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

Wei Xu¹, Fu Zhang¹

Abstract—This paper 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 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 1,200 effective feature points in a scan and completes all iterations of an iEKF step within 25 ms. Our codes are open-sourced online².

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 its 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 small FoV. 2) Due to the high-resolution along the scanning direction, a LiDAR scan usually contains many feature points (e.g. a few thousands). While these feature points are not adequate to reliably determine the pose in case of degeneration, tightly fusing such large number of feature points to IMU measurements requires tremendous computation resources that are not affordable by the UAV onboard computer. 3) Since the LiDAR samples points 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 the motion distortion; 2) To lower the computation load caused by large number of LiDAR feature points, we propose a new formula for computing the Kalman gain and proved its equivalence to the conventional

¹All authors are with Department of Mechanical Engineering, University of Hong Kong. {xuweii, fuzhang}@hku.hk
²https://github.com/hku-mars/FAST_LIO
³https://www.uav.org/dji-drone-LiDAR-introducing-livox-and-avia/
⁴https://www.dji.com/cn/manifold-2/specs
Kalman gain formula. The new formula has a computation complexity depending on the state dimension instead of measurements 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 robustness when fast motion or intense vibration noise exists.

The remaining paper is organized as follows: In Section II, we discuss relevant research works. We give an overview of the complete system pipeline and the details of each key components in Section. III. The experimental results are presented in Section. IV, followed by conclusions in Section. V.

II. RELATED WORKS

Existing works on LiDAR SLAM are extensive, and a full review would be too time-consuming nor necessary. Hence we limit our review to the most relevant work: 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 cloud 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 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 constrains [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 LiDAR 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 Klamm 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 $O(m^2)$ 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

The overview of our workflow is shown in Fig. 2. The LiDAR inputs are fed into the feature extraction module to obtain the planar and edge features following the methods in LOAM [9]. Then the extracted features and IMU measurements are fed into an iterated Kalman filter based state estimator. The state estimator interact with the global map which contains all the history features and output the real-time odometry. When each states estimation finished, the map will also be updated with the new features. The state estimator will be detailed described in Section III-C.

B. System Description

1) Continuous model:

Assuming an IMU is rigidly attached to the LiDAR with a known extrinsic $^T L = (^T R_L, ^p_L)$. Taking the IMU
Frame (denoted as I) as the body frame of reference leads to a kinematic model:

\[ \begin{align*}
  & G_p f = G_{V_f} \quad G_{V_f} = G_{R_f} (a_m - b_a - n_a) + G_{g}, \\
  & G_{R_f} = G_{R_f} |\omega_m - b_\omega - n_\omega|, \quad b_\omega = n_\omega a_b, \quad b_a = n_a b_a \quad (1)
\end{align*} \]

where \( G_p f, G_{R_f} \) are the position and attitude of IMU in the global frame (i.e., the first IMU frame, denoted as \( G \)), \( G_g \) is the unknown gravity vector in the global frame, \( a_m \) and \( \omega_m \) are IMU measurements, \( n_a \) and \( n_\omega \) are the noise of IMU measurements, \( b_a \) and \( b_\omega \) are the IMU bias modeled as the random walk process with Gaussian noises \( n_{ba} \) and \( n_{b_\omega} \), and the notation \( [a]_\wedge \) denotes the skew-symmetric matrix of vector \( a \in \mathbb{R}^3 \).

2) Discrete model:

Based on the \( \mathbb{H} \) operation defined in [24], we can discretize the continuous model in (1) at the IMU sampling period \( \Delta t \) using a zero-order holder. The resultant discrete model is

\[ x_{i+1} = x_i \mathbb{H} (\Delta t f(x_i, u_i, w_i)) \quad (2) \]

where the function \( f \), state \( x \), inputs \( u \) and noise vector \( w \) are defined as below:

\[ \begin{align*}
  & x = [G_{R_f}^T \quad G_p f^T \quad G_{V_f}^T \quad b_\omega^T \quad b_a^T \quad G_g^T]^T, \\
  & u = [\omega_m^T \quad a_m^T]^T, \quad w = [n_{\omega_m}^T \quad n_a^T \quad n_{b_\omega}^T \quad n_{ba}^T]^T, \\
  & f(x_k, u_k, w_k) = [G_{R_f} f_k (a_m - b_a - n_a) + G_{g} G_{R_f} G_{V_f} b_\omega - n_\omega k, \quad \omega_{mk} - b_\omega - n_\omega, \quad \text{etc.}] \quad (3)
\end{align*} \]

3) Measurement model:

The measurements are feature points from the feature extraction module. Since the lidar points are at a very high sample rate (e.g., 200kHz), 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. Such an accumulated set of points is called a scan and the time for processing it is denoted as \( t_k \) (see Fig. 3). During a lidar scan, there are multiple IMU measurements, each sampled at time \( \tau_i \in [t_{k-1}, t_k] \) with the respective state \( x_i \) as in (2). There are also many feature points in a scan. 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 \( L_{j} p_{f_j} \), where \( L_j \) is the lidar frame at the time \( \rho_j \).

C. States Estimator

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 \( t_{k-1} = \tilde{x}_{k-1} \) with covariance matrix \( \tilde{P}_{k-1} \). Then \( \tilde{x}_{k-1} \) represents the covariance of the random error state vector defined below:

\[ x_{k-1} \mathbb{H} (t_{k-1} = \tilde{x}_{k-1} \mathbb{H} (t_{k-1} = \delta \theta^T \quad G_{p_f}^T \quad G_{V_f}^T \quad b_\omega \quad b_a \quad G_g^T)^T \quad (4) \]

where \( \delta \theta = \text{Log}(G_{R_f} G_{R_f} G_{R_f}) \) is the attitude error and the rests are standard additive errors (i.e., the error in the estimate \( \tilde{x} \) of a quantity \( x \) is \( \tilde{x} = x - \tilde{x} \)). Intuitively, the attitude error \( \delta \theta \) 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 \( \times \) 3 covariance matrix \( \mathbb{E} \{ \delta \theta \delta \theta^T \} \). Since the attitude has 3 degree of freedom (DOF), this is a minimal representation.

1) Forward Propagation:
The forward propagation is performed once receiving
an IMU input (see Fig. 3). More specifically, the state is
propagated following (2) by setting the process noise
$w$ an IMU input (see Fig. 3). More specifically, the state is
zero:

$$\hat{x}_{i+1} = \hat{x}_i \oplus (\Delta t f(\hat{x}_i, u_i, 0)); \hat{x}_0 = \hat{x}_{k-1}. \quad (5)$$

To propagate the covariance, we use the error state dy-
namic model obtained below:

$$\tilde{x}_{i+1} = x_{i+1} \oplus \hat{x}_{i+1} = (x_i \oplus \Delta t f(x_i, u_i, w_i)) \oplus (\hat{x}_i \oplus \Delta t f(\hat{x}_i, u_i, 0))$$

(25) $F_x \hat{x}_i + F_w w_i$. \quad (6)

The matrix $F_x$ and $F_w$ in (6) is computed following the
Appendix. A. The result is shown in (7). Denoting the covari-
ance of white noises $w$ as $Q$, then the propagated covari-
ance $\hat{P}_i$ can be computed iteratively following the
below equation.

$$\hat{P}_{i+1} = F_x \hat{P}_i F_x^T + F_w Q F_w^T; \hat{P}_0 = P_{k-1}. \quad (8)$$

The propagation continues until reaching a new scan of
feature points at $t_k$ where the propagated state and covariance
are denoted as $\hat{x}_k, \hat{P}_k$. Then $\hat{P}_k$ represents the covari-
ance of the error between the ground-truth state $x_k$ and the
state propagation $\hat{x}_k$ (i.e., $x_k \oplus \hat{x}_k$).

2) Backward Propagation and Motion Compensation:

When certain number (or time) of feature points are
accumulated into a new scan at time $t_k$, the new scan of
feature points should be fused with the propagated state
$\hat{x}_k$ and covariance $\hat{P}_k$ 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 $\rho_j < t_k$
(see Section. III-B.3), causing a mismatch in the body frame
of reference.

To compensate the relative motion (i.e., motion distortion)
between time $\rho_j$ and time $t_k$, we propagate (2) backward as
$x_{j-1} = x_j \oplus (\Delta t f(x_j, u_j, 0)$ backwards, starting from zero
pose and rests states (e.g., velocity and bias) from $\hat{x}_k$. The
backward propagation is performed at the frequency of lidar
point rate, which is usually much higher than the IMU rate.
Back propagating the full state in (2) is time-consuming due
to the high point rate. In practice, we need only to back-
propagate the pose. Moreover, due to the constant angular
speed during two IMU measurements, back-propagating the
pose between two IMU measurements can be computed in
a closed-form solution. The resultant back-propagation is
usually very fast.

The backward propagation will produce a relative
pose between time $\rho_j$ and scan-end time $t_k$: $\hat{T}_{fj} =
\begin{bmatrix}
I_k & \hat{R}_{fj} & I_k \\
\end{bmatrix}$. This relative pose enables us to project the
local measurement $L_j p_{fj}$ to scan-end measurement $L_k p_{fj}$ as
follows (see Fig. 3):

$$L_k p_{fj} = \hat{T}_j L_j p_{fj}, \quad (9)$$

where $\hat{T}_j$ is the known extrinsic (see Section. III-B.1). Then the projected point $\hat{L}_k p_{fj}$ is used to construct a residual
in the following section.

3) Residual computation:

With the motion compensation in (9), we can view the scan
of feature points $\{L_k p_{fj}\}$ 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 $k$, and the corresponding state
estimate at $t_k$ is $\hat{x}_k$. When $\kappa = 0$, $\hat{x}_k = x_k$, the predicted
state from the propagation in (5). Then, the feature points
$\{L_k p_{fj}\}$ sampled at time $t_k$ can be projected to the global
frame as below:

$$G \hat{p}_{fj} = \hat{T}_j L_j \hat{p}_{fj}; \quad j = 1, \ldots, m. \quad (10)$$

With the coordinate in the global frame $G \hat{p}_{fj}$, we can
search for the corresponding plane (or edge) in the existing
map following the method in [11]. Denoting $u_j$, the normal
vector (or edge orientation) of corresponding plane (or edge), on which lying a point $G \hat{q}_j$, then this feature point
contributes a residual $z^j$ as below:

$$z^j = G_j (G \hat{p}_{fj} - G \hat{q}_j), \quad (11)$$

where $G_j = u_j^T$ for planar features and $G_j = [u_j] \wedge$ for
edge features.

4) Iterated update:

To fuse the residual $z^j$ computed in (11) with the state
prediction $\hat{x}_k$ and covariance $\hat{P}_k$ propagated from the IMU
data, we need to linearize the measurement model that
relates the measurements $z^j$ to the ground-truth state $x_k$ and
measurement noise. The measurement noise originates from
the lidar point noise $L_j n_{fj}$ when measuring each raw feature
point $L_j p_{fj}$. Compensating this measuring noise in the raw
feature point leads to the ground-truth measurement

$$L_j p_{fj}^a = L_j p_{fj} + L_j n_{fj}. \quad (12)$$
Assume the relative pose \( l_k \hat{T}_{l_j} \) obtained from the back propagation is accurate, then the ground-truth measurement in (12) when compensated the ground-truth state \( x_k \) should lie exactly on the plane (or edge). That is, plugging (12) into (9), then into (10), and further into (10) should result in zero. i.e.,

\[
0 = h_j(x_k) = G_j \left( G^T_{l_k} l_k \hat{T}_{l_j} L_j L_j^T (\mathbf{p}_{f_j} + L_j n_{f_j}) - G \mathbf{q}_j \right).
\]  

(13)

Approximating the above equation by its first order approximation made at \( \hat{x}_k^0 \) leads to

\[
0 = h_j(x_k) \approx z_k^c + H_j^c \delta \bar{x}_k^0 + v_j,
\]  

(14)

where \( H_j^c \) is the Jacobin matrix of \( h_j(x) \) in (13) with respect to \( x \), evaluated at \( \hat{x}_k^0 \), \( \delta \bar{x}_k^0 \) is the error between \( x_k \) and \( \hat{x}_k^0 \), and \( v_j \in \mathcal{N}(0, R_j) \) comes from the raw measurement noise \( v_j \). Assume the relative pose \( \hat{x}_k^0 \) is the ground-truth state \( x_k \). If the iterated Kalman filter fully converged, i.e., \( \delta \bar{x}_k^0 = 0 \), then \( J = I \). The obtained state update (19) and covariance update (21) are ready to be used for the next scan.

A problem with the commonly used form in (18) is that it requires to invert the matrix \( HPH^T + R \) which is in the dimension of the measurements. In practice, the number of LiDAR feature points are very large in number (e.g., more than 1,000 effective feature points in a scan), inverting a matrix of this size is prohibitive. As such, existing works [22, 26] only use a small number of measurements.

In this paper, 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:

\[
K = (H^T R^{-1} H + P^{-1})^{-1} H^T R^{-1}.
\]  

(23)

We prove in Appendix B that the two forms of Kalman gains are indeed equivalent based on the matrix inverse lemma [27]. Since the LiDAR measurements are independent, the covariance matrix \( R \) is (block) diagonal and hence the new formula only requires to invert two matrix in the dimension of state in stead of measurements. The new formula greatly saves the computation as the state dimension is usually much lower than measurements in LIO.

D. Initialization

To obtain a good initial estimate of the system state (e.g., gravity vector \( g \), bias, and noise covariance) so to speedup the state estimator, an initialization is required. In FAST-LIO, the initialization is simple: keeping the LiDAR will static for several seconds. The collected data will be used to initialize the the IMU bias and the gravity vector. If non-repetitive scanning is supported by the LiDAR (e.g., Livox AIVA), keeping static also allows the LiDAR to capture a high-resolution initial 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 by the old formula in our system and compare their computation time under the same system pipeline and number of feature points required for the next scan should be for \( \bar{x}_k \mathrel{\boxtimes} x_k \mathrel{\boxtimes} \bar{x}_k \) (see (4)). To fill in the gap, we notice

\[
x_k \mathrel{\boxtimes} \bar{x}_k^c = (x_k \mathrel{\boxtimes} \bar{x}_k) \mathrel{\boxtimes} \bar{x}_k^c = J (\bar{x}_k^{c+1} \mathrel{\boxtimes} \bar{x}_k^c) \bar{x}_k
\]  

(20)

where \( J (\bar{x}_k^{c+1} \mathrel{\boxtimes} \bar{x}_k^c) \) is defined in (16). Hence

\[
\bar{x}_k = J^{-1} (x_k \mathrel{\boxtimes} \bar{x}_k^c),
\]  

(21)

and

\[
\bar{P}_k = J^{-1} (I - KH) PJ^{-T}.
\]  

(22)
points. The results are shown in Fig. 4. It is obvious that the complexity of the new formula is much lower than the old one.

![Running Time Comparison](image)

Fig. 4. The running time comparison between two Kalman gain computation methods.

**B. UAV Flight Experiment**

In order to validate the robustness and computational efficiency of FAST-LIO in actual mobile robots, we build a small-scale quadrotor which can carry a Livox Avia LiDAR with 70° FoV and a DJI Manifold 2-C onboard computer with a 1.8 GHz Intel i7-8550U CPU and 8 G RAM, as shown in Fig. 5. The UAV airframe have only 280 mm wheelbase and the LiDAR is directly installed on the airframe. The actual flight experiments show that FAST-LIO can achieve real-time and stable odometry output and mapping. The flight trajectory and mapping result is shown in Fig. 6. The average number of feature points and average running time is 1294 and 21 ms, also shown in Fig. 10. The drift in this experiment is smaller than 0.3% (0.051 m drift in 19 m trajectory).

![Small-scale Quadrotor Platform](image)

Fig. 5. The small-scale quadrotor platform carrying a Livox AVIA LiDAR and a DJI Manifold 2 onboard computer.

**C. Indoor Experiments**

Here we test FAST-LIO in a challenging indoor environment with large rotation speeds. To generate large movements, the sensor suite is handheld. Fig. 8 shows the angular velocity and acceleration during the experiment. It is seen that the angular velocity often exceeds 100 deg/s. A state of the art implementation of LOAM on Livox LiDARs\(^5\) [9], LOAM with IMU\(^6\) [9], and LINS\(^7\) [22] are also tested as comparisons when the feature extraction are replaced with the one of FAST-LIO. The results shows that FAST-LIO can output odometry faster and stabler than others, as shown in Fig. 7 and Table. I. It should be noted that the LOAM+IMU is a loosely-coupled method, hence results in inconsistent mapping. The LINS package diverges from the beginning. This happens because the EKF formula in LINS package has high computational complexity (see Section. III-C-4)), so it randomly downsamples the feature points and segment the ground plane to decrease the measurement dimension. Since no obvious ground plane presents, the downsampled feature points lead to degeneration in this challenging indoor environment.

![Mapping Results](image)

Fig. 7. The Mapping results of different LIO packages in an indoor environment with large rotation speed.

**D. Outdoor Experiments**

Here we show the performance of FAST-LIO in outdoor environments. Fig. 9 shows the mapping results of the Main Building in University of Hong Kong. The sensor suite is handheld during the data collection. The average processing

\(^5\)https://github.com/Livox-SDK/livox\_mapping
\(^6\)https://github.com/Livox-SDK/livox\_horizon\_loam
\(^7\)https://github.com/ChaoqinRobotics/LINS—LiDAR-inertial-SLAM
time of a scan is \( 23 \text{ ms} \) with average. Further, we also summarize the number of feature points and running time of the above UAV experiment and indoor experiment in Fig. 10. It should be noted that the FAST-LIO is real-time running in the DJI Manifold2 onboard computer for all the experiments.

![Angular Velocity (deg/s) vs time (s)](image1)

![Acceleration (m/s²) vs time (s)](image2)

**Table I**

| Packages   | Num. of effective features | Running time |
|------------|----------------------------|--------------|
| LOAM       | 1107                       | 39 ms        |
| LOAM+IMU   | 1107                       | 44 ms        |
| LINS       | 210                        | Null         |
| FAST-LIO   | 1430                       | 23 ms        |

**Fig. 8.** The angular velocity and acceleration in the indoor experiments.

**Fig. 9.** Mapping results of the Main Building, University of Hong Kong.

**V. CONCLUSION**

This paper proposed FAST-LIO, a computationally efficient and robust LiDAR-inertial odometry framework by 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 \( F_\hat{x} \) and \( F_w \)**

Recall \( \hat{x}_i = \tilde{x}_i \oplus \hat{x}_i \), denote \( g(\tilde{x}_i, w_i) = f(\hat{x}_i, u_i, w_i) \Delta t \). Then the error state model (6) is rewritten as:

\[
\tilde{x}_{i+1} = \left( (\tilde{x}_i \oplus \hat{x}_i) \oplus g(\tilde{x}_i, w_i) \right) \boxplus \left( \tilde{x}_i \oplus g(0, 0) \right) \quad (24)
\]

Following the chain rule of partial differentiation, the matrix \( F_\hat{x} \) and \( F_w \) in (6) are computed as below.

\[
F_\hat{x} = \left. \frac{\partial G(\tilde{x}_i, g(\tilde{x}_i, 0))}{\partial \hat{x}_i} \right|_{\hat{x}_i = 0} = \left( \frac{\partial G(\tilde{x}_i, g(0, 0))}{\partial \tilde{x}_i} + \frac{\partial G(0, g(\tilde{x}_i, 0))}{\partial \tilde{x}_i} \right) \mid_{\tilde{x}_i = 0} \quad (25)
\]

\[
F_w = \left. \frac{\partial G(0, g(0, w_i))}{\partial w_i} \right|_{w_i = 0} = \left( \frac{\partial G(0, g(0, w_i))}{\partial g(0, w_i)} \frac{\partial g(0, w_i)}{\partial w_i} \right) \mid_{w_i = 0}
\]

For the sake of readability, we hide the \( |\tilde{x}_i = 0 \) and \( |w_i = 0 \), then all the partial differentials in (25) are computed as below.

\[
\frac{\partial G(\tilde{x}_i, g(0, 0))}{\partial \tilde{x}_i} = \left[ \begin{array}{c} \exp(-\hat{\omega}_t \Delta t) \ 0 \\ 0 \end{array} \right] I_{15}
\]

\[
\frac{\partial G(0, g(0, w_i))}{\partial w_i} = \left[ \begin{array}{c} A(\hat{\omega}_t \Delta t) ^T \ 0 \\ 0 \end{array} \right] I_{15}
\]

\[
\frac{\partial g(0, 0)}{\partial \tilde{x}_i} = \left[ \begin{array}{c} 0 \ 0 \ -I_3 \ 0 \ 0 \\ 0 \ 0 \ I_3 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \end{array} \right] \Delta t
\]

\[
\frac{\partial g(0, w_i)}{\partial w_i} = \left[ \begin{array}{c} -G \hat{R}_f |\hat{a}_i| \wedge \sum_0 \ 0 \ 0 \ -G \hat{R}_f I_3 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \end{array} \right] \Delta t
\]
\( a_m \sim \hat{b}_n \), and \( A (u)^{-1} \) is computed as below.

\[
A (u)^{-1} = I - \frac{1}{2} \left[ u \right] \Lambda + \left( 1 - \alpha \left( \| u \| \right) \right) \frac{u^2}{\| u \|^2}
\]

\[
\alpha (m) = \frac{m}{2} \cot \left( \frac{m}{2} \right) = \frac{m \cos (m/2)}{2 \sin (m/2)}
\]

**B. Equivalent Kalman Gain formula**

Based on the matrix inverse lemma [27], we can get:

\[
(P^{-1} + H^T R^{-1} H)^{-1} = P - PH (HPH^T + R)^{-1} HP
\]

Substituting above into (23), we can get:

\[
K = (H^T R^{-1} H + P^{-1})^{-1} H^T R^{-1} = PH^T R^{-1} - PH^T (HPH^T + R)^{-1} HPH^T R^{-1}
\]

Now note that \( HPH^T R^{-1} = (HPH^T + R)^{-1} R - I \). Substituting it into (26), we can get the standard Kalman gain formula in (18), as shown below.

\[
K = PH^T R^{-1} - PH^T - PH^T (HPH^T + R)^{-1} = PH^T (HPH^T + R)^{-1} .
\]

**REFERENCES**

[1] K. Sun, K. Mohta, B. Pfister, M. Waterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, “Robust stereo visual inertial odometry for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[2] A. Howard, “Real-time stereo visual odometry for autonomous ground vehicles,” in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2008, pp. 3949–3952.

[3] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, “Svo: Fast semi-direct monocular visual odometry,” in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[5] D. Wang, C. Watkins, and H. Xie, “Mems mirrors for lidar: A review,” *Micromachines*, vol. 11, no. 5, p. 456, 2020.

[6] Z. Liu, F. Zhang, and X. Hong, “Low-cost retina-like robotic lidars based on incommensurable scanning,” *arXiv preprint arXiv:2006.11034*, 2020.

[7] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[8] A. Segal, D. Haehnel, and S. Thrun, “Generalized-icp,” in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[9] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” in *Robotics: Science and Systems*, 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[11] J. Lin and F. Zhang, “Loam-jivox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov,” *arXiv preprint arXiv:1909.06700*, 2019.

[12] W. Zhen, S. Zeng, and S. Soberer, “Robust localization and localizability estimation with a rotating laser scanner,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6240–6245.

[13] Y. Balazadeg Sarvrood, S. Hosseinyalamdary, and Y. Gao, “Visual-lidar odometry aided by reduced imu,” *ISPRS international journal of geo-information*, 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,” *arXiv preprint arXiv:1909.04102*, 2019.

[15] P. Geneva, K. Eckert, Y. Yang, and G. Huang, “Lips: Lidar-inertial 3d plane slam,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 123–130.