FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter

Wei Xu and Fu Zhang

***Abstract*—This letter presents a computationally efficient and robust LiDAR-inertial odometry framework. We fuse LiDAR feature points with IMU data using a tightly-coupled iterated extended Kalman filter to allow robust navigation in fast-motion, noisy or cluttered environments where degeneration occurs. To lower the computation load in the presence of a large number of measurements, we present a new formula to compute the Kalman gain. The new formula has computation load depending on the state dimension instead of the measurement dimension. The proposed method and its implementation are tested in various indoor and outdoor environments. In all tests, our method produces reliable navigation results in real-time: running on a quadrotor onboard computer, it fuses more than 1200 effective feature points in a scan and completes all iterations of an iEKF step within 25 ms. Our codes are open-sourced on Github.[1]**

***Index Terms*—Aerial systems, localization, perception and autonomy, sensor fusion.**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is a fundamental prerequisite of mobile robots, such as unmanned aerial vehicles (UAVs). Visual (-inertial) odometry (VO), such as Stereo VO [1], [2] and Monocular VO [3], [4] are commonly used on mobile robots due to their lightweight and low-cost. Although providing rich RGB information, visual solutions lack direct depth measurements and require much computation resources to reconstruct the 3D environment for trajectory planning. Moreover, they are very sensitive to lighting conditions. Light detection and ranging (LiDAR) sensors could overcome all these difficulties but have been too costly (and bulky) for small-scale mobile robots.

Solid-state LiDARs recently emerge as main trends in LiDAR developments, such as those based on micro-electro-mechanical-system (MEMS) scanning [5] and rotating prisms [6]. These LiDARs are very cost-effective (in a cost range similar to global shutter cameras), lightweight (can be carried by a small-scale UAV), and of high performance (producing active and direct 3D measurements of long-range and high-accuracy). These features make such LiDARs viable for UAVs, especially industrial UAVs, which need to acquire accurate 3D maps of the environments (e.g., aerial mapping) or may operate in cluttered environments with severe illumination variations (e.g., post-disaster search and inspection).

Despite the great potentiality, solid-state LiDARs bring new challenges to SLAM: 1) the feature points in LiDAR measurements are usually the geometrical structures (e.g., edges and planes) in the environments. When the UAV is operating in cluttered environments where no strong features are present, the LiDAR-based solution easily degenerates. This problem is more obvious when the LiDAR has a small FoV. 2) Due to the high-resolution along the scanning direction, a LiDAR scan usually contains many feature points (e.g., a few thousand). While these feature points are not adequate to reliably determine the pose in case of degeneration, tightly fusing such a large number of feature points to IMU measurements requires tremendous computational resources that are not affordable by the UAV onboard computer. 3) Since the LiDAR samples point sequentially with a few laser/receiver pairs, laser points in a scan are always sampled at different times, resulting in motion distortion that will significantly degrade a scan registration [7]. The constant rotations of UAV propellers and motors also introduce significant noises to the IMU measurements.

To make the LiDAR navigation viable for small-scale mobile robots such as UAVs, we propose the FAST-LIO, a computationally efficient and robust LiDAR-inertial odometry package. More specifically, our contributions are as follows: 1) To cope with fast-motion, noisy or cluttered environments where degeneration occurs, we adopt a tightly-coupled iterated Kalman filter to fuse LiDAR feature points with IMU measurements. We propose a formal back-propagation process to compensate for the motion distortion; 2) To lower the computation load caused by a large number of LiDAR feature points, we propose a new formula for computing the Kalman gain and prove its equivalence to the conventional Kalman gain formula. The new formula has a computation complexity depending on the state dimension instead of the measurement dimension. 3) We implement our formulations into a fast and robust LiDAR-inertial odometry software package. The system is able to run on a small-scale quadrotor onboard computer. 4) We conduct experiments in various indoor and outdoor environments and with actual UAV flight tests (Fig. 1) to validate the system's robustness when fast motion or intense vibration noise exists.

The remaining letter is organized as follows: In Section II, we discuss relevant research works. We give an overview of the

Manuscript received October 15, 2020; accepted February 16, 2021. Date of publication March 8, 2021; date of current version March 23, 2021. This letter was recommended for publication by Associate Editor M. Chli and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported by DJI (Project 200009538). *(Corresponding author: Fu Zhang.)*

The authors are with the Mechatronics and Robotic Systems (MaRS) Laboratory, Department of Mechanical Engineering, University of Hong Kong, Hong Kong (e-mail: xuweii@hku.hk; fuzhang@hku.hk).

Digital Object Identifier 10.1109/LRA.2021.3064227

[1]https://github.com/hku-mars/FAST_LIO

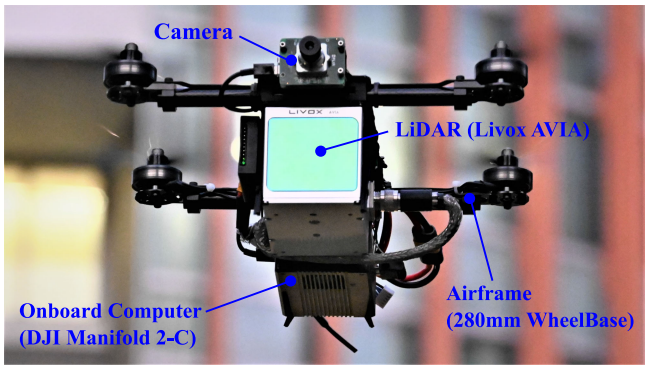

Fig. 1. Our LiDAR-inertial navigation system runs on a Livox AVIA LiDAR[2] and a DJI Manifold 2-C onboard computer[3], all on a customized small-scale quadrotor UAV (280 mm wheelbase). The RGB camera is not used in our algorithm, but only for visualization. Video is available at https://youtu.be/iYCY6T79oNU.

complete system pipeline and the details of each key component in Section III. The experiments are presented in Section IV, followed by conclusions in Section V.

II. Related Works

Existing works on LiDAR SLAM are extensive. Here we limit our review to the most relevant works: LiDAR-only odometry and mapping, loosely-coupled and tightly-coupled LiDAR-Inertial fusion methods.

A. LiDAR Odometry and Mapping

Besl *et al.* [7] propose an iterated closest points (ICP) method for scan registration, which builds the basis for LiDAR odometry. ICP performs well for dense 3D scans. However, for sparse point clouds of LiDAR measurements, the exact point matching required by ICP rarely exists. To cope with this problem, Segal *et al.* [8] propose a generalized-ICP based on the point-to-plane distance. Then Zhang *et al.* [9] combine this ICP method with a point-to-edge distance and developed a LiDAR odometry and mapping (LOAM) framework. Thereafter, many variants of LOAM have been developed, such as LeGO-LOAM [10] and LOAM-Livox [11]. While these methods work well for structured environments and LiDARs of large FoV, they are very vulnerable to featureless environments or small FoV LiDARs [11].

B. Loosely-Coupled LiDAR-Inertial Odometry

IMU measurements are commonly used to mitigate the problem of LiDAR degeneration in featureless environments. Loosely-coupled LiDAR-inertial odometry (LIO) methods typically process the LiDAR and IMU measurements separately and fuse their results later. For example, IMU-aided LOAM [9] takes the pose integrated from IMU measurements as the initial estimate for LiDAR scan registration. Zhen *et al.* [12] fuse the IMU measurements and the Gaussian Particle Filter output of LiDAR measurements using the error-state EKF. Balazadegan *et al.* [13] add the IMU-gravity model to estimate the 6-DOF ego-motion to aid the LiDAR scan registration. Zuo *et al.* [14] use a Multi-State Constraint Kalman Filter (MSCKF) to fuse the scan registration results with IMU and visual measurements. A common procedure of the loosely-coupled approach is obtaining a pose measurement by registering a new scan and then fusing the pose measurement with IMU measurements. The separation between scan registration and data fusion reduces the computation load. However, it ignores the correlation between the system's other states (e.g., velocity) and the pose of the new scan. Moreover, in the case of featureless environments, the scan registration could degenerate in certain directions and causes unreliable fusion in later stages.

C. Tightly-Coupled LiDAR-Inertial Odometry

Unlike the loosely-coupled methods, tightly-coupled LiDAR-inertial odometry methods typically fuse the raw feature points (instead of scan registration results) of LiDAR with IMU data. There are two main approaches to tightly-coupled LIO: optimization-based and filter-based. Geneva *et al.* [15] use a graph optimization with IMU pre-integration constrains [16] and plane constraints [17] from LiDAR feature points. Recently, Ye *et al.* [18] propose the LIOM package which uses a similar graph optimization but is based on edge and plane features. For filter-based methods, Bry [19] uses a Gaussian Particle Filter (GPF) to fuse the data of IMU and a planar 2D LiDAR. This method has also been used in the Boston Dynamics Atlas humanoid robot. Since the computation complexity of particle filter grows quickly with the number of feature points and the system dimension, Kalman filter and its variants are usually more preferred, such as extended Kalman filter [20], unscented Kalman filter [21], and iterated Kalman filter [22].

Our method falls into the tightly-coupled approach. We adopt an iterated extended Kalman filter similar to [22] to mitigate linearization errors. Kalman filter (and its variants) has a time complexity $\mathcal{O}(m^2)$ where m is the measurement dimension [23], this may lead to remarkably high computation load when dealing with a large number of LiDAR measurements. Naive down-sampling would reduce the number of measurements but at the cost of information loss. [22] reduces the number of measurements by extracting and fitting ground planes similar to [10]. This, however, does not apply to aerial applications where the ground plane may not always present.

III. Methodology

A. Framework Overview

This letter will use the notations shown in Table I. The overview of our workflow is shown in Fig. 2. The LiDAR inputs are fed into the feature extraction module to obtain planar and edge features. Then the extracted features and IMU measurements are fed into our state estimation module for state estimation at $10\,\text{Hz} - 50\,\text{Hz}$. The estimated pose then registers the feature points to the global frame and merges them with the

[2]https://www.livoxtech.com/de/avia
[3]https://www.dji.com/cn/manifold-2/specs

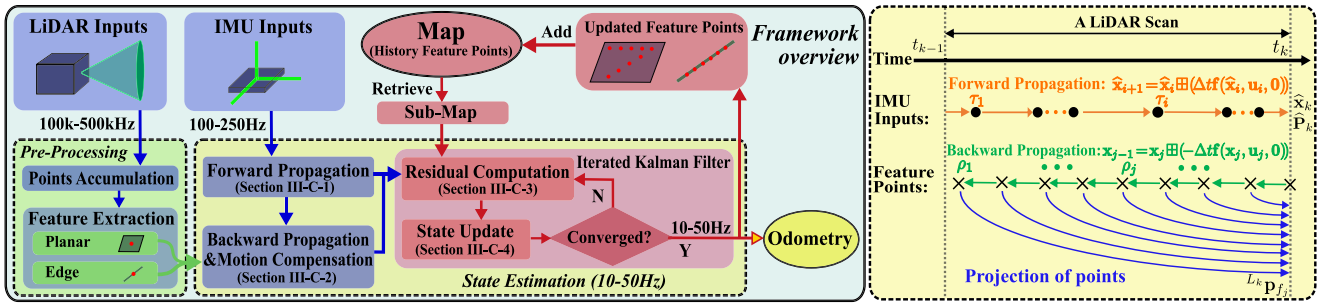

Fig. 2. System overview of FAST-LIO. (a): the overall pipeline; (b): the forward and backward propagation.

feature points map built so far. The updated map is finally used to register further new points in the next step.

B. System Description

1) $\boxplus/\boxminus$ Operator: Let $\mathcal{M}$ be the manifold of dimension n in consideration (e.g., $\mathcal{M}=SO(3)$). Since manifolds are locally homeomorphic to $\mathbb{R}^n$, we can establish a bijective mapping from a local neighborhood on $\mathcal{M}$ to its tangent space $\mathbb{R}^n$ via two encapsulation operators $\boxplus$ and $\boxminus$ [24]:

$$\boxplus:\mathcal{M}\times\mathbb{R}^n\to\mathcal{M};\qquad \boxminus:\mathcal{M}\times\mathcal{M}\to\mathbb{R}^n$$

$$\mathcal{M}=SO(3):\ \mathbf{R}\boxplus\mathbf{r}=\mathbf{R}\mathrm{Exp}(\mathbf{r});\quad \mathbf{R}_1\boxminus\mathbf{R}_2=\mathrm{Log}(\mathbf{R}_2^{\mathrm{T}}\mathbf{R}_1)$$

$$\mathcal{M}=\mathbb{R}^n:\quad \mathbf{a}\boxplus\mathbf{b}=\mathbf{a}+\mathbf{b};\qquad \mathbf{a}\boxminus\mathbf{b}=\mathbf{a}-\mathbf{b}$$

where $\mathrm{Exp}(\mathbf{r})=\mathbf{I}+\frac{\mathbf{r}}{\|\mathbf{r}\|}\sin(\|\mathbf{r}\|)+\frac{\mathbf{r}^2}{\|\mathbf{r}\|^2}(1-\cos(\|\mathbf{r}\|))$ is the exponential map [24] and $\mathrm{Log}(\cdot)$ is its inverse map. For a compound manifold $\mathcal{M}=SO(3)\times\mathbb{R}^n$ we have:

$$\begin{bmatrix}\mathbf{R}\\ \mathbf{a}\end{bmatrix}\boxplus\begin{bmatrix}\mathbf{r}\\ \mathbf{b}\end{bmatrix}=\begin{bmatrix}\mathbf{R}\boxplus\mathbf{r}\\ \mathbf{a}+\mathbf{b}\end{bmatrix};\quad \begin{bmatrix}\mathbf{R}_1\\ \mathbf{a}\end{bmatrix}\boxminus\begin{bmatrix}\mathbf{R}_2\\ \mathbf{b}\end{bmatrix}=\begin{bmatrix}\mathbf{R}_1\boxminus\mathbf{R}_2\\ \mathbf{a}-\mathbf{b}\end{bmatrix}$$

From the above definition, it is easy to verify that

$$(\mathbf{x}\boxplus\mathbf{u})\boxminus\mathbf{x}=\mathbf{u};\ \mathbf{x}\boxplus(\mathbf{y}\boxminus\mathbf{x})=\mathbf{y};\ \forall\mathbf{x},\mathbf{y}\in\mathcal{M},\ \forall\mathbf{u}\in\mathbb{R}^n.$$

2) Continuous Model: Assuming an IMU is rigidly attached to the LiDAR with a known extrinsic ${}^I\mathbf{T}_L=({}^I\mathbf{R}_L,\ {}^I\mathbf{p}_L)$. Taking the IMU frame (denoted as I) as the body frame of reference leads to a kinematic model:

$$\begin{aligned}&{}^G\dot{\mathbf{p}}_I={}^G\mathbf{v}_I,\ {}^G\dot{\mathbf{v}}_I={}^G\mathbf{R}_I\left(\mathbf{a}_m-\mathbf{b}_\mathbf{a}-\mathbf{n}_\mathbf{a}\right)+{}^G\mathbf{g},\ {}^G\dot{\mathbf{g}}=\mathbf{0}\\ &{}^G\dot{\mathbf{R}}_I={}^G\mathbf{R}_I\lfloor\boldsymbol{\omega}_m-\mathbf{b}_\boldsymbol{\omega}-\mathbf{n}_\boldsymbol{\omega}\rfloor_\wedge,\ \dot{\mathbf{b}}_\boldsymbol{\omega}=\mathbf{n}_{\mathbf{b}\boldsymbol{\omega}},\ \dot{\mathbf{b}}_\mathbf{a}=\mathbf{n}_\mathbf{ba}\end{aligned}\tag{1}$$

where ${}^G\mathbf{p}_I$, ${}^G\mathbf{R}_I$ are the position and attitude of IMU in the global frame (i.e., the first IMU frame, denoted as G), ${}^G\mathbf{g}$ is the unknown gravity vector in the global frame, $\mathbf{a}_m$ and $\boldsymbol{\omega}_m$ are IMU measurements, $\mathbf{n}_\mathbf{a}$ and $\mathbf{n}_\boldsymbol{\omega}$ are the white noise of IMU measurements, $\mathbf{b}_\mathbf{a}$ and $\mathbf{b}_\boldsymbol{\omega}$ are the IMU bias modelled as the random walk process with Gaussian noises $\mathbf{n}_\mathbf{ba}$ and $\mathbf{n}_{\mathbf{b}\boldsymbol{\omega}}$, and the notation $\lfloor\mathbf{a}\rfloor_\wedge$ denotes the skew-symmetric matrix of vector $\mathbf{a}\in\mathbb{R}^3$ that maps the cross product operation.

3) Discrete Model: Based on the $\boxplus$ operation defined above, we can discretize the continuous model in (1) at the IMU sampling period Δt using a zero-order holder. The resultant discrete model is

$$\mathbf{x}_{i+1}=\mathbf{x}_i\boxplus\left(\Delta t\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{w}_i)\right)\tag{2}$$

where i is the index of IMU measurements, the function $\mathbf{f}$, state $\mathbf{x}$, input $\mathbf{u}$ and noise $\mathbf{w}$ are defined as below:

$$\mathcal{M}=SO(3)\times\mathbb{R}^{15},\ \dim(\mathcal{M})=18$$

$$\mathbf{x}\doteq\begin{bmatrix}{}^G\mathbf{R}_I^T & {}^G\mathbf{p}_I^T & {}^G\mathbf{v}_I^T & \mathbf{b}_\boldsymbol{\omega}^T & \mathbf{b}_\mathbf{a}^T & {}^G\mathbf{g}^T\end{bmatrix}^T\in\mathcal{M}$$

$$\mathbf{u}\doteq\begin{bmatrix}\boldsymbol{\omega}_m^T & \mathbf{a}_m^T\end{bmatrix}^T,\ \mathbf{w}\doteq\begin{bmatrix}\mathbf{n}_\boldsymbol{\omega}^T & \mathbf{n}_\mathbf{a}^T & \mathbf{n}_{\mathbf{b}\boldsymbol{\omega}}^T & \mathbf{n}_\mathbf{ba}^T\end{bmatrix}^T$$

$$\mathbf{f}\left(\mathbf{x}_i,\mathbf{u}_i,\mathbf{w}_i\right)=\begin{bmatrix}\boldsymbol{\omega}_{m_i}-\mathbf{b}_{\boldsymbol{\omega}_i}-\mathbf{n}_{\boldsymbol{\omega}_i}\\ {}^G\mathbf{v}_{I_i}\\ {}^G\mathbf{R}_{I_i}\left(\mathbf{a}_{m_i}-\mathbf{b}_{\mathbf{a}_i}-\mathbf{n}_{\mathbf{a}_i}\right)+{}^G\mathbf{g}_i\\ \mathbf{n}_{\mathbf{b}\boldsymbol{\omega}_i}\\ \mathbf{n}_{\mathbf{ba}_i}\\ \mathbf{0}_{3\times1}\end{bmatrix}\tag{3}$$

4) Preprocessing of LiDAR Measurements: LiDAR measurements are point coordinates in its local body frame. Since the raw LiDAR points are sampled at a very high rate (e.g., 200 kHz), it is usually not possible to process each new point once being received. A more practical approach is to accumulate these points for a certain time and process them all at once. In FAST-LIO, the minimum accumulation interval is set to 20 ms, leading to up to 50 Hz full state estimation (i.e., odometry output) and map update as shown in Fig. 2 (a). Such an accumulated set of points is called a scan, and the time for processing it is denoted as t_k (see Fig. 2 (b)). From the raw points, we extract planar points with high local smoothness [9] and edge points with low local smoothness as in [11]. Assume the number of feature points is m, each is sampled at time $\rho_j\in(t_{k-1},t_k]$ and is denoted as ${}^{L_j}\mathbf{p}_{f_j}$, where L_j is the LiDAR local frame at the time ρ_j. During a LiDAR scan, there are also multiple IMU measurements, each sampled at time $\tau_i\in[t_{k-1},t_k]$ with the respective state $\mathbf{x}_i$ as in (2). Notice that the last LiDAR feature point is the end of a scan, i.e., $\rho_m=t_k$, while the IMU measurements may not necessarily be aligned with the start or end of the scan.

C. State Estimation

To estimate the states in the state formulation (2), we use an iterated extended Kalman filter. Moreover, we characterize the estimation covariance in the tangent space of the state estimate as in [24], [25]. Assume the optimal state estimate of the last LiDAR scan at t_{k-1} is $\bar{\mathbf{x}}_{k-1}$ with covariance matrix $\bar{\mathbf{P}}_{k-1}$. Then $\bar{\mathbf{P}}_{k-1}$ represents the covariance of the random error state vector defined below:

$$\widetilde{\mathbf{x}}_{k-1} \doteq \mathbf{x}_{k-1} \boxminus \bar{\mathbf{x}}_{k-1} = \begin{bmatrix} \delta\boldsymbol{\theta}^T & {}^G\widetilde{\mathbf{p}}_I^T & {}^G\widetilde{\mathbf{v}}_I^T & \widetilde{\mathbf{b}}_{\boldsymbol{\omega}}^T & \widetilde{\mathbf{b}}_{\mathbf{a}}^T & {}^G\widetilde{\mathbf{g}}^T \end{bmatrix}^T$$

where $\delta\boldsymbol{\theta} = \mathrm{Log}({}^G\bar{\mathbf{R}}_I^T {}^G\mathbf{R}_I)$ is the attitude error and the rests are standard additive errors (i.e., the error in the estimate $\bar{\mathbf{x}}$ of a quantity $\mathbf{x}$ is $\widetilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$). Intuitively, the attitude error $\delta\boldsymbol{\theta}$ describes the (small) deviation between the true and the estimated attitude. The main advantage of this error definition is that it allows us to represent the attitude uncertainty by the 3×3 covariance matrix $\mathbb{E}\{\delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^T\}$. Since the attitude has 3 ° of freedom (DOF), this is a minimal representation.

1) Forward Propagation: The forward propagation is performed once receiving an IMU input (see Fig. 2). More specifically, the state is propagated following (2) by setting the process noise $\mathbf{w}_i$ to zero:

$$\widehat{\mathbf{x}}_{i+1} = \widehat{\mathbf{x}}_i \boxplus (\Delta t \mathbf{f}(\widehat{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{0}))\,;\ \widehat{\mathbf{x}}_0 = \bar{\mathbf{x}}_{k-1}. \tag{4}$$

To propagate the covariance, we use the error state dynamic model obtained below:

$$\begin{aligned} \widetilde{\mathbf{x}}_{i+1} &= \mathbf{x}_{i+1} \boxminus \widehat{\mathbf{x}}_{i+1} \\ &= (\mathbf{x}_i \boxplus \Delta t \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)) \boxminus (\widehat{\mathbf{x}}_i \boxplus \Delta t \mathbf{f}(\widehat{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{0})) \\ &\overset{(23)}{\simeq} \mathbf{F}_{\widetilde{\mathbf{x}}} \widetilde{\mathbf{x}}_i + \mathbf{F}_{\mathbf{w}} \mathbf{w}_i. \end{aligned} \tag{5}$$

The matrix $\mathbf{F}_{\widetilde{\mathbf{x}}}$ and $\mathbf{F}_{\mathbf{w}}$ in (5) is computed following the Appendix. V-A. The result is shown in (7) shown at the bottom of this page, where $\widehat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_{m_i} - \widehat{\mathbf{b}}_{\boldsymbol{\omega}_i}$, $\widehat{\mathbf{a}}_i = \mathbf{a}_{m_i} - \widehat{\mathbf{b}}_{\mathbf{a}_i}$ and $\mathbf{A}(\mathbf{u})^{-1}$ follows the same definition in [26] as below:

$$\begin{aligned} \mathbf{A}(\mathbf{u})^{-1} &= \mathbf{I} - \tfrac{1}{2}\lfloor \mathbf{u} \rfloor_\wedge + (1 - \boldsymbol{\alpha}(\|\mathbf{u}\|)) \tfrac{\lfloor \mathbf{u} \rfloor_\wedge^2}{\|\mathbf{u}\|^2} \\ \alpha(\mathrm{m}) &= \tfrac{\mathrm{m}}{2} \cot\left(\tfrac{\mathrm{m}}{2}\right) = \tfrac{\mathrm{m}}{2} \tfrac{\cos(\mathrm{m}/2)}{\sin(\mathrm{m}/2)} \end{aligned} \tag{6}$$

Denoting the covariance of white noises $\mathbf{w}$ as $\mathbf{Q}$, then the propagated covariance $\widehat{\mathbf{P}}_i$ can be computed iteratively following the below equation.

$$\widehat{\mathbf{P}}_{i+1} = \mathbf{F}_{\widetilde{\mathbf{x}}} \widehat{\mathbf{P}}_i \mathbf{F}_{\widetilde{\mathbf{x}}}^T + \mathbf{F}_{\mathbf{w}} \mathbf{Q} \mathbf{F}_{\mathbf{w}}^T;\ \widehat{\mathbf{P}}_0 = \bar{\mathbf{P}}_{k-1}. \tag{8}$$

The propagation continues until reaching the end time of a new scan at t_k where the propagated state and covariance are denoted as $\widehat{\mathbf{x}}_k, \widehat{\mathbf{P}}_k$. Then $\widehat{\mathbf{P}}_k$ represents the covariance of the error between the ground-truth state $\mathbf{x}_k$ and the state propagation $\widehat{\mathbf{x}}_k$ (i.e., $\mathbf{x}_k \boxminus \widehat{\mathbf{x}}_k$).

2) Backward Propagation and Motion Compensation: When the points accumulation time interval is reached at time t_k, the new scan of feature points should be fused with the propagated state $\widehat{\mathbf{x}}_k$ and covariance $\widehat{\mathbf{P}}_k$ to produce an optimal state update. However, although the new scan is at time t_k, the feature points are measured at their respective sampling time $\rho_j \le t_k$ (see Section. III-B4 and Fig. 2 (b)), causing a mismatch in the body frame of reference.

To compensate the relative motion (i.e., motion distortion) between time ρ_j and time t_k, we propagate (2) backward as $\check{\mathbf{x}}_{j-1} = \check{\mathbf{x}}_j \boxplus (-\Delta t \mathbf{f}(\check{\mathbf{x}}_j, \mathbf{u}_j, \mathbf{0}))$, starting from zero pose and rest states (e.g., velocity and bias) from $\widehat{\mathbf{x}}_k$. The backward propagation is performed at the frequency of feature point, which is usually much higher than the IMU rate. For all the feature points sampled between two IMU measurements, we use the left IMU measurement as the input in the back propagation. Furthermore, noticing that the last three block elements (corresponding to the gyro bias, accelerometer bias, and gravity) of $\mathbf{f}(\mathbf{x}_j, \mathbf{u}_j, \mathbf{0})$ (see (3)) are zeros, the back propagation can be reduced to:

$$\begin{aligned} {}^{I_k}\check{\mathbf{p}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{p}}_{I_j} - {}^{I_k}\check{\mathbf{v}}_{I_j}\Delta t, \qquad \text{s.f. } {}^{I_k}\check{\mathbf{p}}_{I_m} = \mathbf{0}; \\ {}^{I_k}\check{\mathbf{v}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{v}}_{I_j} - {}^{I_k}\check{\mathbf{R}}_{I_j}(\mathbf{a}_{m_{i-1}} - \widehat{\mathbf{b}}_{\mathbf{a}_k})\Delta t - {}^{I_k}\widehat{\mathbf{g}}_k \Delta t, \\ &\qquad \text{s.f. } {}^{I_k}\check{\mathbf{v}}_{I_m} = {}^G\widehat{\mathbf{R}}_{I_k}^T {}^G\widehat{\mathbf{v}}_{I_k},\ {}^{I_k}\widehat{\mathbf{g}}_k = {}^G\widehat{\mathbf{R}}_{I_k}^T {}^G\widehat{\mathbf{g}}_k; \\ {}^{I_k}\check{\mathbf{R}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{R}}_{I_j} \mathrm{Exp}((\widehat{\mathbf{b}}_{\boldsymbol{\omega}_k} - \boldsymbol{\omega}_{m_{i-1}})\Delta t),\ \text{s.f. } {}^{I_k}\mathbf{R}_{I_m} = \mathbf{I}, \end{aligned} \tag{9}$$

where $\rho_{j-1} \in [\tau_{i-1}, \tau_i)$ and s.f. means "starting from".

The backward propagation will produce a relative pose between time ρ_j and the scan-end time t_k: ${}^{I_k}\check{\mathbf{T}}_{I_j} = ({}^{I_k}\check{\mathbf{R}}_{I_j}, {}^{I_k}\check{\mathbf{p}}_{I_j})$. This relative pose enables us to project the local measurement ${}^{L_j}\mathbf{p}_{f_j}$ to scan-end measurement ${}^{L_k}\mathbf{p}_{f_j}$ as follows (see Fig. 2):

$${}^{L_k}\mathbf{p}_{f_j} = {}^I\mathbf{T}_L^{-1} {}^{I_k}\check{\mathbf{T}}_{I_j} {}^I\mathbf{T}_L {}^{L_j}\mathbf{p}_{f_j}, \tag{10}$$

where ${}^I\mathbf{T}_L$ is the known extrinsic (see Section. III-B2). Then the projected point ${}^{L_k}\mathbf{p}_{f_j}$ is used to construct a residual in the following section.

3) Residual Computation: With the motion compensation in (10), we can view the scan of feature points $\{{}^{L_k}\mathbf{p}_{f_j}\}$ all sampled at the same time t_k and use it to construct the residual. Assume the current iteration of the iterated Kalman filter is κ, and the corresponding state estimate is $\widehat{\mathbf{x}}_k^\kappa$. When $\kappa = 0$, $\widehat{\mathbf{x}}_k^\kappa = \widehat{\mathbf{x}}_k$, the

$$\mathbf{F}_{\widetilde{\mathbf{x}}} = \begin{bmatrix} \mathrm{Exp}(-\widehat{\boldsymbol{\omega}}_i\Delta t) & \mathbf{0} & \mathbf{0} & -\mathbf{A}(\widehat{\boldsymbol{\omega}}_i\Delta t)^T\Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{I}\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -{}^G\widehat{\mathbf{R}}_{I_i}\lfloor \widehat{\mathbf{a}}_i \rfloor_\wedge \Delta t & \mathbf{0} & \mathbf{I} & \mathbf{0} & -{}^G\widehat{\mathbf{R}}_{I_i}\Delta t & \mathbf{I}\Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \ \mathbf{F}_{\mathbf{w}} = \begin{bmatrix} -\mathbf{A}(\widehat{\boldsymbol{\omega}}_i\Delta t)^T\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -{}^G\widehat{\mathbf{R}}_{I_i}\Delta t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{7}$$

predicted state from the propagation in (4). Then, the feature points $\{{}^{L_k}\mathbf{p}_{f_j}\}$ can be transformed to the global frame as below:

$$ {}^{G}\widehat{\mathbf{p}}_{f_j}^{\kappa} = {}^{G}\widehat{\mathbf{T}}_{I_k}^{\kappa}\,{}^{I}\mathbf{T}_{L}\,{}^{L_k}\mathbf{p}_{f_j};\ j=1,\ldots,m. \tag{11}$$

For each LiDAR feature point, the closest plane or edge defined by its nearby feature points in the map is assumed to be where the point truly belongs to. That is, the residual is defined as the distance between the feature point's estimated global frame coordinate ${}^{G}\widehat{\mathbf{p}}_{f_j}^{\kappa}$ and the nearest plane (or edge) in the map. Denoting $\mathbf{u}_j$ the normal vector (or edge orientation) of the corresponding plane (or edge), on which lying a point ${}^{G}\mathbf{q}_j$, then the residual $\mathbf{z}_j^{\kappa}$ is computed as:

$$ \mathbf{z}_j^{\kappa} = \mathbf{G}_j\left({}^{G}\widehat{\mathbf{p}}_{f_j}^{\kappa} - {}^{G}\mathbf{q}_j\right) \tag{12}$$

where $\mathbf{G}_j = \mathbf{u}_j^T$ for planar features and $\mathbf{G}_j = \lfloor \mathbf{u}_j \rfloor_{\wedge}$ for edge features. The computation of the $\mathbf{u}_j$ and the search of nearby points in the map, which define the corresponding plane or edge, is achieved by building a KD-tree of the points in the most recent map [11]. Moreover, we only consider residuals whose norm is below certain threshold (e.g., $0.5m$). Residuals exceeding this threshold are either outliers or newly observed points.

4) Iterated State Update: To fuse the residual $\mathbf{z}_j^{\kappa}$ computed in (12) with the state prediction $\widehat{\mathbf{x}}_k$ and covariance $\widehat{\mathbf{P}}_k$ propagated from the IMU data, we need to linearize the measurement model that relates the residual $\mathbf{z}_j^{\kappa}$ to the ground-truth state $\mathbf{x}_k$ and measurement noise. The measurement noise originates from the LiDAR ranging and beam-directing noise ${}^{L_j}\mathbf{n}_{f_j}$ when measuring the point ${}^{L_j}\mathbf{p}_{f_j}$. Removing this noise from the point measurement ${}^{L_j}\mathbf{p}_{f_j}$ leads to the true point location

$$ {}^{L_j}\mathbf{p}_{f_j}^{\mathrm{gt}} = {}^{L_j}\mathbf{p}_{f_j} - {}^{L_j}\mathbf{n}_{f_j}. \tag{13}$$

This true point, after projecting to the frame L_k via (10) and then to the global frame with the ground-truth state $\mathbf{x}_k$ (i.e, pose), should lie exactly on the plane (or edge) in the map. That is, plugging (13) into (10), then into (11), and further into (12) should result in zero. i.e.,

$$ \mathbf{0} = \mathbf{h}_j\left(\mathbf{x}_k, {}^{L_j}\mathbf{n}_{f_j}\right) = \mathbf{G}_j\Big({}^{G}\mathbf{T}_{I_k}\,{}^{I_k}\widehat{\mathbf{T}}_{I_j}\,{}^{I}\mathbf{T}_{L}\big({}^{L_j}\mathbf{p}_{f_j} - {}^{L_j}\mathbf{n}_{f_j}\big) - {}^{G}\mathbf{q}_j\Big)$$

Approximating the above equation by its first order approximation made at $\widehat{\mathbf{x}}_k^{\kappa}$ leads to

$$\begin{aligned} \mathbf{0} &= \mathbf{h}_j\left(\mathbf{x}_k, {}^{L_j}\mathbf{n}_{f_j}\right) \simeq \mathbf{h}_j\left(\widehat{\mathbf{x}}_k^{\kappa}, \mathbf{0}\right) + \mathbf{H}_j^{\kappa}\widetilde{\mathbf{x}}_k^{\kappa} + \mathbf{v}_j \\ &= \mathbf{z}_j^{\kappa} + \mathbf{H}_j^{\kappa}\widetilde{\mathbf{x}}_k^{\kappa} + \mathbf{v}_j \end{aligned} \tag{14}$$

where $\widetilde{\mathbf{x}}_k^{\kappa} = \mathbf{x}_k \boxminus \widehat{\mathbf{x}}_k^{\kappa}$ (or equivalently $\mathbf{x}_k = \widehat{\mathbf{x}}_k^{\kappa} \boxplus \widetilde{\mathbf{x}}_k^{\kappa}$), $\mathbf{H}_j^{\kappa}$ is the Jacobin matrix of $\mathbf{h}_j(\widehat{\mathbf{x}}_k^{\kappa} \boxplus \widetilde{\mathbf{x}}_k^{\kappa}, {}^{L_j}\mathbf{n}_{f_j})$ with respect to $\widetilde{\mathbf{x}}_k^{\kappa}$, evaluated at zero, and $\mathbf{v}_j \in \mathcal{N}(\mathbf{0}, \mathbf{R}_j)$ comes from the raw measurement noise ${}^{L_j}\mathbf{n}_{f_j}$.

Notice that the prior distribution of $\mathbf{x}_k$ obtained from the forward propagation in Section. III-C1 is for

$$ \mathbf{x}_k \boxminus \widehat{\mathbf{x}}_k = \left(\widehat{\mathbf{x}}_k^{\kappa} \boxplus \widetilde{\mathbf{x}}_k^{\kappa}\right) \boxminus \widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^{\kappa} \boxminus \widehat{\mathbf{x}}_k + \mathbf{J}^{\kappa}\widetilde{\mathbf{x}}_k^{\kappa} \tag{15}$$

where $\mathbf{J}^{\kappa}$ is the partial differentiation of $(\widehat{\mathbf{x}}_k^{\kappa} \boxplus \widetilde{\mathbf{x}}_k^{\kappa}) \boxminus \widehat{\mathbf{x}}_k$ with respect to $\widetilde{\mathbf{x}}_k^{\kappa}$ evaluated at zero:

$$ \mathbf{J}^{\kappa} = \begin{bmatrix} \mathbf{A}\left({}^{G}\widehat{\mathbf{R}}_{I_k}^{\kappa} \boxminus {}^{G}\widehat{\mathbf{R}}_{I_k}\right)^{-T} & \mathbf{0}_{3\times 15} \\ \mathbf{0}_{15\times 3} & \mathbf{I}_{15\times 15} \end{bmatrix} \tag{16}$$

where $\mathbf{A}(\cdot)^{-1}$ is defined in (6). For the first iteration (i.e., the case of extended Kalman filter), $\widehat{\mathbf{x}}_k^{\kappa} = \widehat{\mathbf{x}}_k$, then $\mathbf{J}^{\kappa} = \mathbf{I}$.

Combining the prior in (15) with the posteriori distribution from (14) yields the maximum a-posteriori estimate (MAP):

$$ \min_{\widetilde{\mathbf{x}}_k^{\kappa}} \left(\|\mathbf{x}_k \boxminus \widehat{\mathbf{x}}_k\|^2_{\widehat{\mathbf{P}}_k^{-1}} + \sum_{j=1}^{m} \|\mathbf{z}_j^{\kappa} + \mathbf{H}_j^{\kappa}\widetilde{\mathbf{x}}_k^{\kappa}\|^2_{\mathbf{R}_j^{-1}} \right) \tag{17}$$

where $\|\mathbf{x}\|^2_{\mathbf{M}} = \mathbf{x}^T\mathbf{M}\mathbf{x}$. Substituting the linearization of the prior in (15) into (17) and optimizing the resultant quadratic cost leads to the standard iterated Kalman filter [22], which can be computed below (to simplify the notation, let $\mathbf{H} = [\mathbf{H}_1^{\kappa^T}, \ldots, \mathbf{H}_m^{\kappa^T}]^T$, $\mathbf{R} = \mathrm{diag}(\mathbf{R}_1, \cdots \mathbf{R}_m)$, $\mathbf{P} = (\mathbf{J}^{\kappa})^{-1}\widehat{\mathbf{P}}_k(\mathbf{J}^{\kappa})^{-T}$, and $\mathbf{z}_k^{\kappa} = [\mathbf{z}_1^{\kappa^T}, \ldots, \mathbf{z}_m^{\kappa^T}]^T$):

$$\begin{aligned} \mathbf{K} &= \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}, \\ \widehat{\mathbf{x}}_k^{\kappa+1} &= \widehat{\mathbf{x}}_k^{\kappa} \boxplus \left(-\mathbf{K}\mathbf{z}_k^{\kappa} - (\mathbf{I} - \mathbf{K}\mathbf{H})(\mathbf{J}^{\kappa})^{-1}\left(\widehat{\mathbf{x}}_k^{\kappa} \boxminus \widehat{\mathbf{x}}_k\right)\right). \end{aligned} \tag{18}$$

The updated estimate $\widehat{\mathbf{x}}_k^{\kappa+1}$ is then used to compute the residual in Section. III-C3 and repeat the process until convergence (i.e., $\|\widehat{\mathbf{x}}_k^{\kappa+1} \boxminus \widehat{\mathbf{x}}_k^{\kappa}\| < \epsilon$). After convergence, the optimal state estimation and covariance is:

$$ \bar{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^{\kappa+1},\ \bar{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\,\mathbf{P} \tag{19}$$

A problem with the commonly used Kalman gain form in (18) is that it requires to invert the matrix $\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}$ which is in the dimension of the measurements. In practice, the number of LiDAR feature points are very large in number, inverting a matrix of this size is prohibitive. As such, existing works [22], [27] only use a small number of measurements. In this letter, we show that this limitation can be avoided. The intuition originates from (17) where the cost function is over the state, hence the solution should be calculated with complexity depending on the state dimension. In fact, if directly solving (17), we can obtain the same solution in (18) but with a new form of Kalman gain shown below:

$$ \mathbf{K} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \mathbf{P}^{-1}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}. \tag{20}$$

We prove in Appendix B that the two forms of Kalman gains are indeed equivalent based on the matrix inverse lemma [28]. Since the LiDAR measurements are independent, the covariance matrix $\mathbf{R}$ is (block) diagonal and hence the new formula only requires to invert two matrices both in the dimension of state instead of measurements. The new formula greatly saves the computation as the state dimension is usually much lower than measurements in LIO (e.g., more than 1000 effective feature points in a scan for 10 Hz scan rate while the state dimension is only 18).

Algorithm 1: State Estimation.

Input: Last optimal estimation $\bar{\mathbf{x}}_{k-1}$ and $\bar{\mathbf{P}}_{k-1}$,
IMU inputs $(\mathbf{a}_m, \boldsymbol{\omega}_m)$ in current scan;
LiDAR feature points ${}^{L_j}\mathbf{p}_{f_j}$ in current scan. Forward propagation to obtain state prediction $\widehat{\mathbf{x}}_k$ via (4) and covariance prediction $\widehat{\mathbf{P}}_k$ via (8);
Backward propagation to obtain ${}^{L_k}\mathbf{p}_{f_j}$ via (9), (10);
$\kappa = -1, \widehat{\mathbf{x}}_k^{\kappa=0} = \widehat{\mathbf{x}}_k$;
repeat $\|\widehat{\mathbf{x}}_k^{\kappa+1} \boxminus \widehat{\mathbf{x}}_k^{\kappa}\| < \epsilon$ $\kappa = \kappa + 1$;
Compute $\mathbf{J}^{\kappa}$ via (16) and $\mathbf{P} = (\mathbf{J}^{\kappa})^{-1}\widehat{\mathbf{P}}_k(\mathbf{J}^{\kappa})^{-T}$;
Compute residual $\mathbf{z}_j^{\kappa}$ (12) and Jocobin $\mathbf{H}_j^{\kappa}$ (14);
Compute the state update $\widehat{\mathbf{x}}_k^{\kappa+1}$ via (18) with the Kalman gain $\mathbf{K}$ from (20);
$\bar{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^{\kappa+1}$; $\bar{\mathbf{P}}_k = (\mathbf{I} - \mathbf{KH})\mathbf{P}$.
Output: Current optimal estimation $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{P}}_k$.

5) The Algorithm: Our state estimation is summarized in Algorithm 1.

D. Map Update

With the state update $\bar{\mathbf{x}}_k$ (hence ${}^{G}\bar{\mathbf{T}}_{I_k} = ({}^{G}\bar{\mathbf{R}}_{I_k}, {}^{G}\bar{\mathbf{p}}_{I_k})$), each feature point $({}^{L_k}\mathbf{p}_{f_j})$ projected to the body frame L_k (see (10)) is then transformed to the global frame via:

$$ {}^{G}\bar{\mathbf{p}}_{f_j} = {}^{G}\bar{\mathbf{T}}_{I_k}\,{}^{I}\mathbf{T}_{L}\,{}^{L_k}\mathbf{p}_{f_j};\ j = 1, \ldots, m. \tag{21}$$

These features points are finally appended to the existing map containing feature points from all previous steps.

E. Initialization

To obtain a good initial estimate of the system state (e.g., gravity vector ${}^{G}\mathbf{g}$, bias, and noise covariance) so to speedup the state estimator, initialization is required. In FAST-LIO, the initialization is simple: keeping the LiDAR static for several seconds (2 seconds for all the experiments in this letter), the collected data is then used to initialize the IMU bias and the gravity vector. If non-repetitive scanning is supported by the LiDAR (e.g., Livox AVIA), keeping static also allows the LiDAR to capture an initial high-resolution map that is beneficial for the subsequent navigation.

IV. Experiment Results

A. Computational Complexity Experiments

In order to validate the computational efficiency of the proposed new formula for computing Kalman gains. We intentionally replace the computation of Kalman gains with the old formula in our system and compare their computation time under the same system pipeline and number of feature points. The results are shown in Table. II. It is obvious that the complexity of the new formula is much lower than the old one.

TABLE I
SOME IMPORTANT NOTATIONS

Symbols	Meaning
t_k	The scan-end time of the k-th LiDAR scan.
τ_i	The i-th IMU sample time in a LiDAR scan.
ρ_j	The j-th feature point's sample time in a LiDAR scan.
I_i, I_j, I_k	The IMU body frame at the time τ_i, ρ_j and t_k.
L_j, L_k	The LiDAR body frame at the time ρ_j and t_k.
$\mathbf{x}, \widehat{\mathbf{x}}, \bar{\mathbf{x}}$	The ground-true, propagated, and updated value of $\mathbf{x}$.
$\tilde{\mathbf{x}}$	The error between ground-true $\mathbf{x}$ and its estimation $\bar{\mathbf{x}}$.
$\widehat{\mathbf{x}}^{\kappa}$	The κ-th update of $\mathbf{x}$ in the iterated Kalman filter.
$\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$	The vector (e.g.,state) $\mathbf{x}$ at time τ_i, ρ_j and t_k.
$\check{\mathbf{x}}_j$	Estimate of $\mathbf{x}_j$ relative to $\mathbf{x}_k$ in the back propagation.

TABLE II
THE RUNNING TIMES OF TWO KALMAN GAIN FORMULAS

Feature Num.	307	717	998	1243	1453	1802
Old Formula (ms)	7.1	23.4	109.3	251	1219	1621
New Formula (ms)	0.07	0.11	0.25	0.37	0.59	1.16

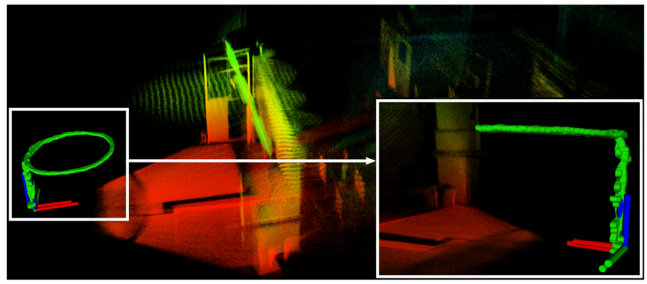

Fig. 3. During the flight experiment, the UAV is automatically flying in a circle path with 1.8 m radius and 1.4 m height. The circle path is conducted repeatedly for 4 times with different periods (6-10 s). The yaw command of the UAV maintains constant during the flight. In the end, the UAV is manually controlled to land at the take-off point, which enables us to measure the drift.

B. UAV Flight Experiments

In order to validate the robustness and computational efficiency of FAST-LIO in actual mobile robots, we build a small-scale quadrotor that can carry a Livox Avia LiDAR with 70° FoV and a DJI Manifold 2-C onboard computer with a 1.8 GHz quad-core Intel i7-8550 U CPU and 8 GB RAM, as shown in Fig. 1. The UAV has only 280 mm wheelbase, and the LiDAR is directly installed on the airframe. The LiDAR-inertial odometry is sent to the flight controller tracking a circle trajectory (Fig. 3). The actual flight experiments show that FAST-LIO can achieve real-time and stable odometry output and mapping in a maximum of 50 Hz for the indoor environment. The flight trajectory and mapping results of 50 Hz frame rate indoor experiment are shown in Fig. 3. The average number of effective feature points and running time is 270 and 6.7 ms, respectively, the drift is smaller than 0.3% (0.08 m drift over 32 m trajectory). The flight video can be found at https://youtu.be/iYCY6T79oNU.

C. Indoor Experiments

Then we test FAST-LIO in a challenging indoor environment with large rotation speeds. In order to generate large rotation, the sensor suite is held on hands and undergoes quickly shaking. Fig. 4 shows the angular velocity and acceleration during the

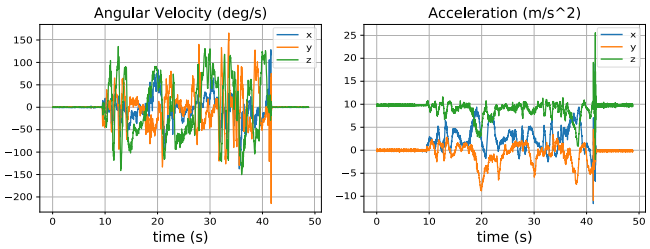

Fig. 4. The angular velocity and acceleration in the indoor experiments.

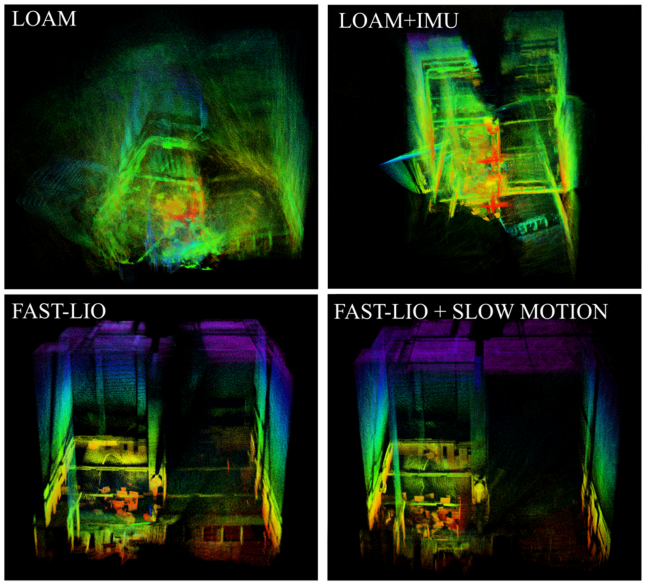

Fig. 5. The Mapping results of different LIO packages in an indoor environment with large rotation speed.

TABLE III
COMPARISON OF PROCESSING TIME FOR A LIDAR SCAN AT 10 HZ

Packages	Num. of effective features	Running time
LOAM	1107	59 ms
LOAM+IMU	1107	44 ms
FAST-LIO	1430	23 ms

experiment. It is seen that the angular velocity often exceeds 100 deg/s. A state of the art implementation of LOAM on Livox LiDARs[4] [11] and LOAM with IMU[5] [9] are also tested as comparisons when the feature extraction are replaced with the one of FAST-LIO. The results show that FAST-LIO can output odometry faster and more stable than others, as shown in Fig. 5 and Table. III . It should be noted that the LOAM+IMU is a loosely-coupled method, hence results in inconsistent mapping. To further verify the mapping result, we perform a second experiment in the same environment but with a much slower motion. The map built by FAST-LIO is shown in the lower-right figure of Fig. 4. Since the two experiments have non-identical movements, it leads to slight visual differences at places occlusions occur. The rest mapping results are very close.

[4]https://github.com/Livox-SDK/livox_mapping
[5]https://github.com/Livox-SDK/livox_horizon_loam

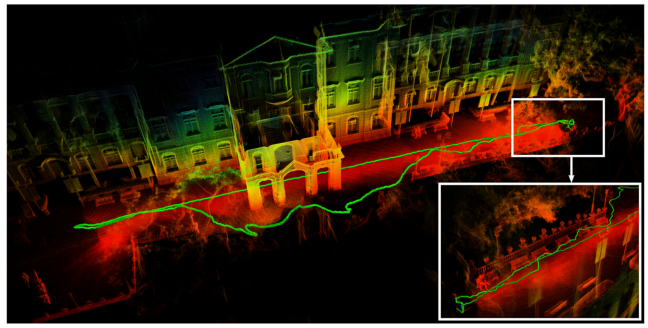

Fig. 6. Mapping results of the Main Building, University of Hong Kong. The LiDAR platform, the same one in Fig. 1, is handheld randomly walking to scan the building. In order to show the drift, the experiment are started and ended at the same place.

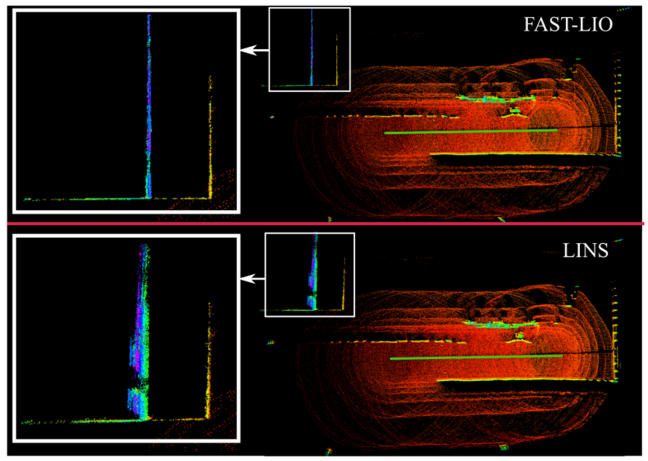

Fig. 7. Comparison between LINS [22] and FAST-LIO. The data is from [22] and is collected by a Velodyne VLP-16 LiDAR and an Xsens MTiG-710 IMU, the green straight line in the center is the odometry output.

D. Outdoor Experiments

Here we show the performance of FAST-LIO in outdoor environments. Fig. 6 shows the mapping results (displaying all raw points) of the Main Building in the University of Hong Kong. The sensor suite is handheld during the data collection and returned to the starting position after traveling around 140m. The drift in this experiment is smaller than 0.05% (0.07 m drift over 140 m trajectory). The scan rate is set to 10 Hz in this experiment, and the average processing time of a scan is 25 ms with average 1497 effective feature points.

Further, we compare FAST-LIO with LINS[6] [22]. To make a fair comparison, we use the dataset from LINS [22], which is a seaport area data collected by a Velodyne VLP-16 and an Xsens MTiG-710 IMU.[6] The results show that the FAST-LIO can achieve better mapping accuracy (see Fig. 7) and only consumes 7.3 ms processing time in average while LINS takes 34.5 ms in average, both running at 10 Hz. It should be noted that since the EKF formula in LINS package has high computational complexity (see Section. III-C4)), it down samples the feature points to average 147 points in a scan (while 784 in a scan for FAST-LIO). This leads to degraded mapping accuracy for

[6]https://github.com/ChaoqinRobotics/LINS---LiDAR-inertial-SLAM

LINS. The result in Fig. 7 shows all the feature points (before down-sample) of FAST-LIO and LINS. All the experiments are conducted on the DJI Manifold2 onboard computer.

V. Conclusion

This letter proposed FAST-LIO, a computationally efficient and robust LiDAR-inertial odometry framework by a tightly-coupled iterated Kalman filter. We used the forward and backward propagation to predict the states and compensate for the motion in a LiDAR scan. Besides, we proved and implemented an equivalent formula that can achieve much lower complexity for the Kalman gain computation. FAST-LIO was tested in the UAV flight experiment, challenging indoor environment with large rotation speed and outdoor environment. In all tests, our method produced precise, real-time, and reliable navigation results.

APPENDIX

A. Computation of $\mathbf{F}_{\widetilde{\mathbf{x}}}$ and $\mathbf{F}_{\mathbf{w}}$

Recall $\mathbf{x}_i = \widehat{\mathbf{x}}_i \boxplus \widetilde{\mathbf{x}}_i$, denote $\mathbf{g}(\widetilde{\mathbf{x}}_i, \mathbf{w}_i) = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)\Delta t = \mathbf{f}(\widehat{\mathbf{x}}_i \boxplus \widetilde{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{w}_i)\Delta t$. Then the error state model (5) is rewriten as:

$$\widetilde{\mathbf{x}}_{i+1} = \underbrace{\left(\left(\widehat{\mathbf{x}}_i \boxplus \widetilde{\mathbf{x}}_i\right) \boxplus \mathbf{g}\left(\widetilde{\mathbf{x}}_i, \mathbf{w}_i\right)\right) \boxminus \left(\widehat{\mathbf{x}}_i \boxplus \mathbf{g}\left(\mathbf{0}, \mathbf{0}\right)\right)}_{\mathbf{G}(\widetilde{\mathbf{x}}_i, \mathbf{g}(\widetilde{\mathbf{x}}_i, \mathbf{w}_i))} \tag{22}$$

Following the chain rule of partial differention, the matrix $\mathbf{F}_{\widetilde{\mathbf{x}}}$ and $\mathbf{F}_{\mathbf{w}}$ in (5) are computed as below.

$$\begin{aligned} \mathbf{F}_{\widetilde{\mathbf{x}}} &= \left(\frac{\partial \mathbf{G}(\widetilde{\mathbf{x}}_i, \mathbf{g}(\mathbf{0},\mathbf{0}))}{\partial \widetilde{\mathbf{x}}_i} + \frac{\partial \mathbf{G}(\mathbf{0}, \mathbf{g}(\widetilde{\mathbf{x}}_i, \mathbf{0}))}{\partial \mathbf{g}(\widetilde{\mathbf{x}}_i, \mathbf{0})} \frac{\partial \mathbf{g}(\widetilde{\mathbf{x}}_i, \mathbf{0})}{\partial \widetilde{\mathbf{x}}_i}\right)\Big|_{\widetilde{\mathbf{x}}_i = \mathbf{0}} \\ \mathbf{F}_{\mathbf{w}} &= \left(\frac{\partial \mathbf{G}(\mathbf{0}, \mathbf{g}(\mathbf{0}, \mathbf{w}_i))}{\partial \mathbf{g}(\mathbf{0}, \mathbf{w}_i)} \frac{\partial \mathbf{g}(\mathbf{0}, \mathbf{w}_i)}{\partial \mathbf{w}_i}\right)\Big|_{\mathbf{w}_i = \mathbf{0}} \end{aligned} \tag{23}$$

B. Equivalent Kalman Gain Formula

Based on the matrix inverse lemma [28], we can get:

$$\left(\mathbf{P}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1} = \mathbf{P} - \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1}\mathbf{H}\mathbf{P}$$

Substituting above into (20), we can get:

$$\begin{aligned} \mathbf{K} &= \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \mathbf{P}^{-1}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1} \\ &= \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} - \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1}\mathbf{H}\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} \end{aligned}$$

Now note that $\mathbf{H}\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} = (\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})\mathbf{R}^{-1} - \mathbf{I}$. Substituting it into above, we can get the standard Kalman gain formula in (18), as shown below.

$$\begin{aligned} \mathbf{K} &= \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} - \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1} \\ &= \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1}. \end{aligned}$$

■

References

[1] K. Sun *et al.*, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 965–972, Apr. 2018.

[2] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3946–3952.

[3] T. Qin, P. Li, and S. Shen, "VINS-MONO: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.* , 2014, pp. 15–22.

[5] D. Wang, C. Watkins, and H. Xie, "Mems mirrors for lidar: A review," *Micromachines*, vol. 11, no. 5, pp. 456–479, 2020.

[6] Z. Liu, F. Zhang, and X. Hong, "Low-cost retina-like robotic lidars based on incommensurable scanning," *IEEE/ASME Trans. Mechatronics*, pp. 1–1, 2021, early access, doi: 10.1109/TMECH.2021.3058173.

[7] P. J. Besl and N. D. McKay, "Method for registration of 3D shapes," in *Sensor fusion IV: control paradigms and data structures*. Int. Soc. Opt. Photonics, vol. 1611, 1992, pp. 586–606.

[8] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot.: Sci. Syst.*, Seattle, WA, vol. 2, no. 4, p. 435, 2009.

[9] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robot.: Sci. Syst.*, vol. 2, no. 9, 2014.

[10] T. Shan and B. Englot, "LEGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[11] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for LiDARs of small FoV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3126–3131.

[12] W. Zhen, S. Zeng, and S. Soberer, "Robust localization and localizability estimation with a rotating laser scanner," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6240–6245.

[13] Y. Balazadegan Sarvrood, S. Hosseinyalamdary, and Y. Gao, "Visual-lidar odometry aided by reduced IMU," *ISPRS Int. J. Geo-inf.*, vol. 5, no. 1, p. 3, 2016.

[14] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "Lic-fusion: Lidar-inertial-camera odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5848–5854.

[15] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "Lips: Lidar-inertial 3 d plane slam," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 123–130.

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

[17] M. Hsiao, E. Westman, and M. Kaess, "Dense planar-inertial SLAM with structural constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6521–6528.

[18] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3 d lidar inertial odometry and mapping," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 3144–3150.

[19] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 1–8.

[20] J. A. Hesch, F. M. Mirzaei, G. L. Mariottini, and S. I. Roumeliotis, "A laser-aided inertial navigation system (l-ins) for human localization in unknown indoor environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 5376–5382.

[21] Z. Cheng *et al.*, "Practical phase unwrapping of interferometric fringes based on unscented Kalman filter technique," *Opt. Exp.*, vol. 23, no. 25, pp. 32337–32349, 2015.

[22] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8899–8906.

[23] M. Raitoharju and R. Piché, "On computational complexity reduction methods for kalman filter extensions," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, no. 10, pp. 2–19, Oct. 2019.

[24] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[25] W. Xu, D. He, Y. Cai, and F. Zhang, "Robots state estimation and observability analysis based on statistical motion models," 2020, *arXiv:2010.05957*.

[26] F. Bullo and R. M. Murray, "Proportional derivative (PD) control on the euclidean group," in *Proc. Eur. Control Conf.*, vol. 2, 1995, pp. 1091–1097.

[27] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.

[28] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA, USA: SIAM, 2002.