sensors

MDPI

Article

RTLIO: Real-Time LiDAR-Inertial Odometry and Mapping for UAVs

Jung-Cheng Yang, Chun-Jung Lin, Bing-Yuan You, Yin-Long Yan and Teng-Hu Cheng *

Department of Mechanical Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; johnsongash.gdr07g@nctu.edu.tw (J.-C.Y.); chadlin.gdr07g@nctu.edu.tw (C.-J.L.); physical31031.c@nycu.edu.tw (B.-Y.Y.); yanlong658.gdr08g@nctu.edu.tw (Y.-L.Y.)
* Correspondence: tenghu@g2.nctu.edu.tw

Abstract: Most UAVs rely on GPS for localization in an outdoor environment. However, in GPS-denied environment, other sources of localization are required for UAVs to conduct feedback control and navigation. LiDAR has been used for indoor localization, but the sampling rate is usually too low for feedback control of UAVs. To compensate this drawback, IMU sensors are usually fused to generate high-frequency odometry, with only few extra computation resources. To achieve this goal, a real-time LiDAR inertial odometer system (RTLIO) is developed in this work to generate high-precision and high-frequency odometry for the feedback control of UAVs in an indoor environment, and this is achieved by solving cost functions that consist of the LiDAR and IMU residuals. Compared to the traditional LIO approach, the initialization process of the developed RTLIO can be achieved, even when the device is stationary. To further reduce the accumulated pose errors, loop closure and pose-graph optimization are also developed in RTLIO. To demonstrate the efficacy of the developed RTLIO, experiments with long-range trajectory are conducted, and the results indicate that the RTLIO can outperform LIO with a smaller drift. Experiments with odometry benchmark dataset (i.e., KITTI) are also conducted to compare the performance with other methods, and the results show that the RTLIO can outperform ALOAM and LOAM in terms of exhibiting a smaller time delay and greater position accuracy.

Keywords: LiDAR-inertial odometry; state estimation; sensor fusion; SLAM

Citation: Yang, J.-C.; Lin, C.-J.; You, B.-Y.; Yan, Y.-L.; Cheng, T.-H. RTLIO: Real-Time LiDAR-Inertial Odometry and Mapping for UAVs. *Sensors* **2021**, *21*, 3955. https://doi.org/10.3390/s21123955

Academic Editors: Kamil Krasuski and Damian Wierzbicki

Received: 9 May 2021
Accepted: 4 June 2021
Published: 8 June 2021

1. Introduction

1.1. Background

Precise ego-motion estimation and active perception play important roles when performing navigation tasks or exploring unknown environments in robotics applications, and the potential of small unmanned airborne (S-UAS) platforms applied to collect remote sensing data have been analyzed [1]. Unmanned aerial vehicles (UAVs) running simultaneous localization and mapping (SLAM) algorithms can also be used to perform numerous tasks, including surveillance, rescue, and transportation in extreme environments [2–4]. In the field of SLAM, the performance of state estimation is highly reliant on sensors, such as cameras, LiDAR, and inertial measurement units (IMUs). However, there are limitations associated with each type of sensor, such as minimum illumination requirements and the presence of noise. To overcome these shortcomings of stand-alone sensors, multiple sensors have been used to increase the reliability of estimation [5–9]. The methods utilizing multiple sensors for state estimation are categorized into two types: loosely coupled (cf. [5,6]) and tightly coupled (cf. [7–9]). The tightly coupled approach directly fuses LiDAR and inertial measurements through a joint optimization that minimizes some residuals, whereas the loosely coupled approach deals with the multiple sensors separately. The tightly coupled method is less computationally efficient and more difficult to implement than the loosely coupled approach, but it is more robust in its approach to noise and more accurate [8].

Accurate and real-time localization is crucial to the feedback control of UAVs in practical applications. Acquiring accurate localization information by solving the tightly coupled problem requires a considerable amount of computation, which decreases the frequency at which state estimation can be performed for providing real-time feedback. Moreover, the requirements of robust, precise, and fast localization increase the difficulty of designing algorithms. The visual inertial odometry (VIO) method [7] proposes achieving precise and real-time results based on tightly coupled VIO that fuses camera and IMU measurements for state estimation. However, its performance can be impaired by poor lighting conditions. Since 3D LiDAR sensors are less influenced by lighting conditions and can also provide range measurements of the surrounding environment, they have been successfully used for ego-motion estimation [8–11]. Most LiDAR systems update at a lower frequency than cameras (usually 10 Hz), which means that the point cloud can be distorted when the LiDAR moves aggressively. In contrast to LiDAR, an IMU is capable of extremely high update rates, and so combining LiDAR and IMU allows their individual deficiencies to be compensated, and the state estimation for UAVs can be solved by tightly coupled optimization.

In [11], the feature points were extracted from the LiDAR point cloud, and the corresponding features from the last LiDAR measurement were matched to estimate the ego-motion. To refine the odometry, feature points were matched and registered to the feature maps. In [9], it was shown that using tightly coupled LiDAR inertial odometry (LIO) with multiple window frames to local map is too time-consuming, while the accuracy is significantly degraded if there are too few windows. The approach taken in [7] cannot be used, either in a dark or highly dynamic illumination environment. Therefore, there is always a trade-off between high accuracy and computation efficiency, and the localization performance using other sensors (e.g., cameras) to reduce the computation loading can be affected by environmental factors.

1.2. Related Works

In [11], IMUs play an important role by providing an initial guess before performing estimation. However, this approach mainly relies on LiDAR information to estimate the motion, by matching feature points extracted from the local surface to the corresponding feature, estimating the relative transform, and constructing a global map. In [7], the gyroscope bias, scale, and direction of gravity can be corrected through the initialization step. In [12], the estimator combines IMU data and plane features obtained from LiDAR for joint optimization. Notably, feature planes are compressed into the closest point in each frame, so that the estimator is able to run in real time. In [9], tightly coupled 3D LIO for graph optimization was demonstrated in both indoor and outdoor environments, but the estimation process still took too long. In [13], a feature extraction algorithm for LiDAR systems was proposed with small fields of view, and feature points were used to estimate odometry and mapping. Moreover, linear interpolation was used to suppress the effect of motion blur associated with LiDAR movement, with each point in the same frame being compensated for this movement.

Loop closure is an approach that corrects the drift that occurs during long-term operation. The method starts by identifying previously visited places, and the iterative closest point (ICP) is the most common approach that involves searching the matches between the current laser scan and the existing map. Another method computes feature descriptors from laser scans, and then verifies loop closure based on certain conditions. In [10], features were segmented into many clusters, which enables the method to perform real-time pose estimation, even in a large-scale environment. In the back-end, the *k*-dimensional (KD)-tree method was used to search for the closest keyframe when performing loop closure, with loop closure established once the residual from ICP is sufficiently small. In [14], a real-time mapping approach was proposed that involved inserting laser scans into a probability grid. In that method, a branch-and-bound search runs in the back-end for loop closure, and if a sufficient match is found in the search window, the loop closure

constraint is added to the optimization problem. Overall, this system achieves a moderate capability and pixel-level accuracy using 2D LiDAR, but it is still too time-consuming for applying to 3D LiDAR data. The approach in [15] classified laser scans into segments with feature descriptors, and the transformation was obtained by matching these segments with the map. This method saves time compared to matching the entire laser scan, but its performance may be highly dependent on the accuracy of the classifier.

A tightly coupled LIO is developed in this work to obtain high-accuracy and high-frequency localization output for the feedback control of UAVs. Although tightly coupled methods normally require more computation loading, the developed approach can generate more-accurate and more-frequent localization information. The frame-to-map estimation process is robust and stable, and loop closure is applied to further correct the accumulated error when a loop is detected. Here, the performance of this new method is compared with other approaches in the literature using the publicly available KITTI (http://www.cvlibs.net/datasets/kitti/eval_odometry.php, accesed on 5 April 2021) dataset [16,17]. KITTI dataset was chosen, since it is the first dataset that provides accurate ground truth. The data were collected using a Velodyne HDL-64E laser scanner that produces more than one million 3D points per second and a state-of-the-art OXTS RT 3003 localization system which combines GPS, GLONASS, an IMU and RTK correction signals. Additionally, a fair comparison is possible using KITTI dataset due to its large scale nature as well as the proposed novel metrics, which capture different sources of error by evaluating error statistics over all sub-sequences of a given trajectory length or driving speed [16]. Many works (e.g., [10,11]) also used KITTI as benchmarks to evaluate the localization accuracy. The results demonstrate that the algorithm applied in this work can outperform other approaches in terms of both accuracy and frequency. The main contributions of this study are summarized as follows:

1. IMU excitation is not required for initialization, in contrast to [7].
2. Online relocalization combined with loop closure and pose-graph optimization methods have been developed for odometry and mapping that are more accurate than in [9].
3. In contrast to the odometry and mapping algorithm [11], the developed RTLIO can provide a high-frequency of odometry for the UAV and constructing maps synchronously.

1.3. Overview

The architecture of RTLIO is shown in Figure 1. The system starts with measurement preprocessing (Sect. IV), in which point clouds from the LiDAR measurement are classified into corner points and plane points. The distorted clouds are corrected by the integration of IMU measurements between two consecutive LiDAR frames. In the front-end (Sect. V, VI), the initialization processing provides the bias of the gyroscope, direction of gravity, and initial velocity for bootstrapping the subsequent nonlinear optimization-based RTLIO. In the sliding window optimization, the cost function is constructed to include the marginalization, LiDAR, and IMUs information for solving the UAV pose. In the back-end (Sect. VII), the loop closure is used to detect whether the current position has been revisited, and the pose graph optimization module is used to reduce the accumulated drift to increase positioning accuracy. Finally, RTLIO provides two frequency poses. One is the LiDAR-rate pose after preprocessing and optimization at 10 Hz; the other is the IMU-rate pose generated by the IMU propagation in RTLIO at 400 Hz.

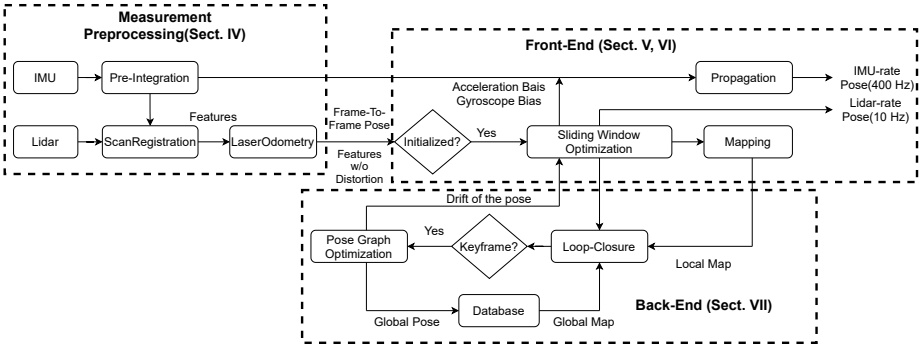

Figure 1. Architecture of the developed RTLIO.

Let body frame b_k be defined on the IMU, where k denotes the frame when the k^{th} LiDAR measurement is acquired. The world frame w is defined on the initial body frame, and the direction of the gravity is aligned with the z axis of the world frame. The LiDAR frame l be defined on the LiDAR. The rotation from frame A to frame B is denoted as q_A^B or R_A^B, and the translation transforming from frame A to frame B is denoted, as p_B^A. $\otimes$ represents the Hamilton product between two quaternions. All other variables are listed in Table 1.

Table 1. Notation.

Index	Note
$p \in \mathbb{R}^3$	position
$v \in \mathbb{R}^3$	velocity
q	quaternion
$\theta \in \mathbb{R}^3$	Euler angle
$R \in \mathrm{SO}(3)$	rotation matrix
$T \in \mathbb{R}^{44}$	transformation matrix
$\omega \in \mathbb{R}^3$	angular velocity
$a \in \mathbb{R}^3$	linear acceleration
$g \in \mathbb{R}^3$	gravity
$b_a, b_\omega \in \mathbb{R}^3$	acceleration and gyroscope bias
$n_a, n_\omega \in \mathbb{R}^3$	acceleration and gyroscope noise
P	point cloud
$\bar{P} \in \mathbb{R}^3$	a point in P
b	body frame
w	world frame
l	LiDAR frame
$(\cdot)^i$	state representation in ith frame
$(\cdot)_t$	state at time t
$\hat{(\cdot)}$	nominal state
$\lvert \cdot \rvert$	cardinality of the denoted argument
$m \in \mathbb{R}$	number of frames in sliding window

2. Methodology

2.1. Measurement Preprocessing

2.1.1. Time Alignment

The time stamps of the measurements from the LiDAR and camera are illustrated in Figure 2, where the sliding window includes the latest m LiDAR frames and each frame contains a set of IMU measurements, since the sensing rate of the IMU is much higher (e.g., 200 Hz).

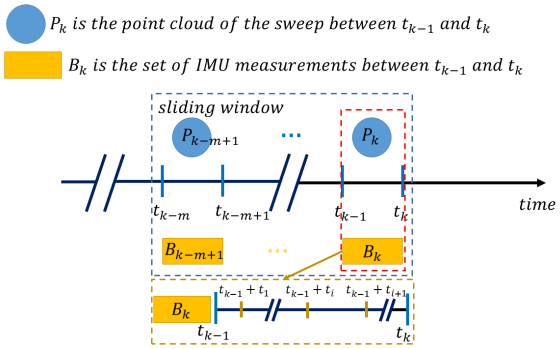

Figure 2. Time alignment between LiDAR point cloud P_k and the set of the IMU measurements B_k.

2.1.2. IMU Preintegration

IMU preintegration and the covariance matrix derivation with the continuous-time IMU dynamics of an error-state Kalman filter were proposed in [7,9,18,19]. Based on [20], the IMU states can be divided into true states X, nominal states $\hat{X}$, and error states δX, whose compositions are defined as

$$\begin{aligned} X &= \hat{X} \boxplus \delta X \\ X &= [\ \alpha,\ \ \theta,\ \ \beta,\ \ b_a,\ \ b_\omega\ ]^{\mathrm{T}}, \end{aligned} \tag{1}$$

where α, θ, and β are the position, orientation, and velocity, respectively, and b_a and b_ω are defined in Table 1. The operation $\boxplus$ for a state v in the vector space is simply the Euclidean addition, i.e., $v = \hat{v} + \delta v$, and for quaternion, it implies the multiplication of the quaternions, i.e., $\otimes$.

Measurements $\hat{a}_t$ and $\hat{\omega}_t$ at time $t \in (t_k,\ t_{k+1}]$ are defined as

$$\begin{aligned} \hat{a}_t &= a_t + b_{a_t} + R_{wg}^t g^w + n_a \\ \hat{\omega}_t &= \omega_t + b_{\omega_t} + n_\omega \\ n_a &\sim \mathcal{N}(0,\ \sigma_a^2) \\ n_\omega &\sim \mathcal{N}(0,\ \sigma_\omega^2), \end{aligned} \tag{2}$$

where n_a and n_ω are defined as random variables with normal distribution (i.e, $\mathcal{N}$) with zero mean and variances σ_a^2 and σ_ω^2.

The position, velocity, and orientation states between two body frames b_k and b_{k+1} can be propagated by integrating IMU measurements $\hat{a}_t$ and $\hat{\omega}_t$ during $t \in (t_k,\ t_{k+1}]$ in the world frame as

$$\begin{aligned} p_{b_{k+1}}^w &= p_{b_k}^w + v_{b_k}^w \Delta t_k \\ &\quad + \int\int_{t\in[t_k,\ t_{k+1}]} (R_t^w(\hat{a}_t - b_{a_t} - n_a) - g^w) dt^2 \\ v_{b_{k+1}}^w &= v_{b_k}^w + \int_{t\in[t_k,\ t_{k+1}]} (R_t^w(\hat{a}_t - b_{a_t} - n_a) - g^w) dt \\ q_{b_{k+1}}^w &= q_{b_k}^w \otimes \int_{t\in[t_k,\ t_{k+1}]} \begin{bmatrix} 0 \\ \frac{1}{2}\Omega(\hat{\omega}_t - b_{\omega_t} - n_\omega) \end{bmatrix} \gamma_t^{bk} dt, \end{aligned} \tag{3}$$

where Ω is the same as defined in (3) of [7]. Transforming (3) from the world frame to frame b_k yields

$$\begin{aligned} R_w^{b_k} p_{b_{k+1}}^w &= R_w^{b_k}(p_{b_k}^w + v_{b_k}^w \Delta t_k - \frac{1}{2} g^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k} \\ R_w^{b_k} v_{b_{k+1}}^w &= R_w^{b_k}(v_{b_k}^w - g^w \Delta t_k) + \beta_{b_{k+1}}^{b_k} \\ \gamma_{b_{k+1}}^{b_k} &= \left[q_{b_k}^w\right]^{-1} \otimes q_{b_{k+1}}^w, \end{aligned} \tag{4}$$

where $\alpha_{b_{k+1}}^{b_k}$, $\beta_{b_{k+1}}^{b_k}$, and $\gamma_{b_{k+1}}^{b_k}$ are the true states of the IMU integration and $\gamma_{b_{k+1}}^{b_k}$ is the quaternion form of $\theta_{b_{k+1}}^{b_k}$ defined as

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \int\int_{t\in[t_k,\, t_{k+1}]} R(\gamma_t^{b_k})(\hat{a}_t - b_{a_t} - n_a) dt^2 \\ \beta_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,\, t_{k+1}]} R(\gamma_t^{b_k})(\hat{a}_t - b_{a_t} - n_a) dt \\ \gamma_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,\, t_{k+1}]} \begin{bmatrix} 0 \\ \frac{1}{2}(\hat{\omega}_t - b_{\omega_t} - n_\omega) \end{bmatrix}_R \gamma_t^{b_k} dt. \end{aligned} \tag{5}$$

The noises in (5) are unknown, and so the nominal states can be expressed as

$$\begin{aligned} \hat{\alpha}_{b_{k+1}}^{b_k} &= \int\int_{t\in[t_k,\, t_{k+1}]} R(\hat{\gamma}_t^{b_k})(\hat{a}_t - \hat{b}_a) dt^2 \\ \hat{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,\, t_{k+1}]} R(\hat{\gamma}_t^{b_k})(\hat{a}_t - \hat{b}_a) dt \\ \hat{\gamma}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,\, t_{k+1}]} \begin{bmatrix} 0 \\ \frac{1}{2}(\hat{\omega}_t - \hat{b}_\omega) \end{bmatrix}_R \hat{\gamma}_t^{bk} dt, \end{aligned} \tag{6}$$

where $\hat{b}_a$ and $\hat{b}_\omega$ are the biases in the accelerometer and gyroscope.

The difference between the nominal states and the true states is minimized by correcting the nominal states, as described in Section 2.1.3.

2.1.3. Correction of Preintegration

Based on (1), the error state can be rewritten as

$$\delta X = X \boxminus \hat{X}, \tag{7}$$

where the operation $\boxminus$ for a state v in the vector space is simply the Euclidean addition, i.e., $v = \hat{v} - \delta v$, and for quaternion, it implies the multiplication of the inverse of the quaternion.

Taking the time derivatives of (5)–(7) yields

$$\begin{aligned} \dot{\delta X_i} &= F_i \delta X_i + V_i n \\ \begin{bmatrix} \dot{\delta\alpha_i} \\ \dot{\delta\theta_i} \\ \dot{\delta\beta_i} \\ \dot{\delta b_{a_i}} \\ \dot{\delta b_{\omega_i}} \end{bmatrix} &= F_i \begin{bmatrix} \delta\alpha_i \\ \delta\theta_i \\ \delta\beta_i \\ \delta b_{a_i} \\ \delta b_{\omega_i} \end{bmatrix} + V_i \begin{bmatrix} n_{a0} \\ n_{\omega 0} \\ n_{a1} \\ n_{\omega 1} \\ n_{b_a} \\ n_{b_\omega} \end{bmatrix} \\ n_{b_a} &\sim \mathcal{N}(0,\, \sigma_{b_a}^2) \\ n_{b_\omega} &\sim \mathcal{N}(0,\, \sigma_{b_\omega}^2), \end{aligned} \tag{8}$$

where F_i and V_i are error state dynamics matrices, n_{a0} and n_{a1} are the acceleration noises, $n_{\omega 0}$ and $n_{\omega 1}$ are the angular velocity noises, n_{b_a} and n_{b_ω} are modeled as random walks applied to the biases, and $\sigma_{b_a}^2$ and $\sigma_{b_\omega}^2$ are variances of n_{b_a} and n_{b_ω}, respectively.

Based on (8), the relation between error states δX_i and δX_{i+1} can be discretized as

$$\begin{aligned} \delta X_{i+1} &= \delta X_i + \delta \dot{X}_i \delta t \\ &= \delta X_i + (F_i \delta X_i + V_i n)\delta t \\ &= (I + F_i \delta t)\delta X_i + (V_i \delta t)n, \end{aligned} \tag{9}$$

which describes the relation of two error states at t_i and t_{i+1}, which can be extended to the two error states at t_{k+1} and t_k by

$$\begin{aligned} \delta X_{b_{k+1}} &= F' \delta X_{b_k} + V' n \\ F' &= \prod_{i=|B_{k+1}|-1}^{1} (I + F_i \delta t) \\ V' &= (V_{N-1}\delta t)n \\ &\quad + \Big(\sum_{i=|B_{k+1}|-2}^{1} \Big(\prod_{j=|B_{k+1}|-1}^{i+1} (I + F_j \delta t) \Big) V_i \delta t \Big) n. \end{aligned} \tag{10}$$

According to [18], covariance matrix $Q_{b_{k+1}}^{b_k}$ of $\delta X_{b_{k+1}}$ can be computed recursively using the first-order discrete-time covariance updated with the initial value $Q_{b_k}^{b_k} = 0$:

$$Q_{i+1}^{b_k} = (I + F_i \delta t) Q_i^{b_k} (I + F_i \delta t)^{\mathrm{T}} + (V_i \delta t) O (V_i \delta t)^{\mathrm{T}}, \tag{11}$$

where O contains the diagonal covariance matrices $\sigma_{a_0}^2$, $\sigma_{\omega_0}^2$, $\sigma_{a_1}^2$, $\sigma_{\omega_1}^2$, $\sigma_{b_a}^2$, and $\sigma_{b_\omega}^2$. Based on (1), (6), and (10), the corrected preintegrations denoted as $\bar{\alpha}_{b_{k+1}}^{b_k}$, $\bar{\beta}_{b_{k+1}}^{b_k}$, and $\bar{\gamma}_{b_{k+1}}^{b_k}$ are defined as

$$\begin{aligned} \bar{\alpha}_{b_{k+1}}^{b_k} &= \hat{\alpha}_{b_{k+1}}^{b_k} + J_{b_a}^{\alpha} \delta b_{a_k} + J_{b_\omega}^{\alpha} \delta b_{\omega_k} \\ \bar{\beta}_{b_{k+1}}^{b_k} &= \hat{\beta}_{b_{k+1}}^{b_k} + J_{b_a}^{\beta} \delta b_{a_k} + J_{b_\omega}^{\beta} \delta b_{\omega_k} \\ \bar{\gamma}_{b_{k+1}}^{b_k} &= \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_\omega}^{\gamma} \delta b_{\omega_k} \end{bmatrix}, \end{aligned} \tag{12}$$

where δb_{a_k} and δb_{ω_k} are obtained from (7), with b_{a_k} and b_{ω_k} discussed in Section 2.3, and $J_{b_a}^{\alpha}$ is the submatrix in F', whose location corresponds to $\frac{\delta \alpha_{b_{k+1}}^{b_k}}{\delta b_{a_k}}$. $J_{b_\omega}^{\alpha}$, $J_{b_a}^{\beta}$, $J_{b_\omega}^{\beta}$, and $J_{b_\omega}^{\gamma}$ also follow the same notation.

2.1.4. LiDAR Feature Extraction and Distortion Compensation

LiDAR measurements are not made synchronously due to the rotating mechanism inside the LiDAR sensor, and therefore the point cloud P_k in the k^{th} frame suffers from distortion, as shown in Figure 3a. This distortion was compensated for using IMU measurements, as shown in Figure 3b. First, P_k is segmented into N subframes by azimuthal angle ϕ, where P_k^i is the i^{th} subframe for $i \in \{1, 2, \ldots, N\}$. Second, the transformation matrix from t_k^i to t_k^N is defined as T_i^N, and is calculated from the IMU integration as $T_i^N = T(t_k^N) T^{-1}(t_k^i)$. Third, by performing subframe-wise transformation, the distortion-compensated point cloud denoted as P_k' is obtained as

$$P_k' = \left\{ T_1^N P_k^1, T_2^N P_k^2, \ldots, T_i^N P_k^i \right\}, i = 1, 2, \ldots, N. \tag{13}$$

The segmentation and t_k^i are depicted in detail in Figure 4. After performing distortion compensation, the feature points on the planes or the edges in each sweep are extracted using the feature extraction procedure proposed by [10,11].

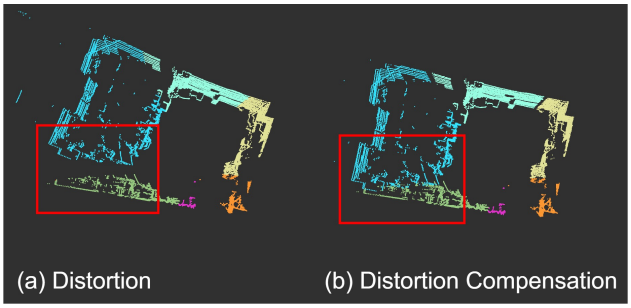

Figure 3. Calibrated results: (**a**) the point cloud suffered from distortion when LiDAR is moving, and (**b**) the result obtained after distortion compensation. Different colors indicate subframes in one sweep.

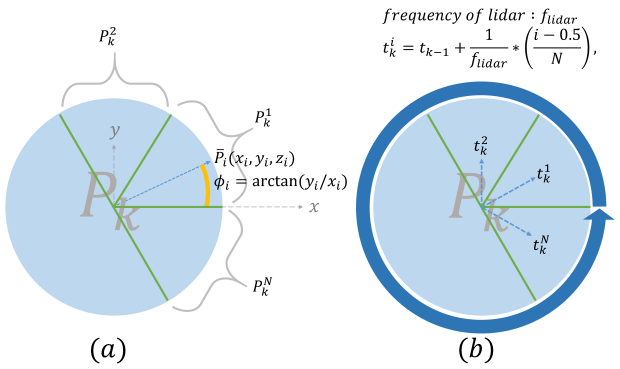

Figure 4. (**a**) Each subframe of P_k is defined as P_k^i. (**b**) The time of the i^{th} subframe is defined as t_k^i for $i \in \{1, \ldots, N\}$.

2.1.5. LiDAR Odometry

In Section 2.1.4, the feature points in each sweep are used to find the corresponding feature points in the last sweep, so that the transformation between each sweep (i.e., P'_k and P'_{k+1}, defined in (13)) can be obtained by minimizing the residual. The procedures are described in detail in [10,11].

2.2. Estimator Initialization

In the monocular visual-inertial system [7], the metric scale was recovered through the initialization process. However, the developed LiDAR-inertial system in this work does not require the initialization process to recover the metric scale thanks to the range measurement from the LiDAR sensor. To help improve the preintegration accuracy, the gyroscope bias b_ω needs to be estimated in Section 2.2.1, and the corrected preintegration can facilitate the estimation of the gravity vector in the first LiDAR frame g^{l_0} in Section 2.2.2.

2.2.1. Rotational Alignment

Consider two consecutive frames b_k and b_{k+1} in the sliding window, where l_0 represents the first LiDAR frame. Rotations $q_{b_k}^{l_0}$ and $q_{b_{k+1}}^{l_0}$ are obtained from the given extrinsic

parameters (p_l^b, q_l^b). Rotations $q_{l_k}^{l_0}$ and $q_{l_{k+1}}^{l_0}$ are from Section 2.1.5. The preintegration $\bar{\gamma}_{b_{k+1}}^{b_k}$ from (12) is combined to estimate δb_ω by minimizing the following cost function:

$$\min_{\delta b_\omega} \sum_{k=0}^{c-1} \left\| \left(q_{b_{k+1}}^{l_0} \right)^{-1} \otimes q_{b_k}^{l_0} \otimes \bar{\gamma}_{b_{k+1}}^{b_k} \right\|^2, \tag{14}$$

where c is the number of frames used for the initialization. Once the gyroscope bias is solved, preintegration terms $\hat{\alpha}_{b_{k+1}}^{b_k}$, $\hat{\beta}_{b_{k+1}}^{b_k}$, and $\hat{\gamma}_{b_{k+1}}^{b_k}$ will be repropagated using (12).

2.2.2. Linear Alignment

After computing the gyroscope bias, another important element to consider is the gravity vector. Initialization state χ_I is defined as

$$\chi_I = \left[v_{b_0},\ v_{b_1},\ v_{b_2},\ \cdots v_{b_{c-1}},\ g^{l_0} \right], \tag{15}$$

which includes the velocities on the body frame of each moment and the gravity vector, where the magnitude of g^{l_0} is known.

Remark 1. *When the UAV is moving during the initialization process, velocity v_{b_i} defined in (15) can be calculated from $p_{b_i}^{l_0}$, $p_{b_{i+1}}^{l_0}$, and $q_{b_i}^{l_0}$.*

Given two consecutive frames b_k and b_{k+1} in the window, $q_{b_k}^{l_0}$, $q_{b_{k+1}}^{l_0}$ and translations $p_{b_k}^{l_0}$, $p_{b_{k+1}}^{l_0}$ obtained are combined with IMU preintegration terms $\hat{\alpha}_{b_{k+1}}^{b_k}$, $\hat{\beta}_{b_{k+1}}^{b_k}$ to form the minimization problem

$$\min_{\chi_I} \sum_{k=0}^{c-1} \left\| \hat{z}_{b_{k+1}}^{b_k} - H_{b_{k+1}}^{b_k} \begin{bmatrix} v_{b_k} \\ v_{b_{k+1}} \\ g^{l_0} \end{bmatrix} \right\|^2, \tag{16}$$

to solve state χ_I defined in (15), where

$$\begin{aligned} \hat{z}_{b_{k+1}}^{b_k} &= \begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} - R_{l_0}^{b_k} \left(p_{b_{k+1}}^{l_0} - p_{b_k}^{l_0} \right) \\ \hat{\beta}_{b_{k+1}}^{b_k} \end{bmatrix} \\ &= H_{b_{k+1}}^{b_k} \chi_I + n_{b_{k+1}}^{b_k} \\ H_{b_{k+1}}^{b_k} &= \begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2} R_{l_0}^{b_k} \Delta t_k^2 \\ -I & R_{l_0}^{b_k} R_{b_{k+1}}^{l_0} & R_{l_0}^{b_k} \Delta t_k \end{bmatrix}, \end{aligned} \tag{17}$$

and $n_{b_{k+1}}^{b_k}$ is the measurement noise. The transformation between each LiDAR measurement, as well as the registered laser scans, will be transformed to the world frame using g^{l_0}. This is useful when l_0 frame might not be horizontal, where LiDAR-only odometry may result in a tilted map. After b_ω and g^{l_0} are estimated, they can be used as the initial conditions for the tightly coupled LIO in Section 2.3.

2.3. Front-End: Tightly Coupled LIO and Mapping

State vector χ, which includes all of the states in the sliding window, is defined as

$$\begin{aligned}\chi &= \left[x_0,\ x_1,\ x_2,\ \cdots,\ x_{m-1},\ x_l^b\right]\\ x_k &= \left[p_{b_k}^w,\ v_{b_k}^w,\ q_{b_k}^w,\ b_{a_k},\ b_{\omega_k}\right],\ k \in \{0,\ 1,\ 2, \ldots,\ m-1\}\\ x_l^b &= \left[p_l^b,\ q_l^b\right],\end{aligned} \tag{18}$$

where m is defined in Table 1, x_k is the state at the time when the k^{th} LiDAR measurements are acquired, and x_l^b are the extrinsic parameters between the LiDAR and IMU.

To estimate the state defined in (18), the following cost function is optimized to obtain a maximum posteriori estimation:

$$\begin{aligned}\min_{\chi}\Bigg\{ &\left\|r_p - H_p\chi\right\|^2 + \sum_{k=0}^{m-1}\left\|r_B\left(\hat{z}_{b_{k+1}}^{b_k},\ x_k,\ x_{k+1}\right)\right\|_{Q_{b_{k+1}}^{b_k}}^2\\ &+ \sum_{j\in L}\rho\left\|r_L\left(\hat{z}_{(j,f)},\ \chi\right)\right\|^2\Bigg\},\end{aligned} \tag{19}$$

where L is a set of indices that characterizes the LiDAR features in the sliding window, which includes two type of sets, namely edge E and plane F, such that $L = \{E,\ F\}$. f is the feature correspondence of j feature. ρ is a loss function used for outlier rejection, r_p and H_p are the prior information from marginalization defined in the subsequent analysis, $r_B\left(\hat{z}_{b_{k+1}}^{b_k},\ x_k,\ x_{k+1}\right)$ is the residual for the IMU measurements, and $r_L\left(\hat{z}_f^j,\ \chi\right)$ is the residual for the LiDAR measurements defined in the subsequent analysis. The residuals are described in detail in Sections 2.3.1 and 2.3.2. To make different types of measurements unitless and scale-invariant, the Mahalanobis norm is applied to (19). $Q_{b_{k+1}}^{b_k}$ is the covariance matrix of the IMU measurement, where $Q_{b_{k+1}}^{b_k}$ is obtained by propagating the uncertainty using (11). The Ceres solver [21] was used to solve the nonlinear problem defined in (19).

2.3.1. IMU Measurement Model

Replacing true states $\alpha_{b_{k+1}}^{b_k}$, $\beta_{b_{k+1}}^{b_k}$, and $\gamma_{b_{k+1}}^{b_k}$ in (4) with the result from (12), allows a residual form to be constructed in (20).

$$\begin{aligned}r_B\left(\hat{z}_{b_{k+1}}^{b_k},\ x_k,\ x_{k+1}\right) &= \begin{bmatrix} r_{\alpha_{b_{k+1}}^{b_k}}\\ r_{\theta_{b_{k+1}}^{b_k}}\\ r_{\beta_{b_{k+1}}^{b_k}}\\ r_{b_a}\\ r_{b_\omega}\end{bmatrix}\\ &= \begin{bmatrix} R(q_{b_k}^w)^T\left(p_{b_{k+1}}^w - p_{b_k}^w + \frac{1}{2}g^w\Delta t_k^2 - v_{b_k}^w\triangle t_k\right) - \bar{\alpha}_{b_{k+1}}^{b_k}\\ 2\left[\left(\bar{\gamma}_{b_{k+1}}^{b_k}\right)^{-1}\otimes\left(q_{b_k}^w\right)^{-1}\otimes q_{b_{k+1}}^w\right]_{xyz}\\ R(q_{b_k}^w)^T\left(v_{b_{k+1}}^w + g^w\Delta t_k - v_{b_k}^w\right) - \bar{\beta}_{b_{k+1}}^{b_k}\\ b_{a_{k+1}} - b_{a_k}\\ b_{\omega_{k+1}} - b_{\omega_k}\end{bmatrix},\end{aligned} \tag{20}$$

where $[\cdot]_{xyz}$ denotes the extraction of the imaginary part of the denoted quaternion.

2.3.2. LiDAR Measurement Model

The LiDAR cost function includes frame-to-frame and frame-to-map matching. The frame-to-map matching provides high precision for each state, while the frame-to-frame matching can suppress the variation of the states in the sliding window.

Consider $\overline{P}_j^{l_k}$ to be a feature in the k^{th} LiDAR frame. For frame-to-map matching, $\overline{P}_j^{l_k}$ is represented in the world frame w as

$$\overline{P}_j^{w} = T_{b_k}^{w} T_l^b \overline{P}_j^{l_k}. \tag{21}$$

For frame-to-frame matching, point $\overline{P}_j^{l_k}$ is represented in the previous LiDAR frame l_{k-1} as

$$\overline{P}_j^{l_{k-1}} = \left(T_l^b\right)^{-1} \left(T_{b_{k-1}}^{w}\right)^{-1} T_{b_k}^{w} T_l^b \overline{P}_j^{l_k}. \tag{22}$$

The residuals for edge E and plane F features are constructed, as shown in Figure 5, and defined as follows.

2.3.3. Residuals for the Edge Features

The point-to-line distance describing the residuals for edge features can be computed as

$$r_E\left(\hat{z}_{(j,f)}^t, \chi\right) = \frac{\left|\left(\overline{P}_j^t - \overline{P}_{f_1}^t\right) \times \left(\overline{P}_j^t - \overline{P}_{f_2}^t\right)\right|}{\left|\overline{P}_{f_1}^t - \overline{P}_{f_2}^t\right|}, \tag{23}$$

where $\overline{P}_j^t$ is $\overline{P}_j^{l_k}$ represented in t frame, t can represent either w or l_{k-1} using (21) or (22), respectively. $\overline{P}_{f_1}^t$ is the closest edge feature to $\overline{P}_j^t$, and $\overline{P}_{f_2}^t$ is the second closest point.

2.3.4. Residuals for the Plane Features

The point-to-plane distance known as the Hesse normal form can be computed

$$r_F\left(\hat{z}_{(j,f)}^t, \chi\right) = n_f^t \cdot \overline{P}_j^t + d_f^t, \tag{24}$$

where n_f^t is the normal vector of the closest plane to $\overline{P}_j^t$, and d_f^t is the distance from the closest plane to the origin of frame t.

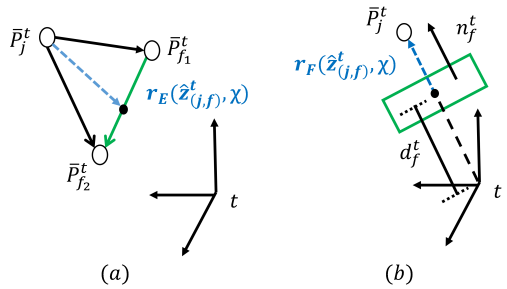

Figure 5. Illustration of residuals for edge and plane features. Residuals are shown by the blue lines. (**a**) The residual of edge feature $\overline{P}_j^t$ with the corresponding line shown in green that is formed by $\overline{P}_{f_1}^t$ and $\overline{P}_{f_2}^t$. (**b**) The residual of the plane feature $\overline{P}_j^t$ with the plane shown in green that is formed by the feature correspondences.

The residuals described in (23) and (24) are applied to construct one of the residuals defined in (19) as

$$\sum_{j\in L}\rho\left\|r_L\left(\hat{z}_{(j,f)},\,\chi\right)\right\|^2=\sum_{j\in E}\rho\left\|r_E\left(\hat{z}^w_{(j,f)},\,\chi\right)\right\|^2+\sum_{j\in F}\rho\left\|r_F\left(\hat{z}^w_{(j,f)},\,\chi\right)\right\|^2+\sum_{j\in F}\rho\left\|r_F\left(\hat{z}^{l_{k-1}}_{(j,f)},\,\chi\right)\right\|^2, \tag{25}$$

where $r_E\left(\hat{z}^w_{(j,f)},\,\chi\right)$ is the residual of the edge feature for frame-to-map matching, $r_F\left(\hat{z}^w_{(j,f)},\,\chi\right)$ is the residual of the plane feature for frame-to-map matching, and $r_F\left(\hat{z}^{l_{k-1}}_{(j,f)},\,\chi\right)$ is the residual of the plane feature for frame-to-frame matching. The residual of the edge feature for frame-to-frame is not considered, since it does not help for boosting the accuracy of RTLIO.

2.3.5. Marginalization

In order to reduce the computational complexity and preserve the history information, a marginalization procedure needs to be applied to the sliding window method. This marginalization aims to keep the most-recent frame in the window, and the Schur complement is applied to construct a prior term based on marginalized measurements. The detail can be referred to [22,23]. A factor graph of such a system is shown in Figure 6. The frame-to-map constraints do not influence the adjacent states, and so only the frame-to-frame constraints are considered.

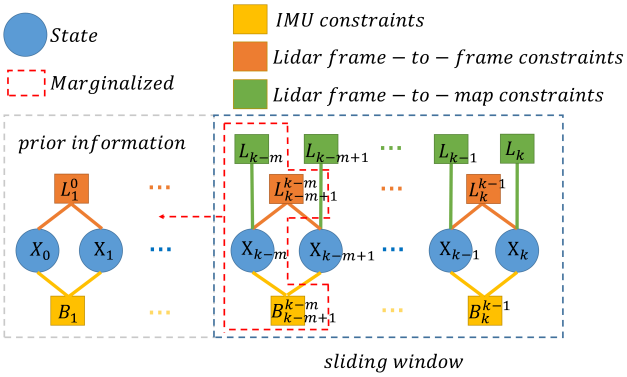

Figure 6. Illustration of the factor graph and the marginalization strategy. The oldest frame in the sliding window will be marginalized into prior information after optimizing (19).

Combining all of the residuals and solving the cost function defined in (19) yields the best estimation of states. The local map is then obtained based on the current state estimation, by applying an appropriate algorithm [11].

2.4. Back-End: Loop Closure and Pose-Graph Optimization

The optimization-based approach provides sufficient accuracy in an indoor environment, but for large-scale cases, it is inevitable that accumulated drift will occur due to various factors, such as extrinsic parameters between the LiDAR and IMU, the asynchronous sampling of measurements, and inaccurate data association during LiDAR matching. One way to correct for such drift is using loop closure. This method starts with identifying the previously visited places. Once a loop is detected by computing feature correspondences, a relocalization process tightly integrates these constraints into a cost function. This procedure minimizes drift and achieves much smoother state estimation. After the loop closure and relocalization are performed, the sliding window shifts and aligns with the

past poses. Then, a pose-graph optimization algorithm can match all keyframes, in order to minimize the drift and ensure the global consistency of the system. These processes might not influence the current state estimation, but the optimized pose-graph can facilitate the consistency of global map reconstruction after performing the state estimation.

2.4.1. Loop Closure

The loop closure algorithm is described in Algorithm 1. Once a frame is marginalized from the sliding window, its point cloud in the body frame, P_m, and its pose, T_m, will be fed into the loop closure algorithm. If the L2-norm between T_m and the pose at the lastest keyframe is higher than an Euclidean distance threshold, the marginalized frame is considered as a new keyframe. In this way, the keyframes are kept uniformly distributed in the space. Then, KD-tree search with search radius of r is performed if the keyframe database, (D_T, D_P), which is the set of the keyframe pose and point cloud, is not empty. $T_m^{'}$ from D_T is the closest keyframe transformation matrix to T_m. If $T_m^{'}$ can be found in D_T, the loop is assumed to be detected. Then, T_{ICP} is obtained by matching P_m with the local map, M, based on the threshold of the point-to-point RMSEs. M is the local map constructed by registering keyframe point cloud D_P to the world frame based on D_T.

Algorithm 1 Loop closure algorithm.

Input: T_m, P_m from the sliding window
Output: T_{ICP}

if (T_m, P_m).isKeyframe() **then**
  if $D_T \neq \phi$ or $D_P \neq \phi$ **then**
    $T_m^{'} \leftarrow$ KDtree.RadiusSearch(T_m, D_T, r);
    $M \leftarrow$ registerPointCloud(D_T, D_P);
    if $T_m^{'} \neq \mathbf{0}$ **then**
      $T_{ICP} \leftarrow$ ComputeICP(P_m, M, T_m);
    end if
  end if
  $T_m^{'} \leftarrow \mathbf{0}$
  $D_T = D_T \cup T_m$
  $D_P = D_P \cup P_m$
end if

2.4.2. Tightly Coupled Relocalization

As long as the current pose in the world coordinate is obtained, T_{ICP} and M are fed back to the RTLIO module for state correction. The relocalization scheme is modified from (19) by solving the following cost function:

$$\min_{\chi}\Bigg\{\left\|r_p - H_p\chi\right\|^2 + \sum_{k=0}^{m-1}\left\|r_B\left(\hat{z}_{b_{k+1}}^{b_k},\, x_k,\, x_{k+1}\right)\right\|_{Q_{b_{k+1}}^{b_k}}^2 + \sum_{j\in L}\rho\left\|r_L\left(\hat{z}_{(j,f)},\, \chi\right)\right\|^2 + \sum_{j\in L}\rho\left\|r_M\left(\hat{z}_{(j,m)},\, \chi\right)\right\|^2\Bigg\}$$

with the constraint

$$\begin{aligned} r_M\left(\hat{z}_{(j,m)},\, \chi\right) &= \overline{P}_j^w - \overline{P}_m^w \\ &= R_{b_k}^w\left(R_l^b\overline{P}_j^{l_k} + p_l^b\right) + p_{b_k}^w - \overline{P}_m^w, \end{aligned} \tag{26}$$

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

2.4.3. Global Pose-Graph Optimization

Due to the LiDAR inertial setup, roll and pitch angles are fully observable once gravity and the bias are estimated. Therefore, the accumulated drift only occurs in the other four degrees of freedom (x, y, z, yaw) and can be reduced by solving keyframe states in the pose-graph. Every keyframe state serves as a vertex in the pose-graph, and two types of edges between the vertices are utilized. The pose-graph is illustrated in Figure 7.

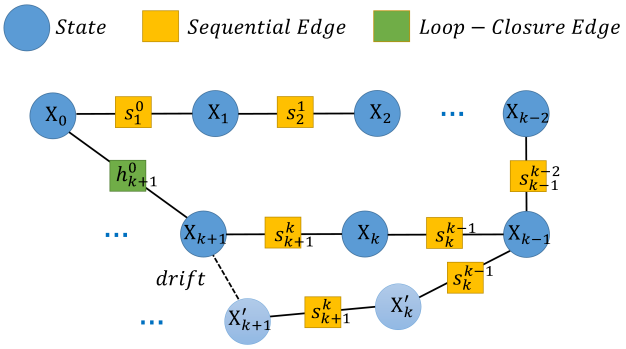

Figure 7. Constructing a pose-graph: every node in the graph represents the state of a keyframe. S is the set of the sequential edges, where $S = \left\{ s_1^0,\ s_2^1,\ \cdots,\ s_{k+1}^k,\ \cdots \right\}$. H is the set of loop closure edges, where $H = \left\{ h_{k+1}^0,\ \cdots \right\}$.

2.4.4. Sequential Edge

A sequential edge represents the relative transformation between each keyframe, which is obtained from the RTLIO results. Considering a keyframe j and its previous keyframe i, the sequential edge is defined as $s_j^i = \left\{ \hat{p}_j^i,\ \hat{\psi}_j^i \right\}$, where $\hat{p}_j^i$ and $\hat{\psi}_j^i$ denote the relative position and the relative yaw angle, respectively:

$$\begin{aligned} \hat{p}_j^i &= \hat{R}_i^w (\hat{p}_j^w - \hat{p}_i^w) \\ \hat{\psi}_j^i &= \hat{\psi}_j^w - \hat{\psi}_i^w. \end{aligned} \tag{27}$$

If the current keyframe j has a corresponding keyframe i, the loop closure edge is defined as $h_j^i = \left\{ \hat{p}_j^i,\ \hat{\psi}_j^i \right\}$ which is obtained in Section 2.4.1. Then loop closure edge will be added to the pose-graph as an additional constraint.

2.4.5. Pose-Graph Optimization for Four Degrees of Freedom

State vector χ_p of the pose-graph is defined as

$$\begin{aligned} \chi_p &= [x_0,\ x_1,\ \cdots,\ x_{n-1}] \\ x_k &= [p_k^w,\ \psi_k^w],\ k \in \{1,\ 2,\ \ldots,\ n-1\}, \end{aligned} \tag{28}$$

where n is the number of vertices in the pose-graph.

To find state χ_p defined in (28), the cost function is formulated as

$$\min_{\chi_p} \left\{ \sum_{s_j^i \in S} \| r_{ij} \|^2 + \sum_{h_j^i \in H} \rho \| r_{ij} \|^2 \right\}, \tag{29}$$

where the residuals of the sequential edge and the loop closure edge between keyframes i and j are defined as

$$r_{ij}(p_i^w,\ \psi_i^w,\ p_j^w,\ \psi_j^w) = \begin{bmatrix} R(\hat{\phi}_i^w,\ \hat{\theta}_i^w,\ \psi_i^w)^{-1}(p_j^w - p_i^w) - \hat{p}_j^i \\ (\psi_j^w - \psi_i^w) - \hat{\psi}_j^i \end{bmatrix}, \tag{30}$$

where $\hat{\phi}_i^w$ and $\hat{\theta}_i^w$ are the roll and pitch angles, respectively, converted from $q_{b_k}^w$ defined in (18). The loss function ρ is applied to penalize wrong connections of the loop closure edges. Once pose-graph optimization is completed, all keyframe states are updated. The global map is then updated by registering the keyframe point cloud according to the states.

3. Experiment Results and Discussions

A series of experiments (https://github.com/ChadLin9596/ncrl_lio, accessed on 2 June 2021) were conducted to analyze the performance of the developed RTLIO algorithm and compare it with the current state-of-the-art algorithms. To demonstrate the real-time capability of our system, multiple indoor tests are presented in Section 3.1. The KITTI dataset was used to compare the results with real-world benchmarks; the results are discussed in Section 3.2.

3.1. Indoor Flight Test

During the experiments, multiple threads were utilized to achieve the desired performance in real time. The first thread performed distortion compensation and feature extraction from LiDAR measurements, as described in Section 2.1.4. The second thread took those features and computed the incremental motion, as described in Section 2.1.5. The third thread (described in Section 2.3) executed the RTLIO algorithm that solves the states based on the initial guess from the second thread. The RTLIO generated two types of odometry defined in Section 1.3: (i) LiDAR-rate pose, and (ii) the IMU-rate pose, which can be obtained with minimal delay. This means that the high-frequency pose can be directly used for real-time feedback control.

The precision and computation time are discussed and compared with other LiDAR-based methods in Section 3.1.2, for experiments conducted in the laboratory with the OptiTrack motion capture system as the ground truth. The flight tests with RTLIO and the other methods are presented in Section 3.1.3.

3.1.1. System Setup

The quadcopter setup used in this work is shown in Figure 8. It comprised a 16-beam LiDAR system (Velodyne VLP-16, 10 Hz), IMU (400 Hz), and Intel NUC (NUC8i7BEH) with an i7-8559U CPU running at 2.70 GHz and 20 GB of memory. The RTLIO algorithm is implemented on the board to perform state estimation in real time.

Figure 8. The quadcopter used for the indoor flight tests.

3.1.2. Precision and Time Cost

Data recorded with quadcopter flying in circular trajectories in the laboratory were used as the input to LOAM (cf. [11]), ALOAM (cf. [11] (https://github.com/HKUST-Aerial-Robotics/A-LOAM, accessed on 1 April 2021), and RTLIO to conduct postprocessing. The comparison of performance is shown in Figure 9, where ALOAM is an extension of LOAM produced by HKUST. The relative pose errors (RPEs) and the time costs of the methods are listed in Table 2, which indicates that the odometry from RTLIO had lower RPEs, but a greater time cost. The methods discussed in this section are able to estimate the pose state to a certain precision, and the impact of the time costs is discussed in Section 3.1.3. The IMU-rate poses from RTLIO cannot be compared with poses from ALOAM and LOAM, because ALOAM and LOAM cannot generate high frequency poses.

Table 2. Comparison of RMSE of RPE and average time costs for the RTLIO, ALOAM and LOAM methods.

Method	Number of Frames	Translation (m)	Rotation (deg)	Computation Time (ms)
LOAM (10 Hz)	1203	0.0599	1.4218	67.5977
ALOAM (10 Hz)	1224	0.0078	0.3955	61.1810
RTLIO (10 Hz)	1224	0.0066	0.1881	96.3577

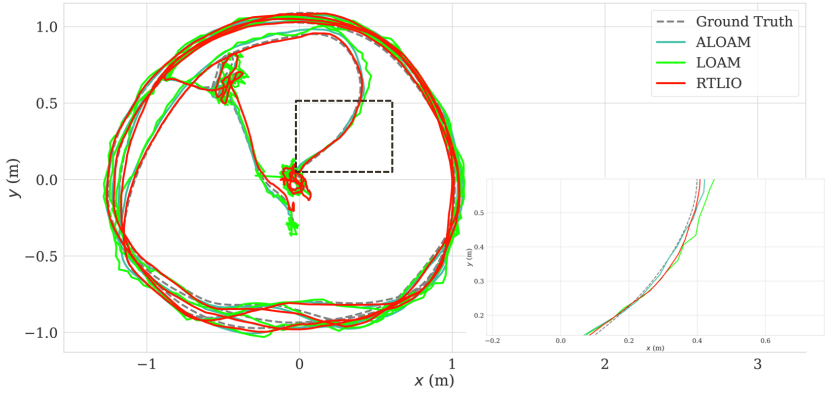

Figure 9. The comparison of trajectory with LOAM, ALOAM, RTLIO.

3.1.3. Indoor Flights

Figure 10 shows the results of RTLIO along the x, y, and z axes, from take off to landing. These results show that high-performance localization is crucial to the real-time feedback control of the quadrotor and trajectory tracking, and the time delays from RTLIO and LOAM are compared in Figure 11. The computation time for RTLIO is 0.2944 ms, and the delay is small enough for feedback control.

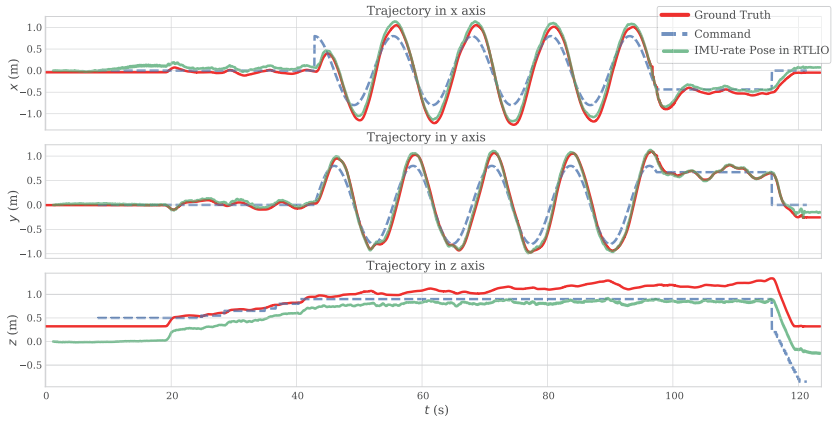

Figure 10. Flight trajectory with RTLIO along the x, y, and z axes.

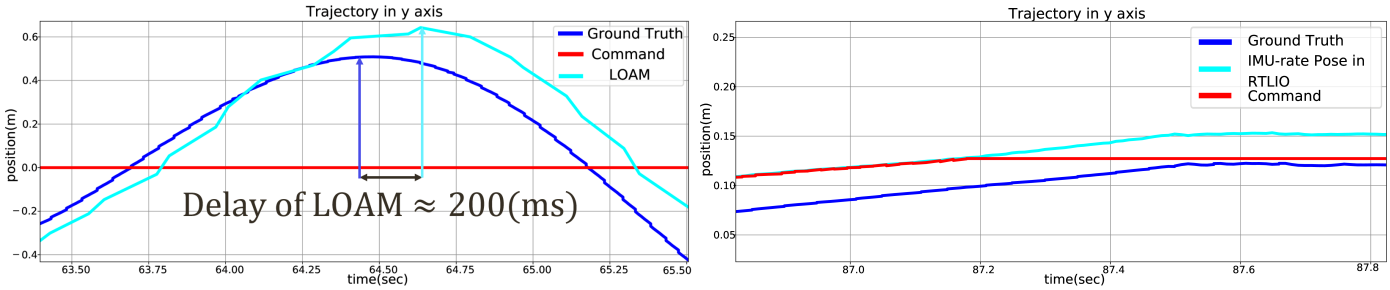

Figure 11. Time delays for (**left**) LOAM, (**right**) IMU-rate pose in RTLIO.

3.1.4. Indoor Flight with an Obstacle

Indoor flight experiments were also conducted with the quadcopter flying along a corridor with the localization obtained by RTLIO. Figure 12a shows the setup of the indoor test environment, Figure 12b shows the top view of the map in the xy plane and the trajectory of the UAV, starting from the "WORLD" to "IMU". These tests and the results presented in Section 3.1.3 demonstrate the capability of the RTLIO algorithm to perform localization for the feedback control of the quadcopter and trajectory tracking, either in a laboratory or corridor environment, and that the generated mapping is reliable.

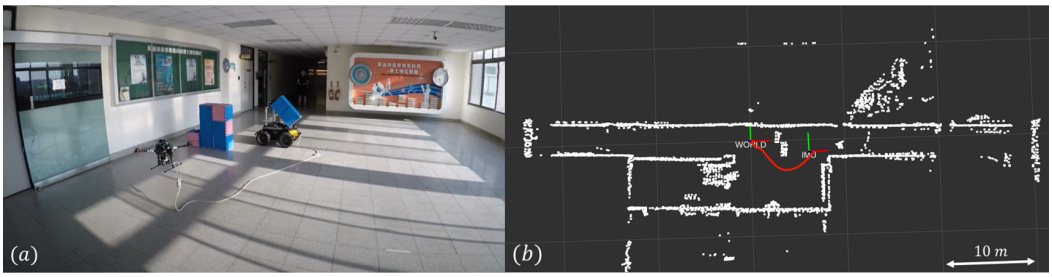

Figure 12. Results for indoor flying with a corridor: (**a**) setup (**b**) top view of the map and the trajectory of the entire flight (from WORLD to IMU).

3.2. KITTI Dataset Evaluation

The developed RTLIO was also evaluated using KITTI dataset, which includes measurements from an inertial navigation system (OXTS RT3003), which provides the ground-truth pose and IMU measurements at 100 Hz, a 64-beam LiDAR (Velodyne HDL-64E, 10 Hz), two grayscale (Point Grey Flea 2 FL2-14S3M-C, 10 Hz), and two color cameras (Point Grey Flea 2 FL2-14S3C-C, 10 Hz). In this test, only the IMU and the LiDAR were used to evaluate our algorithm.

3.2.1. Front-End Performance

The RTLIO in the front-end did not include loop closure and pose-graph optimization. The results show that the average errors of the translation and rotation along the given path were 1.8560 % and 0.0043 deg/m, respectively, as reported by the KITTI evaluation. The average translation and rotation errors over different lengths in each sequence are shown in Figure 13.

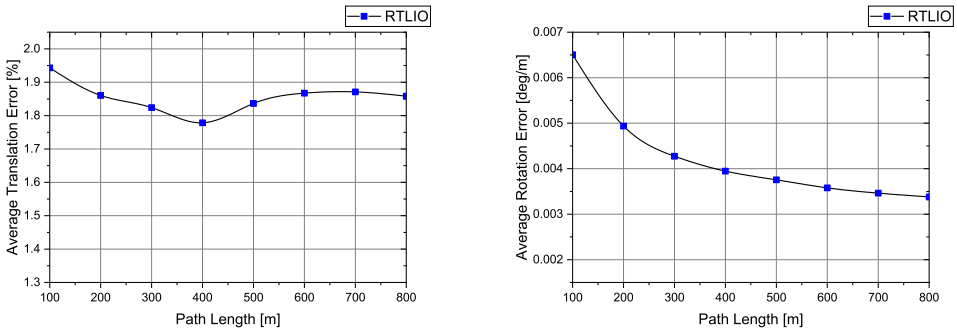

Figure 13. Average translation and rotation errors of the front-end evaluated over different lengths in the KITTI dataset for sequence from 00 to 10.

3.2.2. Full Closed-Loop Performance

After adding the back-end to the RTLIO, the full pipeline was also evaluated using the KITTI dataset. The overall results show that the average errors of the translation and rotation along the given path are 1.6392% and 0.0035 deg/m, respectively. Figure 14 shows that the RTLIO including the back-end effectively reduces the errors compared with the front-end in Figure 13.

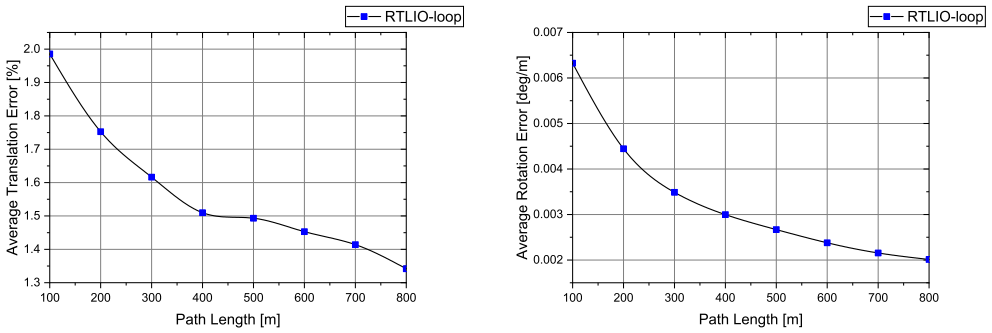

Figure 14. Average translation and rotation errors of the full pipeline evaluated over different lengths in the KITTI dataset for sequences 00 to 10.

The APE (Absolute Pose Error) evaluated with EVO (https://github.com/MichaelGrupp/evo, accessed on 1 April 2021) is listed in Table 3, in which the sequences that contain a trajectory without loop closure are marked with an *.

The results in Table 3 indicate that the RTLIO with back-end can outperform RTLIO in APE (especially in the sequence with loop closure).

Table 3. Translation and rotation of APE in the KITTI dataset.

	RTLIO		RTLIO with Back-End	
Sequence	**Translation (m)**	**Rotation (deg)**	**Translation (m)**	**Rotation (deg)**
00	9.4542	2.5884	**1.8196**	**0.7324**
* 01	**27.5966**	**8.0052**	31.3346	9.2077
02	10.3673	1.5718	**5.8435**	**1.3680**
* 04	1.8050	**1.2320**	**1.0295**	1.3538
05	3.5576	1.7812	**0.9164**	**0.4610**
06	5.7340	2.9552	**1.4797**	**0.6996**
07	1.2983	0.7238	**0.9850**	**0.6497**
08	22.4302	4.0389	**10.2060**	**2.1994**
09	20.9436	5.8630	**3.4717**	2.3290
* 10	2.3719	1.2684	**2.3041**	**1.2050**

3.3. Time Consumption

The time consumption of each module in indoor flight test and KITTI datasets using an Intel i7-7700 CPU with 24~GB of memory is listed in Table 4. Threads 1–3 are used for computing the front-end of the RTLIO, and thread 4 is used for the back-end, which also reconstructs a globally consistent map. However, the RTLIO was unable to run in real time using the KITTI datasets, since scan matching is more difficult in the outdoor environment. Additionally, the time consumption increases with the increase of the number of the channels of the LiDAR. The higher the number of channels, the higher the resolution (i.e., 16 channels for VLP-16, 32 channels for HDL-32E), according to the website from Velodyne: (https://velodynelidar.com/products/hdl-32e, accessed on 2 June 2021).

Table 4. Time Statistics.

Thread	**Module**	**Time (ms)**		**Rate (Hz)**
		Indoor	KITTI	
1	feature extraction	6	25	10
2	frame-to-frame odometry	15	65	10
3	sliding window optimization	65	350	10
4	loop closure	130	200	X
	pose-graph optimization	10	120	X

4. Conclusions and Future Work

The RTLIO developed in this work can generate accurate and reliable odometry information in real time, and the initialization process is performed when the UAV is already in motion. The developed RTLIO method uses LiDAR and IMU to generate high-frequency odometry with improved performance compared to the methods that only use LiDARs. Moreover, a consistent and accurate global map is constructed using the loop closure and pose-graph optimization method. Experiments were conducted with the quadrotor in indoor environment and using KITTI dataset, and the results demonstrated that the RTLIO can outperform ALOAM and LOAM in terms of exhibiting a smaller time delay and greater flight stability. The RTLIO with back-end algorithm can outperform the RTLIO with only front-end algorithm, since the accumulated drift can be reduced by the developed pose graph.

Future works include designing a more stable initialization method to deal with diverse situations. In addition, detection algorithms can be integrated into the method for removing feature points on moving objects. Finally, integrating vision sensors with the current system to improve the precision of odometry will be conducted to increase the stability of pose estimation along the z axis.

Author Contributions: Conceptualization, T.-H.C.; methodology, J.-C.Y. and C.-J.L.; software, B.-Y.Y. and Y.-L.Y.; validation, B.-Y.Y. and Y.-L.Y.; formal analysis, J.-C.Y. and C.-J.L.; investigation, J.-C.Y. and C.-J.L.; resources, B.-Y.Y. and Y.-L.Y.; data curation, B.-Y.Y. and Y.-L.Y.; writing—original draft preparation, J.-C.Y. and C.-J.L.; writing—review and editing, T.-H.C., B.-Y.Y. and Y.-L.Y.; visualization, B.-Y.Y. and Y.-L.Y.; supervision, T.-H.C.; project administration, T.-H.C.; funding acquisition, T.-H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Ministry of Science and Technology, Taiwan (Grant Number MOST 107-2628-E-009-005-MY3) and by Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan (Grant Number MOST 110-2634-F-009-018-).

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. Lippitt, C.D.; Zhang, S. The impact of small unmanned airborne platforms on passive optical remote sensing: A conceptual perspective. *Int. J. Remote Sens.* **2018**, *39*, 4852–4868. [CrossRef]
2. Lin, Y.; Gao, F.; Qin, T.; Gao, W.; Liu, T.; Wu, W.; Yang, Z.; Shen, S. Autonomous aerial navigation using monocular visual-inertial fusion. *J. Field Robot.* **2018**, *35*, 23–51. [CrossRef]
3. Weiss, S.; Achtelik, M.W.; Lynen, S.; Chli, M.; Siegwart, R. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 957–964.
4. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Robot. Res.* **2015**, *34*, 314–334. [CrossRef]
5. Mourikis, A.I.; Roumeliotis, S.I. A multi-state constraint kalman filter for vision-aided inertial navigation. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 10–14 April 2007; pp. 3565–3572.
6. 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]
7. 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]
8. Qin, C.; Ye, H.; Pranata, C.E.; Han, J.; Liu, M. LINS: A lidar-inerital state estimator for robust and fast navigation. *arXiv* **2019**, arXiv:1907.02233.
9. 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.
10. 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.
11. Zhang, J.; Singh, S. Loam: Lidar odometry and mapping in real-time. In Proceedings of the Robotics: Science and Systems Conference, Berkeley, CA, USA, 12–16 July 2014.
12. 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.
13. Lin, J.; Zhang, F. Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. *arXiv* **2019**, arXiv:1909.06700
14. 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.
15. Dube, R.; Dugas, D.; Stumm, E.; Nieto, J.; Siegwart, R.; Cadena, C. Segmatch: Segment based place recognition in 3d point clouds. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
16. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the KITTI vision benchmark suite. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012.
17. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res. IJRR* **2013**, *32*, 1231–1237. [CrossRef]
18. Lupton, T.; Sukkarieh, S. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robot.* **2012**, *28*, 61–76. [CrossRef]

19. Shen, S.; Michael, N.; Kumar, V. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA, 26–30 May 2015; pp. 5303–5310.
20. Solà, J. Quaternion kinematics for the error-state kalman filter. *arXiv* **2017**, arXiv:1711.02508.
21. Agarwal, S.; Mierle, K. Ceres Solver. Available online: (accessed on 3 April 2021).
22. Sibley, G.; Matthies, L.; Sukhatme, G. Sliding window filter with application to planetary landing. *J. Field Robot.* **2010**, *27*, 587–608. [CrossRef]
23. Dong-Si, T.; Mourikis, A.I. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5655–5662.