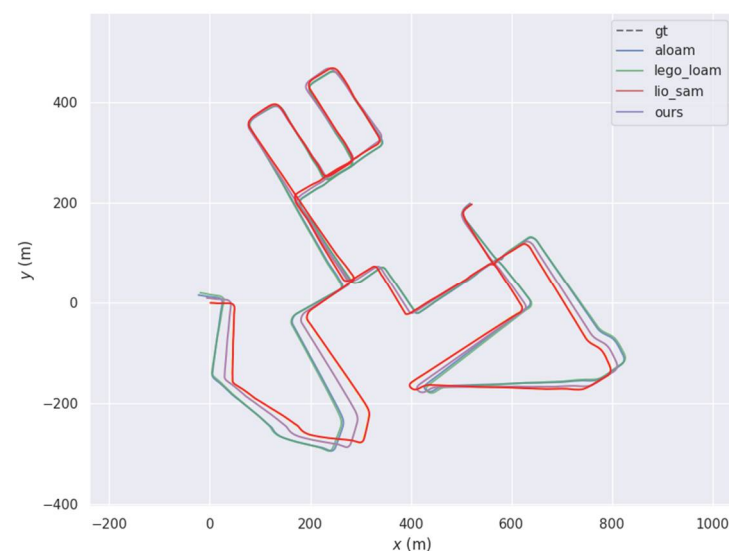


Figure 3. Comparison between the motion trajectories generated by the four different algorithms and the true values.

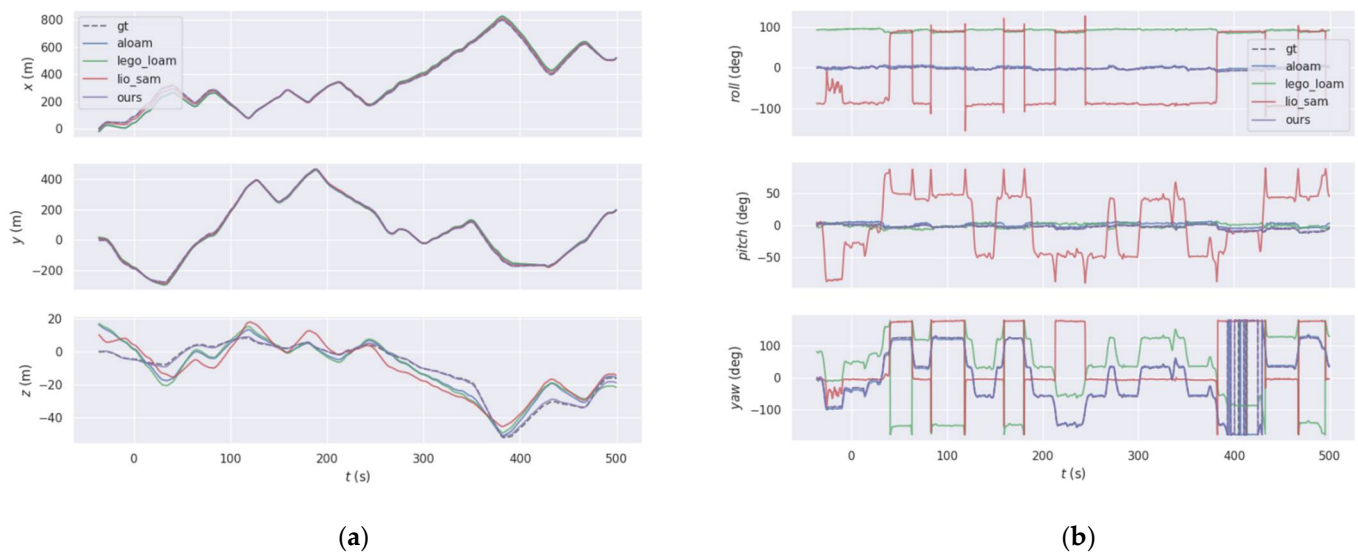


Figure 4. Comparison between the trajectory error and the true value produced by the four different, they should be listed as: (a) translation error; (b) rotation error.

To further refine the comparison experiments, we analyzed the maximum deviation, mean deviation, minimum deviation, root mean square error and standard deviation of the KITTI datasets 2011_09_30_drive_0028 sequence for the four different algorithms for each relative translation and rotation, as shown in Table 3. The best results are shown in bold.

Table 3. Error statistics of relative translations and rotations between the four algorithms and the true values.

Algorithm	Aloam	Lego_Loam	Lio_Sam	Ours	Aloam	Lego_Loam	Lio_Sam	Ours
Error	Translation (m)				Rotation (deg)			
Max	1.2899	2.5120	4.7334	0.0814	2.9888	6.3016	3.2062	0.6162
Mean	0.1015	1.1715	1.5112	0.0105	0.2889	0.9832	0.2777	0.0381
Min	0.0032	0.0092	0.0066	0.0018	0.0027	0.0073	0.0039	0.0168
RMSE	0.1175	1.2414	1.7136	0.0154	0.3678	1.5164	0.3667	0.0641
Std	0.0591	0.4108	0.8079	0.0112	0.2277	1.1545	0.2395	0.0515

The statistical results show that the proposed algorithm had smaller translation and rotation errors than the other three algorithms when tested on the same datasets. Therefore, the accuracy of the state estimation of our proposed algorithm is better than that of the other methods.

7.3. Qualitative Analysis

We conducted three experiments with the same sensor configuration in different test environments to verify the performance of the proposed algorithm for perceptually degraded scenes. The datasets included corridor scene 1 on the 3rd floor of the mechanical building, corridor scene 2 on the 21st floor of the innovation building and corridor scene 3 on the 11th floor of the science and technology building. The proposed algorithm was analyzed and compared with the aloam, lego_loam and lio_sam methods, and its point cloud map was constructed in the current experimental environment, as shown in Figure 5.

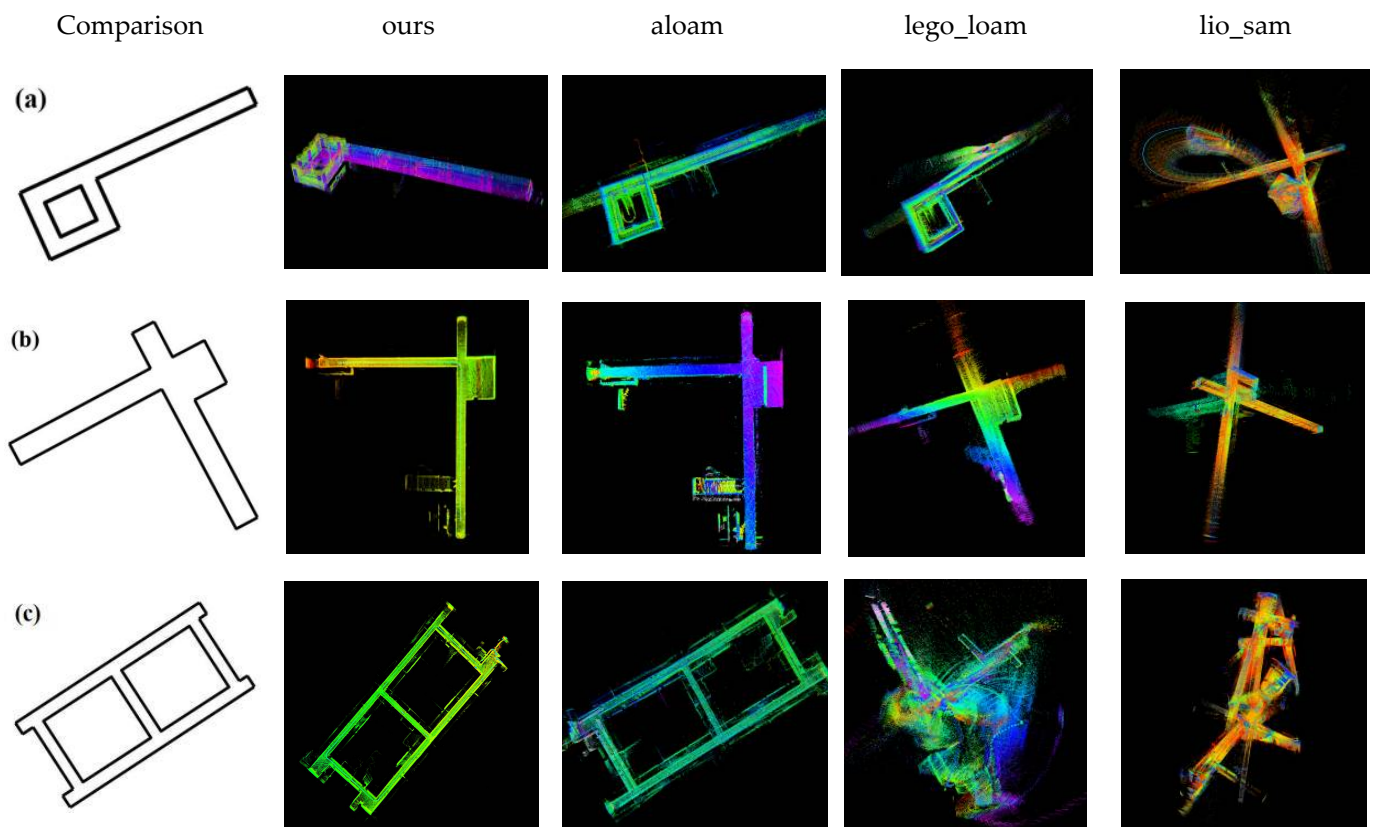


Figure 5. The point cloud map was constructed by four SLAM methods in the current experimental environment, they should be listed as: (a) Corridor scene on the 3rd floor of the mechanical building; (b) corridor scene on the 21st floor of the innovation building; (c) corridor scene on the 11th floor of the science and technology building.

As shown in Figure 5, four different SLAM algorithms were used to construct point cloud maps under three perceptually degraded scenes. The differences between the above three different scenes are: Figure 5a,b scenes do not differ much in form, only in specific details, and the “8-letter” structure of Figure 5c is a loop detection structure. The similarity is that since all three test scenes are all perceptually degraded scenes, their point clouds have relatively few features, thus posing a great challenge to the proposed algorithm. For the four different algorithms, the point cloud maps constructed by the proposed method showed an excellent performance in terms of completeness and the realism of the geometric structure in all three scenarios. While aloam also performed well in all three scenes, a small amount of mismatching of the point cloud was caused by the occurrence of successive similar rotations and translations in scene a. Lego_loam constructed a complete point cloud map for the rough outline of the corridor in both scenarios a and b. However, the structure of the generated point cloud map was distorted and incomplete due to the drift on the Z-axis. In the c scene, the structural symmetry in the annular scene was too similar, resulting in interference of the extracted features of the original point cloud data by a large number of similar point cloud features, and the entire SLAM system failed. Lio_sam constructed the geometric structure of the surrounding environment in a short period of time for scenarios a, b and c. However, as the mileage of the laser odometer increased, when the system detected the loop, the point cloud map construction failed because the scenarios were highly similar, resulting in the establishment of an incorrect loop detection constraint.

In conclusion, the four different algorithms build different point cloud maps in the same environment because of their strengths and weaknesses. The scheme of aloam adopts the assumption of uniform motion model and uses IMU data for distortion correction of point clouds. The system has good real-time performance, but there is no loop detection

module, which causes it to drift in large-scale tests. The *lego_loam* method is an optimized version of *loam* which segments and clusters the raw data frames scanned by LiDAR, labels the ground point cloud, and improves the accuracy of state estimation by two-step LM optimization. However, because it relies too much on the extraction of point surface point clouds, once the ground point cloud extraction fails, the state estimation error will increase. *Lio_sam* is an extended version of *lego_loam* which solves the least-squares problem by continuously adding various factors. However, loop detection requires a high correlation of correct data, and incorrect matching can lead to the failure of the whole SLAM system. The method we propose incorporates the advantages of all the above algorithms. First, the front end adopts the ideas of homogeneous model assumption from *loam* and extraction of ground features from *lego_loam*, respectively. However, we improve the Kalman filter model in LINS by the RANSAC method for point cloud ground extraction and propose an iterative Kalman filter model to eliminate the error caused by mismatching in order to achieve a tightly coupled LiDAR-inertial odometry. Secondly, the back end draws on the idea of factor graph optimization in *lio_sam* to construct a cost function for bit-pose graph optimization by adding ground constraints and loop detection constraints through the selection strategy of keyframes and further improve the accuracy of state estimation by solving the maximum a posteriori problem. Therefore, our proposed method performs well under three different perceptual degradation environments.

7.4. Ablation Experiments

To further analyze the localization and map-building effects of the proposed algorithm, the effects of the ground constraint and loop detection back end pose graph optimization modules were analyzed separately.

7.4.1. Effect of Ground Restraint

The results of the experimental analysis show that the ground constraint of the back end pose graph optimization in the system had a great influence on the map building results, as shown in Figure 6. Figure 6a shows that without adding ground constraints in the back end pose graph optimization, the constructed point cloud map could not be corrected, which tilted it to a certain angle. Figure 6b shows the addition of the ground constraint in the back end pose graph optimization to correct the Z-axis drift during the positioning and map-building process and the construction of a globally consistent point cloud map.

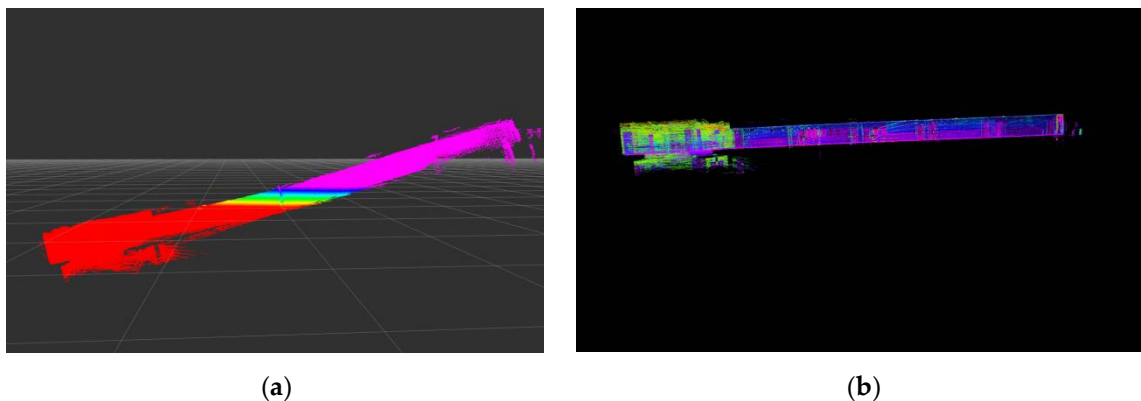


Figure 6. Effect of ground constraints on positioning and map building, they should be listed as: (a) Without ground constraint, (b) with ground constraint.

7.4.2. Effect of Loop Detection

The loop detection of the back end pose graph optimization in the system had a substantial impact on the construction of the map, as shown in Figure 7. Figure 7a shows that no loop constraint was used in the back end pose graph optimization, which led

to a large deviation in the constructed point cloud map, such that part of the map was overshadowed in some areas in the upper left corner. Figure 7b shows the addition of the loop constraint in the back end pose graph optimization. The cumulative error of the global map was corrected, and finally, a globally consistent point cloud map was constructed.

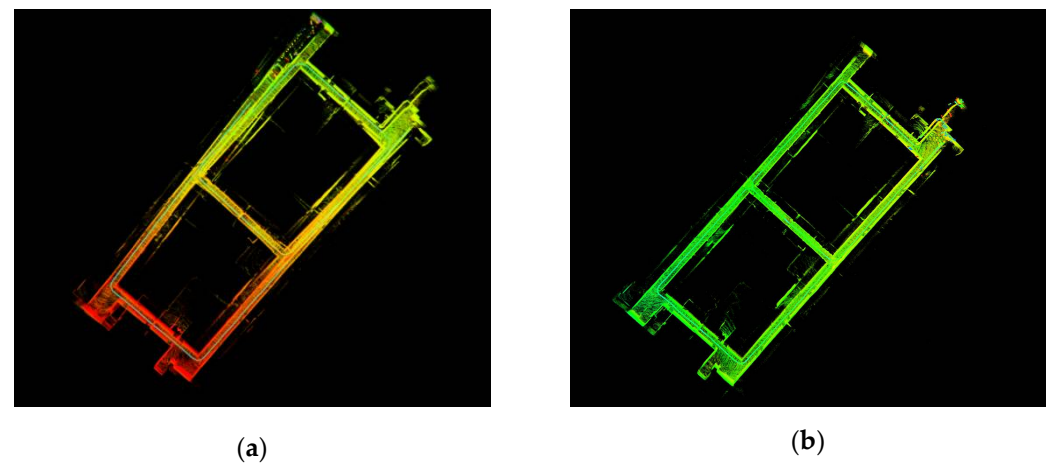


Figure 7. Effect of loop detection on localization and map building: (a) Without loop detection and (b) with loop detection.

7.5. Runtime Analysis

To explore the real-time performance of the proposed algorithm, we conducted experiments on the self-selected dataset and the KITTI dataset and analyzed the running time of the main modules for the down-sampling, pre-processing and scan-matching front end steps; on the back end, the running time was analyzed for the map construction, loop detection and pose graph optimization steps. The details are shown in Table 4. As the table shows, because our self-selected dataset consisted of VLP-16 data and the KITTI dataset consisted of HDL-64 data, more time was required to process the individual modules of the KITTI dataset with the same sampling frequency for the LiDAR-SLAM system. We analyzed the self-selected data, and for the front end LiDAR point cloud and IMU data pre-processing, the average time per frame was only approximately 10 ms and no large deviations were observed for the entire process. For LIO, scan matching required approximately 3 ms, and iterative matching required up to 10 ms for convergence. The construction of the local map required approximately 2 ms. Because loop detection requires circular judgment and ICP-based alignment between loop frames for pose graph optimization, the average loop detection time was approximately 35 ms, and the average time for pose graph optimization was approximately 13 ms, which quickly satisfied the convergence condition.

Table 4. Average running time statistics for each major module.

Submodule Name		Self-Picked Datasets (ms)	KITTI Dataset (ms)
Front End	down sampling	4.14	14.13
	pre-processing	2.89	8.68
	scan matching	3.45	16.21
Back End	map construction	2.19	4.74
	loop detection	34.58	125.35
	pose graph	13.41	118.62
	optimization		

Combined with the system framework diagram (Figure 1), we can see that we tightly couple the 200 hz operating frequency IMU pre-integration with the 10 hz operating frequency LiDAR and synchronize the 10 hz LiDAR-inertial odometry output to the back

end optimization module through the tight coupling. While the front end and back end of our constructed LiDAR-IMU fusion SLAM system work as two independent threads, the sum of each frame of data processing time of each front end and back end module can meet the system's real-time requirements as long as it is less than 100 ms (10 hz).

8. Conclusions

- (1) A tightly coupled LiDAR-IMU fusion SLAM algorithm for perceptually degraded scenes is proposed. The robustness for such special scenarios is enhanced by constructing an iterative Kalman filter at the front end and optimizing the pose map at the back end. By adding loop detection and ground constraints to the tightly coupled frame, the relative poses between adjacent keyframes are optimized, the cumulative sensor errors are reduced, the accuracy of the system is improved and global map consistency is ensured.
- (2) The algorithm was quantitatively and qualitatively analyzed on public and self-selected datasets and compared with existing LiDAR-SLAM algorithms. The results show that the algorithm was able to achieve robust 6DOF state estimation and high-performance SLAM in the test environments and output motion trajectories, global maps and state estimation after pose graph optimization.
- (3) Ablation experiments were performed on the algorithm in terms of ground constraint and loop detection to determine their impacts on the entire system and to explain the rationality of the algorithm. The running time of each major module on the front end and back end of the system was statistically analyzed for the different datasets, and the proposed algorithm was able to meet the real-time performance requirements.
- (4) In subsequent work, research on multi-sensor fusion based on LiDAR, vision and IMU will be conducted in conjunction with other practical scenarios of perceptual degradation to further improve the positioning accuracy and robustness of the system. In addition, the structure of the stored point cloud map data, which is the basis for the map-based positioning and navigation of the robot, was studied and is the key to ensuring real-time performance.

Author Contributions: L.Y. and Y.W. conceived and designed the experiments; H.M. and L.Y. guided the design of the system algorithm; L.Y. wrote the paper, and performed the experiments; J.X. made the result analysis; C.W. revised the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under grant nos. 51975468 and 50674075.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Smith, R.; Self, M.; Cheeseman, P. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1990; pp. 167–193.
2. Anderson, S.; Barfoot, T.D. RANSAC for motion-distorted 3D visual sensors. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2093–2099.
3. Behley, J.; Stachniss, C. Efficient Surlfel-Based SLAM using 3D Laser Range Data in Urban Environments. *Robot. Sci. Syst.* **2018**, *2018*, 59.
4. Sun, K.; Mohta, K.; Pfrommer, B.; Watterson, M.; Liu, S.; Mulgaonkar, Y.; Taylor, C.J.; Kumar, V. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robot. Autom. Lett.* **2018**, *3*, 965–972. [[CrossRef](#)]
5. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [[CrossRef](#)]

6. Qin, C.; Ye, H.; Pranata, C.E.; Han, J.; Zhang, S.; Liu, M. Lins: A lidar-inertial state estimator for robust and efficient navigation. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 8899–8906.
7. Ye, H.; Chen, Y.; Liu, M. Tightly coupled 3d lidar inertial odometry and mapping. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 3144–3150.
8. Barfoot, T.D.; McManus, C.; Anderson, S. Into darkness: Visual navigation based on a lidar-intensity-image pipeline. In *Robotics Research*; Springer: Cham, Switzerland, 2016; pp. 487–504.
9. He, G.; Yuan, X.; Zhuang, Y.; Hu, H. An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–9. [\[CrossRef\]](#)
10. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. Sensor fusion IV: Control paradigms and data structures. In Proceedings of the Sensor Fusion IV: Control Paradigms and Data Structures, Boston, MA, USA, 14–15 November 1992; 1611, pp. 586–606.
11. Segal, A.; Haehnel, D.; Thrun, S. Generalized-ICP. In *Robotics: Science and Systems*; University of Washington: Seattle, WA, USA, 2009.
12. Grant, W.S.; Voorhies, R.C.; Itti, L. Finding planes in LiDAR point clouds for real-time registration. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4347–4354.
13. Zhang, J.; Singh, S. Low-drift and Real-time Lidar Odometry and Mapping. *Auton. Robots* **2017**, *41*, 401–416. [\[CrossRef\]](#)
14. Shan, T.; Englot, B. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765.
15. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to MAV navigation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3923–3929.
16. Yang, S.; Zhu, X.; Nian, X.; Feng, L.; Qu, X.; Ma, T. A robust pose graph approach for city scale LiDAR mapping. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1175–1182.
17. Demir, M.; Fujimura, K. Robust localization with low-mounted multiple lidars in urban environments. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 3288–3293.
18. Gao, Y.; Liu, S.; Atia, M.M.; Noureldin, A. INS/GPS/LiDAR integrated navigation system for urban and indoor environments using hybrid scan matching algorithm. *Sensors* **2015**, *15*, 23286–23302. [\[CrossRef\]](#)
19. Hening, S.; Ippolito, C.A.; Krishnakumar, K.S.; Stepanyan, V.; Teodorescu, M. *3D LiDAR SLAM Integration with GPS/INS for UAVs in Urban GPS-Degraded Environments*; AIAA Information Systems-AIAA Infotech@ Aerospace: Grapevine, TX, USA, 2017; p. 0448.
20. Zhang, S.; Guo, Y.; Zhu, Q.; Liu, Z. Lidar-IMU and wheel odometer based autonomous vehicle localization system. In Proceedings of the 2019 Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 4950–4955.
21. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
22. Li, M.; Kim, B.H.; Mourikis, A.I. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 4712–4719.
23. Huai, Z.; Huang, G. Robocentric visual-inertial odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 6319–6326.
24. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 5135–5142.
25. Huang, G.P.; Mourikis, A.I.; Roumeliotis, S.I. A quadratic-complexity observability-constrained unscented Kalman filter for SLAM. *IEEE Trans. Robot.* **2013**, *29*, 1226–1243. [\[CrossRef\]](#)
26. Qin, C.; Ye, H.; Pranata, C.E.; Han, J.; Zhang, S.; Liu, M. R-lins: A robocentric lidar-inertial state estimator for robust and efficient navigation. *arXiv* **2019**, arXiv:1907.02233.
27. Xu, W.; Zhang, F. Fast-lío: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3317–3324. [\[CrossRef\]](#)
28. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. Fast-lío2: Fast direct lidar-inertial odometry. *arXiv* **2021**, arXiv:2107.06829. [\[CrossRef\]](#)
29. Huang, S.; Dissanayake, G. Convergence and consistency analysis for extended Kalman filter based SLAM. *IEEE Trans. Robot.* **2007**, *23*, 1036–1049. [\[CrossRef\]](#)
30. Bell, B.M.; Cathey, F.W. The iterated Kalman filter update as a Gauss-Newton method. *IEEE Trans. Autom. Control* **1993**, *38*, 294–297. [\[CrossRef\]](#)
31. Sola, J. Quaternion kinematics for the error-state Kalman filter. *arXiv* **2017**, arXiv:1711.02508.
32. Geneva, P.; Eckenhoff, K.; Yang, Y.; Huang, G. Lips: Lidar-inertial 3D plane slam. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 123–130.

33. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [[CrossRef](#)]
34. Ma, L.; Kerl, C.; Stückler, J.; Cremers, D. CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1285–1291.
35. Civera, J.; Grasa, O.G.; Davison, A.J.; Montiel, J.M.M. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.* **2010**, *27*, 609–631. [[CrossRef](#)]
36. Bloesch, M.; Burri, M.; Omari, S.; Hutter, M.; Siegwart, R. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *Int. J. Robot. Res.* **2017**, *36*, 1053–1072. [[CrossRef](#)]
37. Lupton, T.; Sukkariyeh, S. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robot.* **2011**, *28*, 61–76. [[CrossRef](#)]
38. Ren, Z.; Wang, L.; Bi, L. Robust GICP-based 3D LiDAR SLAM for underground mining environment. *Sensors* **2019**, *19*, 2915. [[CrossRef](#)]
39. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [[CrossRef](#)]
40. Yang, J.-C.; Lin, C.-J.; You, B.-Y.; Yan, Y.-L.; Cheng, T.-H. Rtlío: Real-time lidar-inertial odometry and mapping for UAVS. *Sensors* **2021**, *21*, 3955. [[CrossRef](#)] [[PubMed](#)]
41. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*; Cambridge University Press: Cambridge, UK, 2003.
42. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. g2o: A general framework for graph optimization. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3607–3613.