Point-LIO: Robust High-Bandwidth LiDAR-Inertial Odometry

Dongjiao He Wei Xu Nan Chen Fanze Kong Chongjian Yuan Fu Zhang*

Dongjiao He and Wei Xu contribute equally to this work. The authors would like to thank Livox Technology for the equipment support during the whole work. They would also like to thank Mr. Yunfan Ren, Mr. Fangcheng Zhu, and Mr. Yihang Li for the help in the experiment.

Dr. Dongjiao He
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong
Email Address: hdj65822@connect.hku.hk
Dr. Wei Xu
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong
Dr. Nan Chen
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong
Dr. Fanze Kong
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong
Dr. Chongjian Yuan
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong
Prof. Fu Zhang
Address: MaRS Laboratory, Department of Mechanical Engineering, The University of Hong Kong Pokfulam, Hong Kong

Keywords: *High bandwidth, aggressive motion, sensor fusion, point-by-point update, simultaneous localization and mapping (SLAM)*

This article presents Point-LIO: a robust and high-bandwidth LiDAR-inertial odometry with the capability to estimate extremely aggressive robotic motions. Building on the incremental k-d tree (ikd-Tree) for efficient map organization and a tightly-coupled on-manifold Kalaman filter for state estimation, Point-LIO has two key novelties that enable a high-bandwidth LiDAR-inertial odometry (LIO). The first one is a point-by-point LIO framework, where the state is updated at each LiDAR point measurement without accumulating them into a frame. This point-by-point update allows an extremely high-frequency odometry output, significantly increases the odometry bandwidth, and also fundamentally removes the artificial in-frame motion distortion in aggressive motions. The second main novelty is a stochastic process-augmented kinematic model which models the IMU measurements as an output, instead of input as in existing filter-based odometry or SLAM systems, of the model. This new modeling method enables accurate localization and reliable mapping for aggressive motions even when IMU measurements are saturated in the middle of the motion. Various real-world indoor and outdoor experiments on solid-state LiDARs with small field-of-view are conducted for performance evaluation. Overall, Point-LIO is able to provide accurate, high-frequency odometry (4-8 kHz) and reliable mapping under severe vibrations and aggressive motions with high angular velocity (75 rad s^{-1}) beyond the IMU measuring ranges. We further conduct an exhaustive benchmark comparison in 12 sequences from a variety of open LiDAR datasets collected by multi-line spinning LiDARs and solid-state LiDARs. Point-LIO achieves consistently comparable accuracy and time consumption, showing that Point-LIO is computationally efficient (e.g., comparable time consumption with fewer computation resources), robust (e.g., reliable pose estimation under various environments), versatile (e.g., applicable to both multi-line spinning and solid-state LiDARs, automobile vehicles, robot cars, UAVs, and hand-held platforms). Finally, two example applications of Point-LIO are demonstrated, one is a racing quadrotor and the other is a self-rotating UAV, both subject to high angular velocities (e.g., 59 rad s^{-1} and 25 rad s^{-1}) beyond the IMU measuring ranges. Our system will be open on Github https://github.com/hku-mars/Point-LIO.git for the benefit of the community.

1 Introduction

Over the past decades, due to the direct, dense, active and accurate measurements of depth, three-dimensional (3-D) light detection and ranging (LiDAR) sensors have been playing an increasingly important role in autonomous applications, such as view-based SLAM [1, 2], robotic exploration and inspection [3, 4], and autonomous driving [5–8]. Recent developments [9, 10] in LiDAR technologies have enabled the commercialization and mass production of lightweight, cost-efficient and high-performance LiDAR sensors,

with the potential to benefit a range of existing and emerging applications such as autonomous navigation [11] and object detection [12, 13].
A fundamental requirement of LiDAR sensors applied in navigation tasks is to provide accurate position estimations for robot control, and consistent and high-rate mappings for timely perception of the environment. By measuring points at an extremely high rate (e.g., millions per second), LiDAR sensors could enable considerably high-rate odometry and mapping, which allows the tracking of extremely high-speed motions. However, existing approaches are all based on a frame architecture similar to vision-based methods, where the points in a frame are processed periodically at a certain frame rate (e.g., 10 Hz). While, in reality, the LiDAR points are sampled sequentially at different time instants, accumulating these points into a frame will introduce artificial motion distortion and adversely affect the mapping result and odometry accuracy. The low frame rate also increases latency in the odometry and limits the attainable bandwidth, where the odometry bandwidth is defined in analogy to the bandwidth of a dynamic system, which is the frequency where the system gain drops below 0.707. An odometry bandwidth represents how fast a motion could be such that the odometry can estimate satisfactorily.
In this work, we address these issues by two key novel techniques: point-by-point state update and stochastic-process-augmented kinematic model. More specifically, our contributions are as follows:

1. We propose a point-wise LiDAR-inertial odometry framework, which fuses a LiDAR point at its actual sampling time without accumulating into a frame. The elimination of points accumulation removes the in-frame motion distortion and allows high odometry output and mapping update at nearly the point sampling rate, which further enables the system to track very fast motions.

2. To further advance the system bandwidth beyond the IMU measuring range, we use a stochastic process model [14] to model the IMU measurements. Then, we augment this model into the system kinematics and treat the IMU measurements as system output. The stochastic process-augmented kinematic model allows the smooth estimation of system state, including angular velocity and linear acceleration, even when IMU saturates.

3. We integrate these two key techniques into a full tightly-coupled LiDAR-inertial odometry system, termed as Point-LIO. The system uses an on-manifold extended Kalman filter to update the system state by fusing each LiDAR point or IMU data at its respective sampling time. By exploiting the system sparsity and linearity, the developed system achieves real-time state estimation even on low-power ARM-based computer onboard a micro aerial vehicle.

4. The developed system is tested in various challenging real-world data collected by an emerging solid-state LiDAR with very small FoV. The results show the ability of Point-LIO on motion distortion compensation, high odometry output rate (4-8 kHz) and high bandwidth (> 150 Hz). The system is also able to estimate states under extremely aggressive motions (of angular velocity more than 75 rad s^{-1}) with saturated IMU measurements after the initial stage. Furthermore, an exhaustive benchmark comparison on 12 sequences from various open LiDAR datasets shows that Point-LIO achieves consistently comparable accuracy and efficiency to other counterparts while costing fewer computation resources. Real-world applications on actual UAVs are finally demonstrated.

The remaining paper is organized as follows: In section 2, we discuss relevant research works. We give an overview of the complete system pipeline in section 3. section 4 presents the system formulation, an EKF-based state estimator, and summarizes the algorithm. The evaluations of the system are presented in section 5 and benchmark comparison on open datasets is reported in section 6. Finally, applications of the system in real-world UAVs are shown in section 7, followed by conclusions in section 8.

2 Related Works

2.1 LiDAR(-inertial) odometry

Many recent works on 3D LiDAR odometry and mapping are based on the LiDAR-odometry and mapping (LOAM) structure [15], where raw LiDAR points are accumulated into a frame (also called a scan) to extract feature points (e.g., edge and plane). The extracted feature points are then registered to the previous scan to produce an odometry output at the scan rate (i.e., 10 Hz), and a few recent scans are accumulated into a small sub-map which is finally registered and merged to the global map at a lower rate (i.e., 1 Hz) to refine the LiDAR pose with respect to the map. The separate structure between scan-to-scan and scan-to-map in LOAM has been adopted in many following-up works, such as Lego-LOAM [16], which considers the constraints arising from the ground during the scan-to-scan match to improve the odometry accuracy, LINS [17], which fuses the IMU data with scan registration, and others such as [18, 19] , which focus on the the improvements of computation efficiency or accuracy.
While the separation between scan-to-scan and scan-to-map can significantly alleviate the computation load required for the odometry, the scan-to-scan registration in odometry often leads to quick drift accumulation. Moreover, scan-to-scan registration requires large overlaps between consecutive scans, which may not be available in small FoV solid-state LiDARs [20]. To address these problems, direct scan-to-map (or scan-to-local map) has been widely adopted, such as those based on point maps [11, 21], (G-)ICP [22], NDT [22], Surfel maps [23–26], or voxel maps [27]. In particular, [20] proposes a parallel scan-to-map method to deal with the small FoV problems of solid-state LiDARs. [28] fuses the IMU measurements into the scan-to-map registration in an efficient iterated Kalman filter framework. A key problem in the scan-to-map framework is how to maintain the map structure such that it can incrementally add points from new scans while allowing efficient queries. To address this problem, [29] proposes an incremental k-d tree, *ikd-Tree*, as the map structure. Benefiting from this efficient incremental mapping structure, the system FAST-LIO2 is able to perform odometry and mapping in real-time at 10 Hz for spinning LiDARs and 100 Hz for solid-state LiDARs, even on low-power ARM-based computers.
One major drawback of scan-to-map registration is that the odometry is estimated at the rate of the scan (or frame), limiting the odometry output frequency at the frame rate. The limited output frequency will cause a delay in the odometry equal to the scan duration. Furthermore, the limited state estimation rate will put an unnecessary upper bound for the odometry bandwidth due to the Nyquist-Shannon sampling theorem. This problem has been partially addressed in Lola-SLAM [30] and LLOL [31], which propose to slice a scan to multiple sub-scans and register each to the map once it is received, achieving an odometry at 160 Hz and 80 Hz, respectively. Compared with these methods (i.e., the scan-to-scan [15–19], the scan-to-map [11, 21–26], and the sub-scan-to-map [30, 31]), our proposed system is a point-to-map framework, which registers each individual point to the map once it is received. This point-to-map framework allows an odometry at the point sampling rate in theory and 4-8 kHz in practice. The unprecedented high-frequency state update reduces the latency down to microseconds while significantly increasing the odometry bandwidth.

2.2 Motion distortion compensation

As mentioned above, existing works on LiDAR (-inertial) odometry and mapping are almost all based on scans (i.e., frames), which will suffer from in-frame motion distortion resulted from the continuous LiDAR motion during a frame. To correct such distortion, compensation methods are often necessary. Most of efforts assume a constant-velocity motion within the frame to compensate the motion distortion, such as in [15, 17–19, 32–39]. The constant-velocity motion assumption is valid when the scan duration is short or the motion is gentle. But for very aggressive motions where the velocity may change during a scan, e.g., in drone aerobatics, the constant-velocity method will often cause large drift or even failures in odometry.
Another popular method for motion compensation is based on continuous-time trajectory optimization, such as those based on B-Splines [26, 40–42] and Gaussian Process model [43–45]. Continuous-time trajectories allow the evaluation of pose at any time instant, hence can compensate the distortion of each individual point. However, continuous-time trajectory optimization is very time-consuming and often

implemented offline [41, 42, 44, 45]. Although there are some online implementations [26, 40, 43], the odometry rate is often low (e.g., $10Hz$) to ensure real-time optimization. Moreover, they require to accumulate sufficient points for reliable trajectory parameter optimization, which introduces considerable odometry latency. The inherent smoothness of continuous-time trajectories also prevents the description of highly aggressive motions experienced by the robot.

Leveraging IMU measurements is another effective method for motion compensation [29, 46]. These methods integrate the LiDAR pose using the IMU data within a frame to undistort the contained points. Due to the high-frequency of IMU measurements (e.g., 200 Hz), the IMU-based motion compensation is quite effective for usual robot motions and even in fast-rolling drone maneuvers [29] (up to 1242 $^\circ$ s^{-1}). However, the method is still limited by the IMU frequency, and also suffer from IMU measurement noises and bias estimation errors.

The ad-hoc motion compensation reviewed above is ultimately due to the frame-based odometry framework in existing methods. In our system, we fuse each individual LiDAR point at its true sampling time instead of accumulating points into a frame. The elimination of frame accumulation fundamentally removes the motion distortion from the very beginning, hence suffering from no drawbacks mentioned above.

2.3 Formulation of inertial measurements

To fuse IMU measurements with LiDAR point registration, two mainstream methods are often used, i.e., loosely-coupled and tightly-coupled. Loosely-coupled methods integrate the IMU measurements to obtain a pose prior estimation and use this prior estimation as an initial pose for the subsequent scan registration [15, 38, 47–49]. On the other hand, tightly-coupled approaches fuse IMU measurements and LiDAR points in a joint optimization. Two implementations have been proposed for tightly-coupled approaches: extended Kalman filter (EKF) based [17, 29, 50, 51] and optimization based [18, 52, 53]. EKF-based methods [17, 29, 50, 51] integrate the IMU measurements in an EKF's propagation procedure to obtain pose estimates, which are subsequently fused with the LiDAR measurements in an EKF update step. In contrast, optimization-based methods pre-integrate the IMU measurements to obtain the relative pose constraints, and then fuse this pre-integrated relative pose constraints with point registration errors [18, 52, 53].

Tightly-coupled methods often have higher robustness and accuracy than loosely-coupled methods. Yet, in all above tightly-coupled methods, IMU data are used as an input of a kinematic model, so it can be propagated in the EKF propagation or pre-integrated for one frame duration [54]. Such EKF propagation or pre-integration would suffer from saturation problems if the robot motion exceeds the IMU measuring range. In [26, 44, 45], IMU data is used to provide measurements of the angular velocity and linear acceleration predicted from the continuous-time trajectory, based on which the trajectory parameters are optimized along with the LiDAR scan registration factors. Viewing IMU data as measurements of the model output could naturally deal with IMU saturation caused by aggressive motions, although this capability is limited by the continuous-time model as reviewed above. Our method is similar to [26, 44, 45] by viewing IMU data as measurements of the model output, but models the robot motion as a stochastic process, which is then augmented with the kinematics. Such stochastic process-augmented kinematic model allows the IMU to update the state along with LiDAR points in an EKF framework. The developed system is able to deal with saturated IMU measurement in extremely aggressive motions, like vibration and high-speed motion. To the best of our knowledge, it has not been demonstrated in any prior work that the LiDAR-inertial systems could work with saturated IMU measurements.

3 System Overview

Our design philosophy is to truthfully recognize that 1) the LiDAR points are sequentially sampled at respective time, instead of as a frame sampled at the same time; 2) IMU data are measurements, instead of the input, of the system. We fuse these two measurements in an on-manifold extended Kalman filter

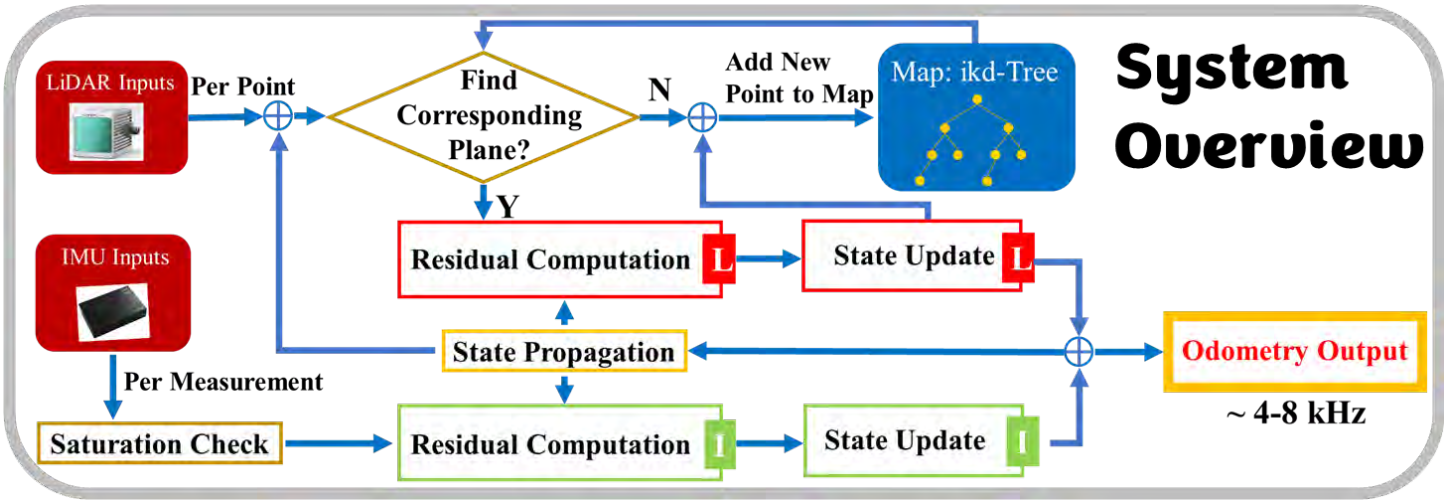


Figure 1: System overview of Point-LIO. $\oplus$ indicates information addition.

framework [55] once the respective measurements (each LiDAR point or IMU data) are received. The overview of our designed system is shown in Figure 1, the sequentially sampled LiDAR points and IMU data are both used to update states at their respective time stamp, leading to an extremely high-rate odometry output, i.e., 4-8 kHz in practice. In particular, for each LiDAR point received, a corresponding plane from the map is searched. If the point is matched with a plane fitted from the points in the map, a residual is computed to update the system state using an on-manifold Kalman filter. The optimized pose finally registers the LiDAR point into global frame and merges to the map, and then proceeds to the next measurement (LiDAR point or IMU data). Otherwise, if the point has no matched plane, it is directly added to the map by the Kalman filter-predicted pose. To enable fast plane correspondence search meanwhile admitting new registered points, we use an incremental k-d tree structure, *ikd-Tree*, originally developed in FAST-LIO2 [29]. For each IMU measurement, the saturation check for each channel of the IMU is conducted separately, the channels that have saturated values would not be used for state update.

4 State Estimation

The state estimation of Point-LIO is a tightly-coupled on-manifold Kalman filter. Here, we briefly explain the essential formulations and workflow of the filter and refer readers to [55] for more detailed and theoretical explanations of the on-manifold Kalman filter.

4.1 Notations

To ease the explanation, we adopt notations as follows:

Symbol	Meaning
$\mathbf{x}_k$	State $\mathbf{x}$ at the k-th measurement sampling time.
$\mathbf{x}$	Ground-true value of state $\mathbf{x}$.
$\hat{\mathbf{x}}$,$\bar{\mathbf{x}}$	Propagated and updated value of state $\mathbf{x}$.
$\delta\mathbf{x}$	Error between ground-true state $\mathbf{x}$ and its estimation $\hat{\mathbf{x}}$.

Furthermore, we introduce two encapsulated operations, $\boxplus$ ("boxplus") and its inverse $\boxminus$ ("boxminus") defined in [55] to describe the system on a manifold $\mathcal{M}$ of dimension n and parameterize the state error in an Euclidean space $\mathbb{R}^n$. Also, these operations can describe the system state space model in discrete

time more compactly. We refer readers to [55] for more detailed definitions and derivations, in this paper, we are only concerned with the manifold $SO(3)$ and $\mathbb{R}^n$:

$$\begin{aligned} \boxplus &: \mathcal{M} \times \mathbb{R}^n \to \mathcal{M};\ \boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n \\ SO(3) &: \mathbf{R} \boxplus \mathbf{r} = \mathbf{R} \cdot \mathrm{Exp}(\mathbf{r});\ \mathbf{R}_1 \boxminus \mathbf{R}_2 = \mathrm{Log}(\mathbf{R}_2^T \cdot \mathbf{R}_1) \\ \mathbb{R}^n &: \mathbf{a} \boxplus \mathbf{b} = \mathbf{a} + \mathbf{b};\ \mathbf{a} \boxminus \mathbf{b} = \mathbf{a} - \mathbf{b} \end{aligned}$$

where $\mathrm{Exp}(\mathbf{r}) = \mathbf{I} + \sin(\|\mathbf{r}\|)\frac{\lfloor \mathbf{r} \rfloor}{\|\mathbf{r}\|} + (1 - \cos(\|\mathbf{r}\|))\frac{\lfloor \mathbf{r} \rfloor^2}{\|\mathbf{r}\|^2}$ is the exponential map on $SO(3)$ and Log is its inverse map. For a compound manifold $\mathcal{M} = SO(3) \times \mathbb{R}^n$ that is the Cartesian product between the two sub-manifolds $\mathcal{M} = SO(3)$ and $\mathbb{R}^n$, we have:

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{a} \end{bmatrix} \boxplus \begin{bmatrix} \mathbf{r} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{R} \boxplus \mathbf{r} \\ \mathbf{a} + \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{a} \end{bmatrix} \boxminus \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 \boxminus \mathbf{R}_2 \\ \mathbf{a} - \mathbf{b} \end{bmatrix}. \tag{1}$$

4.2 Kinematic model

We first derive the system model, which consists of a state transition model and a measurement model.

4.2.1 State transition model

Taking the IMU frame (denoted as I) as the body frame and the first IMU frame as the global frame (denoted as G), the continuous kinematic model is

$$\begin{aligned} &{}^G\dot{\mathbf{R}}_I = {}^G\mathbf{R}_I \lfloor {}^I\boldsymbol{\omega} \rfloor,\ {}^G\dot{\mathbf{p}}_I = {}^G\mathbf{v}_I,\ {}^G\dot{\mathbf{v}}_I = {}^G\mathbf{R}_I {}^I\mathbf{a} + {}^G\mathbf{g},\ {}^G\dot{\mathbf{g}} = \mathbf{0} \\ &\dot{\mathbf{b}}_g = \mathbf{n}_{b_g},\ \dot{\mathbf{b}}_a = \mathbf{n}_{b_a},\ {}^I\dot{\boldsymbol{\omega}} = \mathbf{w}_g,\ {}^I\dot{\mathbf{a}} = \mathbf{w}_a \end{aligned} \tag{2}$$

where ${}^G\mathbf{R}_I$, ${}^G\mathbf{p}_I$ and ${}^G\mathbf{v}_I$ represent the IMU attitude, position and velocity in the global frame, ${}^G\mathbf{g}$ is the gravity vector in the global frame. $\mathbf{b}_g$, $\mathbf{b}_a$ are random-walk IMU biases driven by Gaussian noises $\mathbf{n}_{b_g} \sim \mathcal{N}(\mathbf{0}, \mathcal{Q}_{b_g})$ and $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \mathcal{Q}_{b_a})$, respectively. The notation $\lfloor \mathbf{a} \rfloor$ is the skew-symmetric cross product matrix of $\mathbf{a} \in \mathbb{R}^3$. ${}^I\boldsymbol{\omega}$ and ${}^I\mathbf{a}$ denote the angular velocity and acceleration of IMU in the body frame, i.e., IMU frame. As proposed in [14], a certain robot motion (the angular velocity ${}^I\boldsymbol{\omega}$ and linear acceleration ${}^I\mathbf{a}$) can always be viewed as a sample of a collection or ensemble of signals, which enables us to describe, statistically, the robot motion by a random process. Moreover, as suggested in [14], since the motion of robotic systems usually possesses certain smoothness (e.g., due to actuator delay), quick changes in angular velocities and accelerations are relatively unlikely and a N-th order integrator random process would often suffice the actual use. In particularly, we choose first order integrator models driven by Gaussian noises $\mathbf{w}_g \sim \mathcal{N}(\mathbf{0}, \mathcal{Q}_g)$ and $\mathbf{w}_a \sim \mathcal{N}(\mathbf{0}, \mathcal{Q}_a)$ to model the angular velocity ${}^I\boldsymbol{\omega}$ and linear acceleration ${}^I\mathbf{a}$, respectively.

The continuous model (2) is then discretized at each measurement step k. Denote Δt_k the current measurement interval, which is the time difference between the previous measurement (an IMU data or LiDAR point) and the current measurement (an IMU data or LiDAR point). The continuous model (2) is descretized by assuming the input holds constant for the interval Δt_k, leading to

$$\mathbf{x}_{k+1} = \mathbf{x}_k \boxplus (\Delta t_k \mathbf{f}(\mathbf{x}_k, \mathbf{w}_k)) \tag{3}$$

where the manifold $\mathcal{M}$, function $\mathbf{f}$, state $\mathbf{x}$ and the process noise $\mathbf{w}$ are defined as:

$$\begin{aligned} &\mathcal{M} \triangleq SO(3) \times \mathbb{R}^{21}, \dim(\mathcal{M}) = 24 \\ &\mathbf{x} \triangleq \begin{bmatrix} {}^G\mathbf{R}_I & {}^G\mathbf{p}_I & {}^G\mathbf{v}_I & \mathbf{b}_g & \mathbf{b}_a & {}^G\mathbf{g} & {}^I\boldsymbol{\omega} & {}^I\mathbf{a} \end{bmatrix} \\ &\mathbf{w} \triangleq \begin{bmatrix} \mathbf{n}_{b_g} & \mathbf{n}_{b_a} & \mathbf{w}_g & \mathbf{w}_a \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathcal{Q}) \\ &\mathbf{f}(\mathbf{x}, \mathbf{w}) \triangleq \begin{bmatrix} {}^I\boldsymbol{\omega} & {}^G\mathbf{v}_I & {}^G\mathbf{R}_I {}^I\mathbf{a} + {}^G\mathbf{g} & \mathbf{n}_{b_g} & \mathbf{n}_{b_a} & \mathbf{0}_{3\times 1} & \mathbf{w}_g & \mathbf{w}_a \end{bmatrix} \in \mathbb{R}^{24} \end{aligned} \tag{4}$$

where $\mathcal{Q} = \mathrm{diag}(\mathcal{Q}_{b_g}, \mathcal{Q}_{b_a}, \mathcal{Q}_g, \mathcal{Q}_a)$ is the covariance matrix of the process noise $\mathbf{w}$.

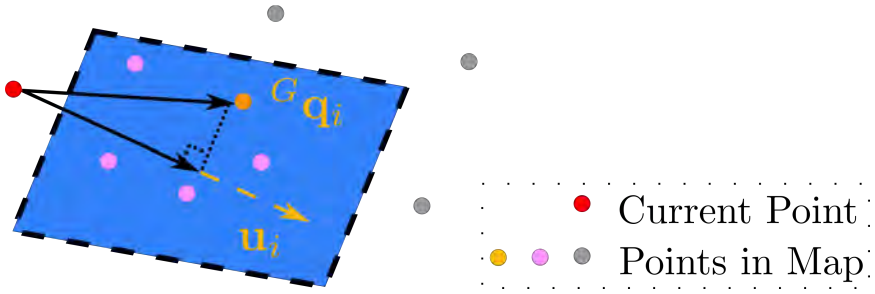


Figure 2: Illustration of direct registration of LiDAR point to map. $^G\mathbf{q}_i$ and $\mathbf{u}_i$ refer to a point in the map and the normal vector of the plane in blue.

4.2.2 Measurement model

The system has two measurements, a LiDAR point or an IMU data (consists of angular velocity and acceleration measurements). These two measurements are often sampled and received by the system at different time, so we model them separately.
Assume the LiDAR frame coincides with the body (i.e., IMU) frame or has pre-calibrated extrinsic, a LiDAR point $^I\mathbf{p}_{m_k}$ is equal as the true position in the local IMU coordinate frame $^I\mathbf{p}_k^{gt}$, which is unknown, contaminated by an additive Gaussian noise $\mathbf{n}_{L_k} \sim \mathcal{N}(\mathbf{0}, \mathcal{R}_{L_k})$:

$$^I\mathbf{p}_{m_k} = {}^I\mathbf{p}_k^{gt} + \mathbf{n}_{L_k} \tag{5}$$

This true point, after projecting to the global frame using the true (yet unknown) IMU pose $^G\mathbf{T}_{I_k} = (^G\mathbf{R}_{I_k}, {}^G\mathbf{p}_{I_k})$, should lie exactly on a local small plane patch in the map (see Figure 2), i.e.,

$$0 = \underbrace{{}^G\mathbf{u}_k^T({}^G\mathbf{T}_{I_k}({}^I\mathbf{p}_{m_k} - \mathbf{n}_{L_k}) - {}^G\mathbf{q}_k)}_{\mathbf{h}_L(\mathbf{x}_k, {}^I\mathbf{p}_{m_k}, \mathbf{n}_{L_k})} \tag{6}$$

where $^G\mathbf{u}_k$ is the normal vector of the corresponding plane and $^G\mathbf{q}_k$ is any point lying on the plane. Note that $^G\mathbf{T}_{I_k}$ is contained in the state vector $\mathbf{x}_k$. (6) imposes an implicit measurement model for the state vector $\mathbf{x}_k$.
The IMU measurement consists of angular velocity measurement ($^I\boldsymbol{\omega}_m$) and acceleration measurement ($^I\mathbf{a}_m$):

$$\begin{bmatrix} {}^I\boldsymbol{\omega}_{m_k} \\ {}^I\mathbf{a}_{m_k} \end{bmatrix} = \underbrace{\begin{bmatrix} {}^I\boldsymbol{\omega}_k + \mathbf{b}_{g_k} + \mathbf{n}_{g_k} \\ {}^I\mathbf{a}_k + \mathbf{b}_{a_k} + \mathbf{n}_{a_k} \end{bmatrix}}_{\mathbf{h}_I(\mathbf{x}_k, \mathbf{n}_{I_k})} \tag{7}$$

where $\mathbf{n}_g \sim \mathcal{N}(\mathbf{0}, \mathcal{R}_g)$, $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \mathcal{R}_a)$ are both Gaussian noises. Collectively, $\mathbf{n}_I = \begin{bmatrix} \mathbf{n}_g^T & \mathbf{n}_a^T \end{bmatrix}^T \sim \mathcal{N}(\mathbf{0}, \mathcal{R}_I) = \mathcal{N}(\mathbf{0}, \mathrm{diag}(\mathcal{R}_g, \mathcal{R}_a))$ is the measurement noise of the IMU. As can be seen, the two states $\boldsymbol{\omega}$, $\mathbf{b}_g$ (and similarly $\mathbf{a}$, $\mathbf{b}_a$), which are separate in the state equation (2) are now correlated in the angular velocity measurement $\boldsymbol{\omega}_m$ (or acceleration measurement $\mathbf{a}$).
To sum, the measurement model of the system could be presented in the following compact form:

$$\begin{aligned} 0 &= \mathbf{h}_L(\mathbf{x}_k, {}^I\mathbf{p}_{m_k}, \mathbf{n}_{L_k}), \\ \begin{bmatrix} {}^I\boldsymbol{\omega}_{m_k} \\ {}^I\mathbf{a}_{m_k} \end{bmatrix} &= \mathbf{h}_I(\mathbf{x}_k, \mathbf{n}_{I_k}). \end{aligned} \tag{8}$$

4.3 Extended Kalman filter

A tightly-coupled extended Kalman filter is used for the state estimation of Point-LIO. The workflow of the EKF is presented in this section.

4.3.1 State Propagation

Assume we have received measurements up to step k and the updated state at that time step is $\bar{\mathbf{x}}_k$ along with the updated covariance matrix $\bar{\mathbf{P}}_k$. The state propagation from step k to next measurement step $k+1$ follows directly the state transition model in equation (3) by setting $\mathbf{w}_k = \mathbf{0}$:

$$\hat{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k \boxplus (\Delta t_k \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{0})) \tag{9}$$

And the covariance is propagated as:

$$\hat{\mathbf{P}}_{k+1} = \mathbf{F}_{\mathbf{x}_k} \bar{\mathbf{P}}_k \mathbf{F}_{\mathbf{x}_k}^T + \mathbf{F}_{\mathbf{w}_k} \mathcal{Q}_k \mathbf{F}_{\mathbf{w}_k}^T \tag{10}$$

where $\mathcal{Q}_k$ is the covariance of the process noise $\mathbf{w}_k$, and the matrices $\mathbf{F}_{\mathbf{x}_k}$, $\mathbf{F}_{\mathbf{w}_k}$ are computed as:

$$\begin{aligned}
\mathbf{F}_{\mathbf{x}_k} &= \frac{\partial(\mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1})}{\partial \delta \mathbf{x}_k}\Big|_{\delta\mathbf{x}_k=\mathbf{0}, \mathbf{w}_k=\mathbf{0}} = \frac{\partial\Big(\big((\bar{\mathbf{x}}_k \boxplus \delta\mathbf{x}_k) \boxplus (\Delta t_k \mathbf{f}(\bar{\mathbf{x}}_k \boxplus \delta\mathbf{x}_k, \mathbf{0}))\big) \boxminus \big(\bar{\mathbf{x}}_k \boxplus (\Delta t_k \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{0}))\big)\Big)}{\partial \delta \mathbf{x}_k} \\
&= \begin{bmatrix}
\mathbf{F}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t_k & \mathbf{0} \\
\mathbf{0} & \mathbf{I} & \mathbf{I}\Delta t_k & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{F}_{31} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t_k & \mathbf{0} & \mathbf{F}_{38} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \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{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\
\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{0} & \mathbf{0} & \mathbf{I}
\end{bmatrix} \\
\mathbf{F}_{\mathbf{w}_k} &= \frac{\partial(\mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1})}{\partial \mathbf{w}_k}\Big|_{\delta\mathbf{x}_k=\mathbf{0}, \mathbf{w}_k=\mathbf{0}} = \frac{\partial\Big(\big(\bar{\mathbf{x}}_k \boxplus (\Delta t_k \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{w}_k))\big) \boxminus \big(\bar{\mathbf{x}}_k \boxplus (\Delta t_k \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{0}))\big)\Big)}{\partial \mathbf{w}_k} = \begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}
\end{bmatrix},
\end{aligned} \tag{11}$$

where $\mathbf{x}_{k+1}$ is true value of state vector at time step k+1, and $\mathbf{F}_{11} = \mathrm{Exp}(-{}^I\bar{\boldsymbol{\omega}}_k \Delta t_k)$, $\mathbf{F}_{31} = -{}^G\bar{\mathbf{R}}_{I_k} \lfloor {}^I\bar{\mathbf{a}}_k \rfloor \Delta t_k$, $\mathbf{F}_{38} = {}^G\bar{\mathbf{R}}_{I_k} \Delta t_k$.

4.3.2 Residual computation

LiDAR measurement With the predicted pose ${}^G\hat{\mathbf{T}}_{I_{k+1}} = ({}^G\hat{\mathbf{R}}_{I_{k+1}}, {}^G\hat{\mathbf{p}}_{I_{k+1}})$ from the Kalman propagation (9), we project the measured LiDAR point ${}^I\mathbf{p}_{m_{k+1}}$ to the global frame ${}^G\hat{\mathbf{p}}_{k+1} = {}^G\hat{\mathbf{R}}_{I_{k+1}} {}^I\mathbf{p}_{m_{k+1}} + {}^G\hat{\mathbf{p}}_{I_{k+1}}$ and search its nearest 5 points (within 5 m distance from ${}^G\hat{\mathbf{p}}_{k+1}$) in the map organized by *ikd-Tree*. The found nearest neighbouring points are then used to fit a local small plane patch with normal vector ${}^G\mathbf{u}_{k+1}$ and centroid ${}^G\mathbf{q}_{k+1}$ as shown in the measurement model (see equation (6) and also Figure 2). If the 5 nearest points do not lie on the fitted plane path (i.e., distance of any point to the plane is larger than 0.1 m), current measurement of LiDAR point ${}^G\hat{\mathbf{p}}_{k+1}$ is directly merged into the map without residual computation or state update. Otherwise, if the local plane succeeds to fit, a residual ($\mathbf{r}_{L_{k+1}}$) is calculated according to equation (8) as:

$$\begin{aligned}
\mathbf{r}_{L_{k+1}} &= 0 - \mathbf{h}_L(\hat{\mathbf{x}}_{k+1}, {}^I\mathbf{p}_{m_{k+1}}, \mathbf{0}) = \mathbf{h}_L(\mathbf{x}_{k+1}, {}^I\mathbf{p}_{m_{k+1}}, \mathbf{n}_{L_{k+1}}) - \mathbf{h}_L(\hat{\mathbf{x}}_{k+1}, {}^I\mathbf{p}_{m_{k+1}}, \mathbf{0}) \\
&\approx \mathbf{H}_{L_{k+1}} \delta\mathbf{x}_{k+1} + \mathbf{D}_{L_{k+1}} \mathbf{n}_{L_{k+1}}
\end{aligned} \tag{12}$$

where $\delta\mathbf{x}_{k+1} = \mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1}$ with $\mathbf{x}_{k+1}$ being the true value of state vector at time step $k+1$, and

$$\begin{aligned}
\mathbf{H}_{L_{k+1}} &= \frac{\partial \mathbf{h}_L(\hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}, {}^I\mathbf{p}_{m_{k+1}}, \mathbf{0})}{\partial \delta \mathbf{x}}\Big|_{\delta\mathbf{x}=\mathbf{0}} = \begin{bmatrix} -{}^G\mathbf{u}_{k+1}^T {}^G\hat{\mathbf{R}}_{I_{k+1}} \lfloor {}^I\mathbf{p}_{m_{k+1}} \rfloor & {}^G\mathbf{u}_{k+1}^T & \mathbf{0}_{1\times 18} \end{bmatrix} \\
\mathbf{D}_{L_{k+1}} &= \frac{\partial \mathbf{h}_L(\hat{\mathbf{x}}_{k+1}, {}^I\mathbf{p}_{m_{k+1}}, \mathbf{n})}{\partial \mathbf{n}}\Big|_{\mathbf{n}=\mathbf{0}} = -{}^G\mathbf{u}_{k+1}^T {}^G\hat{\mathbf{R}}_{I_{k+1}}.
\end{aligned} \tag{13}$$

IMU measurement For an IMU measurement, we first assess if any channel of the IMU is saturated by checking the gap between the current measurement and the rated measuring range. If the gap is too small, this channel of IMU measurement is discarded without updating the state. Then, acceleration and angular velocity measurements from unsaturated IMU channels are collected to calculate the IMU residual ($\mathbf{r}_{I_{k+1}}$) according to equation (7) (to simplify the notation, we use all six channel measurements here):

$$\mathbf{r}_{I_{k+1}} = \begin{bmatrix} {}^I\boldsymbol{\omega}_{m_{k+1}} \\ {}^I\mathbf{a}_{m_{k+1}} \end{bmatrix}^T - \mathbf{h}_I(\hat{\mathbf{x}}_{k+1}, \mathbf{0}) = \mathbf{h}_I(\mathbf{x}_{k+1}, \mathbf{n}_{I_{k+1}}) - \mathbf{h}_I(\hat{\mathbf{x}}_{k+1}, \mathbf{0}) = \mathbf{H}_{I_{k+1}}\delta\mathbf{x}_{k+1} + \mathbf{D}_{I_{k+1}}\mathbf{n}_{I_{k+1}} \tag{14}$$

where $\delta\mathbf{x}_{k+1} = \mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1}$ with $\mathbf{x}_{k+1}$ being the true value of state vector at time step $k+1$, and

$$\begin{aligned} \mathbf{H}_{I_{k+1}} &= \tfrac{\partial \mathbf{h}_I(\hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}, \mathbf{0})}{\partial \delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}} = \begin{bmatrix} \mathbf{0}_{6\times 9} & \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 3} & \mathbf{I}_{6\times 6} \end{bmatrix} \\ \mathbf{D}_{I_{k+1}} &= \tfrac{\partial \mathbf{h}_2(\hat{\mathbf{x}}_{k+1}, \mathbf{n})}{\partial \mathbf{n}}|_{\mathbf{n}=\mathbf{0}} = \mathbf{I}_{6\times 6} \end{aligned} \tag{15}$$

To sum, the residual, from either LiDAR point measurement the (12) or IMU measurement (14), is related to the state $\mathbf{x}_{k+1}$ and the respective measurement noise by the following relation:

$$\mathbf{r}_{k+1} \approx \mathbf{H}_{k+1}\delta\mathbf{x}_{k+1} + \mathbf{D}_{k+1}\mathbf{n}_{k+1}, \qquad \mathbf{n}_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathcal{R}_{k+1}). \tag{16}$$

where for a LiDAR point measurement we have $\mathbf{r}_{k+1} = \mathbf{r}_{L_{k+1}}, \mathbf{H}_{k+1} = \mathbf{H}_{L_{k+1}}, \mathbf{D}_{k+1} = \mathbf{D}_{L_{k+1}}, \mathcal{R}_{k+1} = \mathcal{R}_{L_{k+1}}$, and for an IMU measurement we have $\mathbf{r}_{k+1} = \mathbf{r}_{I_{k+1}}, \mathbf{H}_{k+1} = \mathbf{H}_{I_{k+1}}, \mathbf{D}_{k+1} = \mathbf{D}_{I_{k+1}}, \mathcal{R}_{k+1} = \mathcal{R}_{I_{k+1}}$.

4.3.3 State Update

The propagated state $\hat{\mathbf{x}}_{k+1}$ in (9) and covariance $\hat{\mathbf{P}}_{k+1}$ in (10) impose a prior Gaussian distribution for the unknown state $\mathbf{x}_{k+1}$, as follows:

$$\delta\mathbf{x}_{k+1} = \mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{P}}_{k+1}). \tag{17}$$

And the observation model (16) gives another Gaussian distribution for $\delta\mathbf{x}_{k+1}$:

$$\mathbf{D}_{k+1}\mathbf{n}_{k+1} = \mathbf{r}_{k+1} - \mathbf{H}_{k+1}\delta\mathbf{x}_{k+1} \sim \mathcal{N}\left(\mathbf{0}, \bar{\mathcal{R}}_{k+1}\right), \qquad \bar{\mathcal{R}}_{k+1} = \mathbf{D}_{k+1}\mathcal{R}_{k+1}\mathbf{D}_{k+1}^T. \tag{18}$$

Then combining the prior distribution in (17) with the measurement model from (18) yields the posterior distribution of the state $\mathbf{x}_{k+1}$ (which is represented by $\delta\mathbf{x}_{k+1}$ equivalently):

$$\underset{\delta\mathbf{x}_{k+1}}{\operatorname{argmin}} \left(\|\mathbf{r}_{k+1} - \mathbf{H}_{k+1}\delta\mathbf{x}_{k+1}\|^2_{\bar{\mathcal{R}}_{k+1}} + \|\delta\mathbf{x}_{k+1}\|^2_{\hat{\mathbf{P}}_{k+1}} \right) \tag{19}$$

where $\|\mathbf{x}\|^2_{\mathbf{A}} = \mathbf{x}^T\mathbf{A}^{-1}\mathbf{x}$. The optimization problem in (19) is a standard quadratic programming and the optimal solution $\delta\mathbf{x}^o_{k+1}$ can be easily obtained, which is essentially the Kalman update [56]:

$$\begin{aligned} \delta\mathbf{x}^o_{k+1} &= \mathbf{K}_{k+1}\mathbf{r}_{k+1} \\ \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1}\mathbf{S}^{-1}_{k+1} \\ \mathbf{S}_{k+1} &= \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1} + \bar{\mathcal{R}}_{k+1} \\ \mathbf{P}_{k+1} &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\hat{\mathbf{P}}_{k+1} \end{aligned} \tag{20}$$

Then the update of $\mathbf{x}_{k+1}$ is:

$$\bar{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}^o_{k+1}. \tag{21}$$

The updated state will be used in the next step propagation. To do so, we need also to estimate the covariance, denoted by $\bar{\mathbf{P}}_{k+1}$, of the error between the state estimate $\bar{\mathbf{x}}_{k+1}$ and ground-truth $\mathbf{x}_{k+1}$ defined as $\mathbf{x}_{k+1} \boxminus \bar{\mathbf{x}}_{k+1}$.

$$\begin{aligned} \mathbf{x}_{k+1} \boxminus \bar{\mathbf{x}}_{k+1} &= (\hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}_{k+1}) \boxminus \bar{\mathbf{x}}_{k+1} \\ &\approx \underbrace{\left(\hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}^o_{k+1}\right) \boxminus \bar{\mathbf{x}}_{k+1}}_{=\mathbf{0}} + \mathbf{J}_{k+1}(\delta\mathbf{x}_{k+1} - \delta\mathbf{x}^o_{k+1}) \end{aligned} \tag{22}$$

where $\mathbf{J}_{k+1}$ is the projection matrix

$$\mathbf{J}_{k+1} = \frac{\partial((\hat{\mathbf{x}}_{k+1} \boxplus \delta\mathbf{x}) \boxminus \bar{\mathbf{x}}_{k+1})}{\partial \delta\mathbf{x}}\Big|_{\delta\mathbf{x}=\delta\mathbf{x}^o_{k+1}} = \begin{bmatrix} \mathbf{A}(\boldsymbol{\theta}_{k+1})^{-T} & \mathbf{0}_{3\times 21} \\ \mathbf{0}_{3\times 21} & \mathbf{I}_{21\times 21} \end{bmatrix}$$
$$\mathbf{A}(\mathbf{u})^{-1} = \mathbf{I} - \frac{\lfloor \mathbf{u} \rfloor}{2} + (1 - \frac{\|\mathbf{u}\|}{2}\cot(\frac{\|\mathbf{u}\|}{2}))\frac{\lfloor \mathbf{u} \rfloor}{\|\mathbf{u}\|}, \boldsymbol{\theta}_{k+1} = {}^G\bar{\mathbf{R}}_{I_{k+1}} \boxminus {}^G\hat{\mathbf{R}}_{I_{k+1}}. \tag{23}$$

and the covariance of $(\delta\mathbf{x}_{k+1} - \delta\mathbf{x}^o_{k+1})$ is the inversion of the Hessian matrix in (19), which is also the matrix $\mathbf{P}_{k+1}$ in (20). As a result, the covariance of $\mathbf{x}_{k+1} \boxminus \bar{\mathbf{x}}_{k+1}$, according to (22), is:

$$\bar{\mathbf{P}}_{k+1} = \mathbf{J}_{k+1}\mathbf{P}_{k+1}\mathbf{J}^T_{k+1}. \tag{24}$$

The updated state $\bar{\mathbf{x}}_{k+1}$ along with the covariance matrix $\bar{\mathbf{P}}_{k+1}$ are used in propagation of the next measurement. The overall procedure of our state estimation is summarized in Algorithm 1.

Algorithm 1 State Estimation at Step $k+1$

Input:
Last odometry output $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{P}}_k$;
A LiDAR point or an IMU measurement;
Output:
New odometry output $\bar{\mathbf{x}}_{k+1}$, $\bar{\mathbf{P}}_{k+1}$;
Workflow:

State propagation from the last time step k to current time step $k+1$ via (9) and (10) to obtain state prediction $\hat{\mathbf{x}}_{k+1}$ and its covariance $\hat{\mathbf{P}}_{k+1}$.
if *A LiDAR point* **then**
  ${}^G\hat{\mathbf{p}}_{k+1} = {}^G\hat{\mathbf{R}}_{I_{k+1}}{}^I\mathbf{p}_{m_{k+1}} + {}^G\hat{\mathbf{p}}_{I_{k+1}}$;
  if *PlaneCorrespondenceExist(* ${}^G\hat{\mathbf{p}}_{k+1}$*)* **then**
    Compute $\mathbf{r}_{L_{k+1}}$, $\mathbf{H}_{L_{k+1}}$, $\mathbf{D}_{L_{k+1}}$ via (12), (13);
    Compute update state $\bar{\mathbf{x}}_{k+1}$ via (20), (21);
    Compute update covariance $\bar{\mathbf{P}}_{k+1}$ via (23), (24);
    Add the point transformed with the updated state $\bar{\mathbf{x}}_{k+1}$ to the map;
  else
    Add point ${}^G\hat{\mathbf{p}}_{k+1}$ into the map;
  end if
else if *An IMU measurement* **then**
  if *NoSaturation(*${}^I\boldsymbol{\omega}_{m_{k+1}}$, ${}^I\mathbf{a}_{m_{k+1}}$*)* **then**
    Compute $\mathbf{r}_{I_{k+1}}$, $\mathbf{H}_{I_{k+1}}$, $\mathbf{D}_{I_{k+1}}$ via (14), (15);
    Compute update state $\bar{\mathbf{x}}_{k+1}$ via (20), (21);
    Compute update covariance $\bar{\mathbf{P}}_{k+1}$ via (23), (24);
  end if
end if

4.4 Analysis

In our proposed LIO framework, the state is updated by consuming each LiDAR point at its reception, leading to a point-wise odometry. This point-wise architecture is completely different from existing frame-based LiDAR odometry and mapping frameworks [11, 15–17, 21, 29–31] and enables an extremely high-rate odometry output that is ideally equal to the point rate of the LiDAR sensor (from hundreds of thousands to million points per second). The high-rate state update provides timely correction of the state estimate before the estimation error increases too large in the forward propagation step, leading to a potential high-bandwidth odometry that could even survive in extremely high-speed movements.
Another benefit of the point-wise LIO framework is the fundamental removal of in-frame motion distortion. Existing frame-based framework [11, 15–17, 21, 29–31] accumulates the sequentially-sampled points into a frame and uses to update the state by assuming the frame is sampled at the same time. Since the points in a frame are actually sampled at different times, this accumulation would cause motion distortion for points in a frame. To compensate such distortion, ad-hoc methods, such as IMU integration [17, 18, 21, 29, 38, 44, 53, 57], constant velocity assumption [15, 32–37, 58], or continuous time

trajectory [26, 45, 52], must be used. In contrast, our point-wise LIO framework updates the state at each point's true sampling time, suffering from no such motion distortion.
The second main difference of our LIO framework is how we model the IMU measurements. Unlike existing methods [17, 29, 50, 51], where the IMU measurements are modeled as the input to a kinematic model, we used a colored stochastic process to describe a robot's dynamic behaviors, i.e., angular velocity and acceleration in (2), and model the IMU measurements as the model output (7). Modeling the IMU measurements as the output provides an elegant way to cope with saturated IMU measurements, which allows the estimation of motion beyond the IMU measuring range.
In our system, we adopted an extended Kalman filter, instead of an iterated Kalman filter used in frame-based methods such as FAST-LIO2 [29] and LINS [17]. This is because for the LiDAR measurement, the update rate is very high, so we do not need to iterate the state update exhaustively at each time step; while for IMU measurement, the measurement equation (7) is essentially linear. The elimination of iteration can effectively lower the time on state update. Moreover, since the state is updated by the LiDAR measurements at a point-by-point base, the measurement model of LiDAR points is of one dimension as shown in equation (6). The low-dimension measurement equation along with the great sparsity (e.g., $\mathbf{F}_{\mathbf{x}_k}, \mathbf{F}_{\mathbf{w}_k}$ in (11), $\mathbf{H}_{L_{k+1}}$ in (13) and $\mathbf{H}_{I_{k+1}}$ in (15)) in the system can also effectively lower the computation time.

5 Evaluation

In this section, we evaluate the performance of our developed system in three aspects: 1) removal of motion distortion, 2) high odometry frequency with high bandwidth, and 3) state estimation with saturated IMU measurements in the middle.

5.1 Implementation

The proposed Point-LIO is implemented in *C++* and Robots Operating System (ROS). The extended Kalman filter is implemented based on the *IKFoM* toolbox developed in our previous work [55]. We use the incremental k-d tree, *ikd-Tree*, developed in FAST-LIO2 [29]) as our mapping structure with its default parameters: local map size $L = 2000$ m, spatial downsample resolution $l = 0.25$ m, the re-balancing thresholds of *ikd-Tree* are $\alpha_{\text{bal}} = 0.6$, $\alpha_{\text{del}} = 0.5$, and the sub-tree size threshold for parallel rebuilding (in a second thread) is $N_{\text{max}} = 1500$.
Although our system is designed to perform state estimate after each LiDAR point reception, in practice, limited by the available drivers provided by LiDAR manufacturers, LiDAR points are packaged after accumulating a complete scan and then sent to the LIO systems. To cater for this practical limitation, Point-LIO sorts all LiDAR points and IMU data contained in a received package according to their respective timestamps. Then, the sorted data are processed one by one by Point-LIO.
In all the evaluations, we compare Point-LIO to a state-of-the-art frame-based odometry, FAST-LIO2 [29]. All the experiment results for FAST-LIO2 are collected using the public version of FAST-LIO2 with its default parameter values (which are essentially also the values of our parameters as detailed above). Since FAST-LIO2 performs a spatial downsampling with resolution 0.3 for each received LiDAR scan, for a fair comparison, we also perform such spatial downsampling in Point-LIO before the data sorting and point by point update.
In addition, to study the effect of point-by-point update, we provide a system that only integrates this scheme but not applies the colored stochastic model as an ablation study. The ablation-study system is the same as FAST-LIO2, which models the IMU measurements as the input to the system kinematic model, but differs in that the each individual LiDAR point in a scan is used to update the system sequentially as our proposed system does. This leads to a state update frequency similar to ours. In the

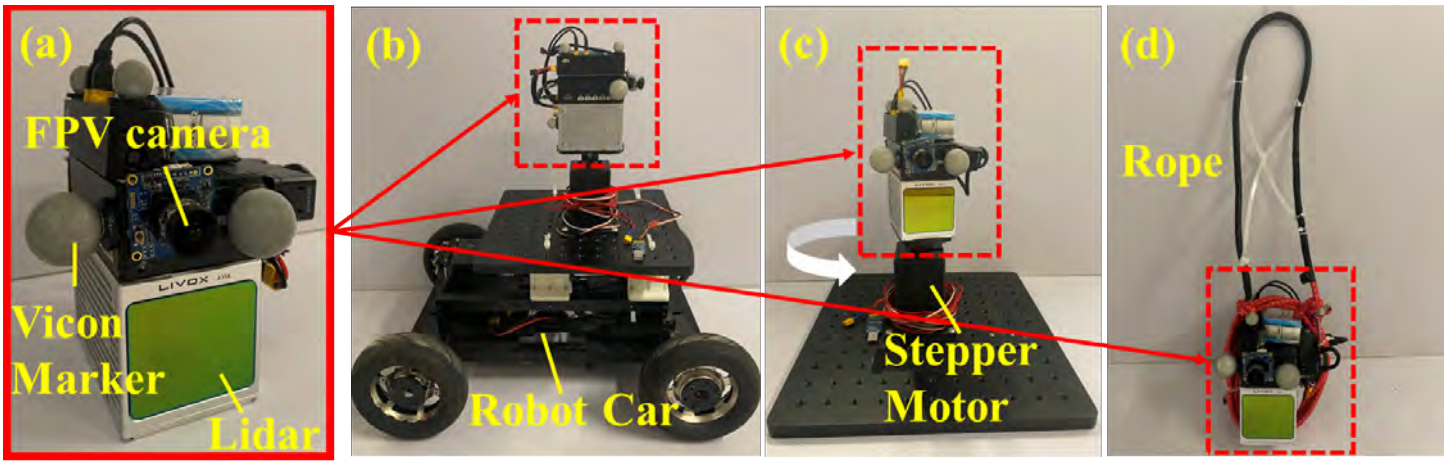


Figure 3: Experimental platforms for evaluations. (a) The sensor suite consisting of a Livox Avia LiDAR, a first-person view (FPV) camera and five Vicon Markers. The sensor suite is carried by (b) a robot car, (c) a rotating platform driven by a step motor, and (d) a pendulum.

following, we denote our developed system Point-LIO as “Point-LIO” and the above ablation-study system as “Point-LIO-input" to distinguish the role of IMU measurements in the two systems.

5.2 Platforms

To collect real-world data, we develop a sensor suite, shown in Figure 3 (a), which consists of a solid-state 3D LiDAR, Livox Avia, a first-person-view (FPV) camera and five Vicon markers for ground-truth measurement. With a 70.4 ° (horizontal) × 77.2 ° (vertical) circular FoV and an unconventional non-repetitive scanning pattern, the Livox Avia LiDAR produces 230,000 Hz point measurements and a built-in IMU (model BMI088) producing 200 Hz IMU data. The points and IMU data are packaged at a frequency adjustable from 10 Hz to 100 Hz. To produce different types of motion, three different platforms are constructed to carry the sensor suite, including a robot car (see Figure 3 (b)), the RoboMaster 2019 AI developed by DJI Shenzhen; a rotating platform driven by a step motor (see Figure 3 (c)), Nimotion STM4260A; and a pendulum (see Figure 3 (d)).

5.3 Resolving motion distortion

To verify the effectiveness of the proposed point-by-point update scheme in addressing motion distortion, we collect some sequences using the robot car shown in Figure 3 (b). Three different scenes are tested, i.e., the Belcher Bay Park (a unstructured scene), the Centennial Small Square of HKU (a semi-structured scene), and a corridor in the Haking Wong building of HKU (a structured scene). The three sequences are denoted as “Park", “Square" and “Corridor", respectively. In all sequences, the robot car returns to the starting point, which enables the drift computation. The LiDAR package frequency is 10 Hz.

One challenge of this experiment is the strong vibration when the robot car moves on the ground. Since the sensor suite is attached to the chassis without any vibration absorber, the vibration will directly pass to the sensor and cause severe jerky motions, as indicated by the built-in IMU data shown in Figure 4. When the robot car is stationary in the beginning, the MU measurements are stably small. As the car starts moving, the measurements of IMU changes rapidly.

5.3.1 Mapping results

The mapping results of Park, Square and Corridor are shown in Figure 5, Figure 6 and Figure 7 respectively. In each figure, we first present an global view of the final mapping results (i.e., sub-figure (a)), then focus on certain local areas (sub-figure (b)) containing large planes (e.g., a wall). We investigate

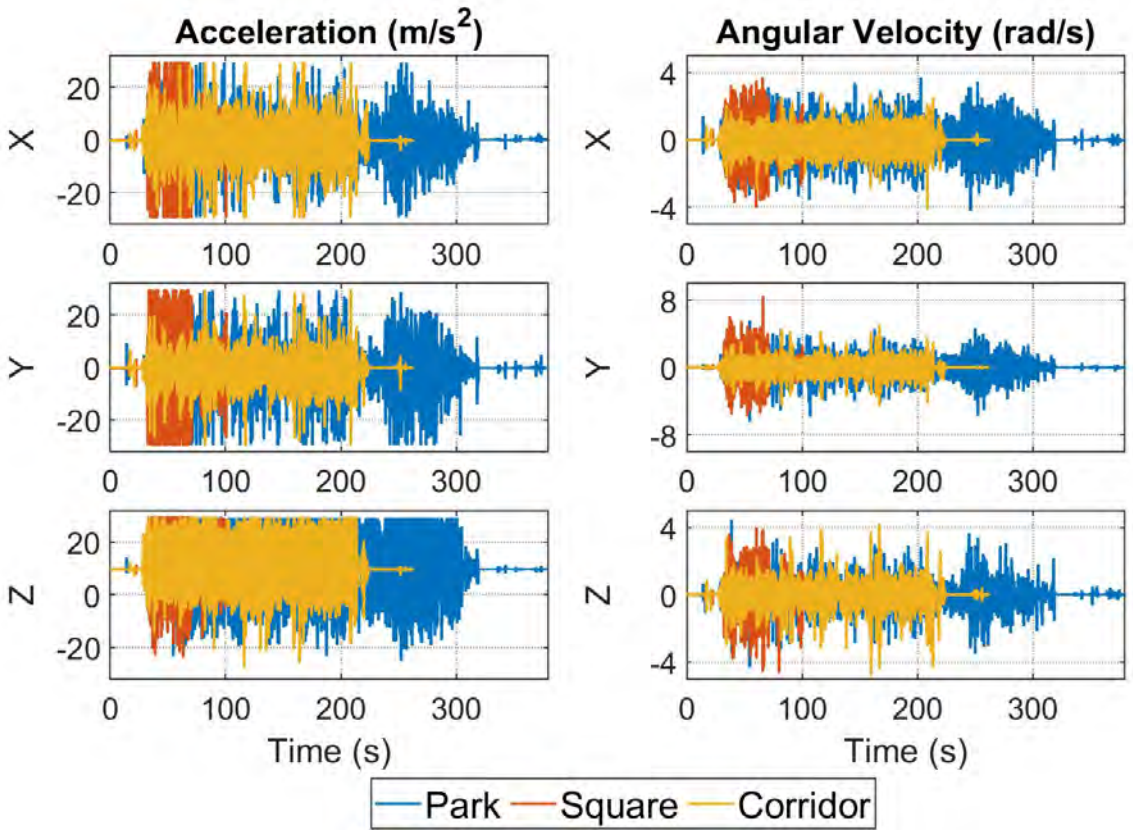


Figure 4: Measurements of IMU for three robot car sequences, i.e., Park, Square and Corridor.

the consistency of points on the wall (sub-figure (c)), from which we can compare the mapping accuracy of FAST-LIO2 (i.e., (b1), (c1)), Point-LIO-input (i.e., (b2), (c2)) and Point-LIO (i.e., (b3), (c3)). Finally, to show the in-frame motion distortion, points of one scan (accumulation over one scan period 0.1 s) in the above-selected local areas are shown (sub-figure (d)) together with the further zoom-in (sub-figure (e)) for FAST-LIO2 (i.e., (d1), (e1)), Point-LIO-input (i.e., (d2), (e2)) and Point-LIO (i.e., (d3), (e3)), where the red points refer to registered LiDAR points in the current scan, and the white points are map results accumulated up to the current scan.

As can be seen from the sub-figures (c) of Figure 5, Figure 6 and Figure 7, the overall map of FAST-LIO2 (c1) is obviously thicker than the Point-LIO-input (c2), and the Point-LIO (c3) produces a further thinner wall than the Point-LIO-input. The reason for this phenomenon lies in the in-frame motion compensation in each individual scan as shown in sub-figure (e) of Figure 5, Figure 6 and Figure 7. As can be seen, all red points around the selected wall are supposed to belong to the same plane, but they actually scatter off the wall for FAST-LIO2 (e1) due to the in-frame motion distortion. This in-frame distortion phenomenon for the Point-LIO-input and Point-LIO is much alleviated ((e2) and (e3)).

FAST-LIO2 uses a backward propagation based on IMU measurements to project all the points of a scan to the scan-end pose. This process is easily disturbed by the IMU measurement noises, biases estimation error, and the limited IMU sampling rate. In particularly, caused by the sensor vibration, the acceleration and angular velocity are changing in a high rate even within one sample interval of the IMU, leading to large IMU propagation errors which assume the angular velocity and acceleration are constant during one sampling interval. Moreover, the low frame rate (i.e., 10 Hz) also requires a long-time (i.e., 100 ms) IMU propagation, which accumulates pose erros and results in large in-frame distortions as shown in sub-figures (b1) and (c1).

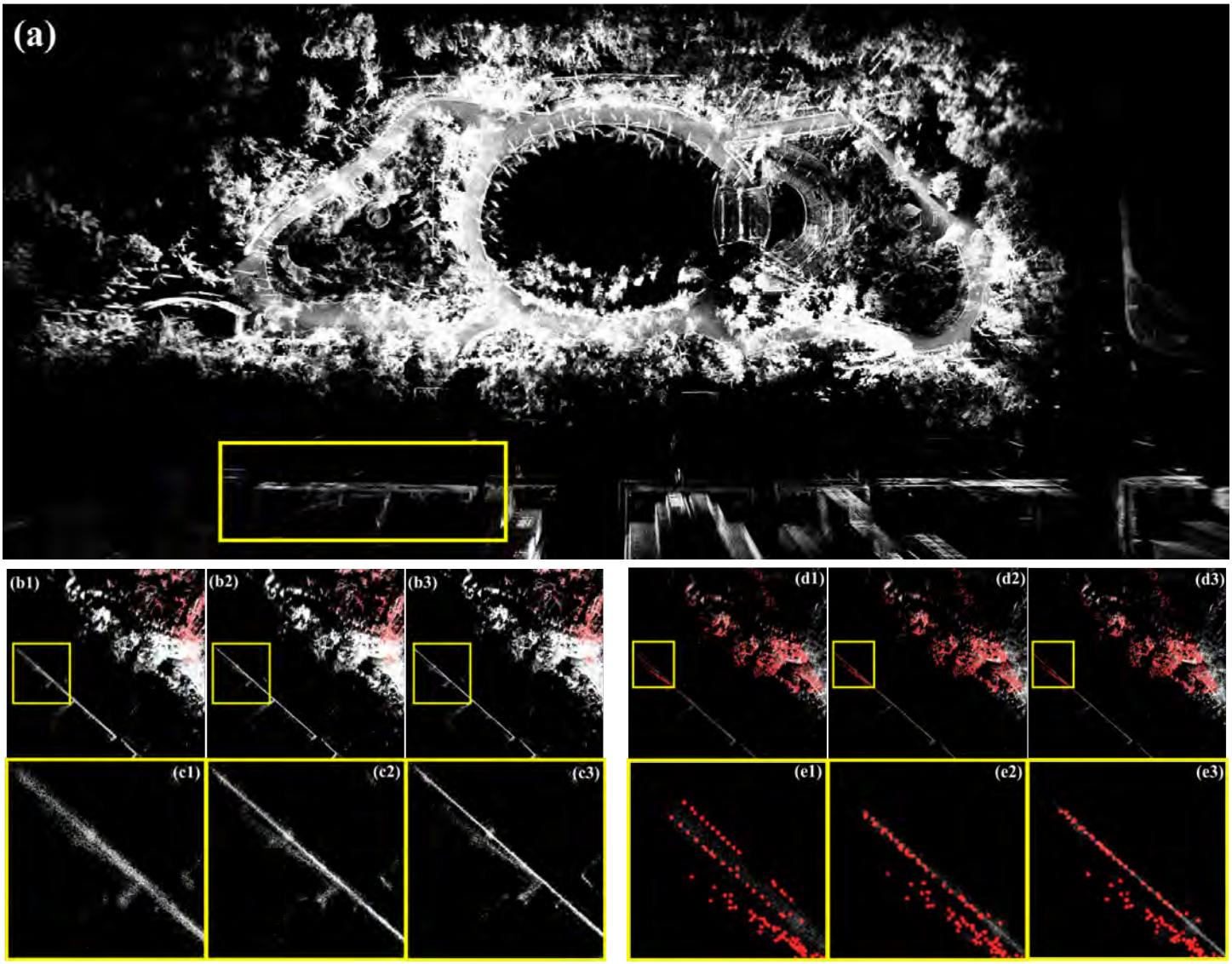


Figure 5: Illustration of motion distortion for the Park sequence.

In contrast, Point-LIO fuses the LiDAR point at its true sampling time without any points accumulation, which fundamentally eliminates the motion distortion as shown in sub-figures (c2) for the Point-LIO-input and sub-figures (c3) for the Point-LIO. Furthermore, when comparing the Point-LIO-input (c2) with the Point-LIO (c3) both with the point-by-point update scheme, the Point-LIO performs slightly better. This is because the Point-LIO-input still uses the IMU measurements to propagate the state (although for only one LiDAR point interval), hence still suffering from the IMU measurements noise and bias estimation errors. In contrast, the Point-LIO uses the filtered but not the raw data of IMU measurements to propagate the state, which slightly reduces the motion distortion as observed in (e3). The video of an example sequence, the Park, is available online https://youtu.be/oS83xUs42Uw.

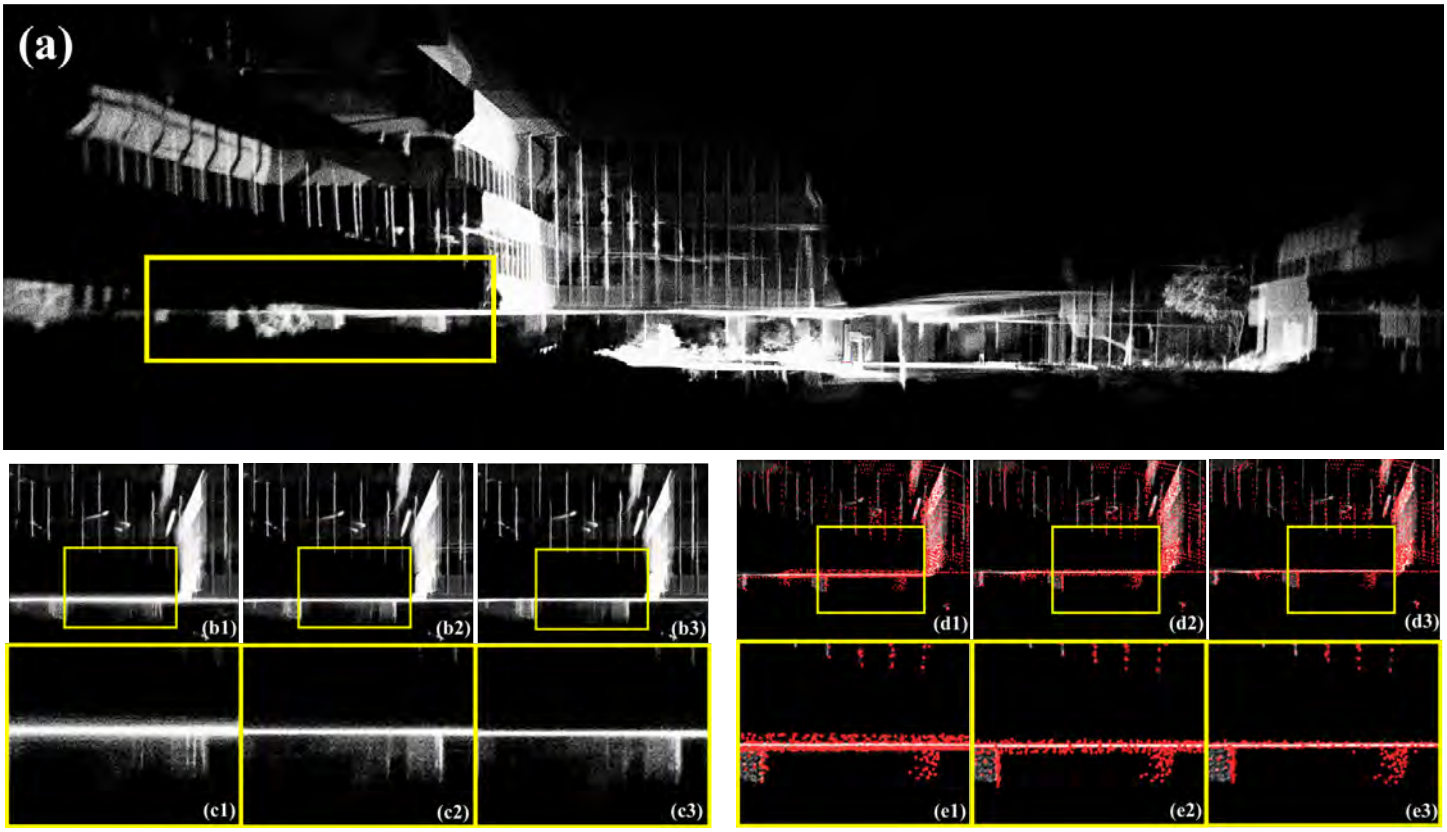


Figure 6: Illustration of motion distortion for the Square sequence.

5.3.2 Drift results

The drifts of FAST-LIO2, Point-LIO-input and Point-LIO are summarized in Table 1. Due to the imperfect operation of the robot car, the distance between the starting and ending position is not exactly zero, but less than 10 cm. As can be seen, all the FAST-LIO2, Point-LIO-input and Point-LIO have comparable drifts for Square and Corridor sequences, while for the Park sequence, FAST-LIO2 fails to return the starting point. This is because that the Park is an unstructured environment, which makes the effect of motion distortion on odometry accuracy more evident.

Table 1: Comparison of odometry drifts (meters)

	FAST-LIO2	Point-LIO-input	Point-LIO
Park:	1.242	0.064	0.080
Square:	0.041	0.038	0.039
Corridor:	0.047	0.043	0.047

5.4 High odometry output frequency and high bandwidth

We test the frequency of odometry output of FAST-LIO2, Point-LIO-input and Point-LIO on an indoor dataset (denoted as "Odo") collected using the rotating platform (see Figure 3 (c)) by giving quickly-varying speed commands to the step motor, the package rate of LiDAR is 100 Hz in this experiment. Even though the dataset is collected with LiDAR rate of 100 Hz, the framework of FAST-LIO2 is naturally extendable to higher state update frequency by splitting one frame into multiples (but below the IMU rate 200 Hz). Thus we divide one LiDAR frame into two for FAST-LIO2. Figure 8 shows the distribution of number of odometry output per second. The output frequency for FAST-LIO2 is 200 Hz, which is the frame rate. As comparison, output frequencies for the Point-LIO-input and the Point-LIO

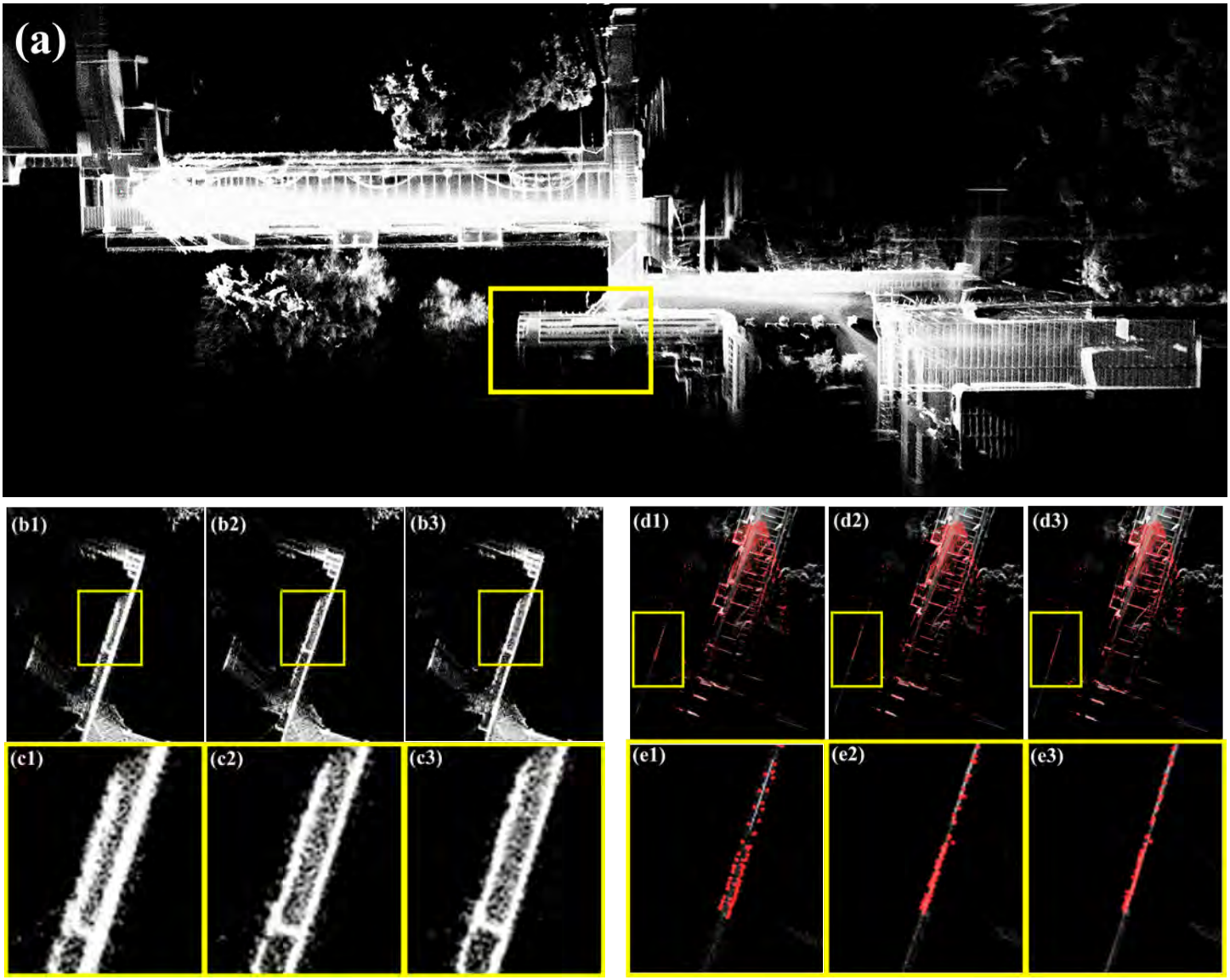


Figure 7: Illustration of motion distortion for the Corridor sequence.

are in the range of 4 kHz and 8 kHz, which is the number of points that pass the plane correspondence check.

To enable the bandwidth analysis, the above experiment is re-conducted with ground-truth measurements from Vicon Tracker recorded at the highest frequency 300 Hz. Dividing system output (the yaw angle estimated from the odometry) by the system input (the ground-true yaw angle measured by the Vicon system), we obtain the magnitude response (dB) of FAST-LIO2, Point-LIO-input and Point-LIO, at different input frequencies as shown in Figure 9. As can be seen, the magnitude response of FAST-LIO2 starts dropping when the input frequency is near to 100 Hz, suggesting a 100 Hz bandwidth (see Table 2). 100 Hz is also the highest attainable bandwidth when the odometry output frequency is 200 Hz according to the Nyquist–Shannon sampling theorem. In contrast, both the Point-LIO-input and the Point-LIO have bandwidth larger than 150 Hz which is beyond the measuring capability of Vicon system.

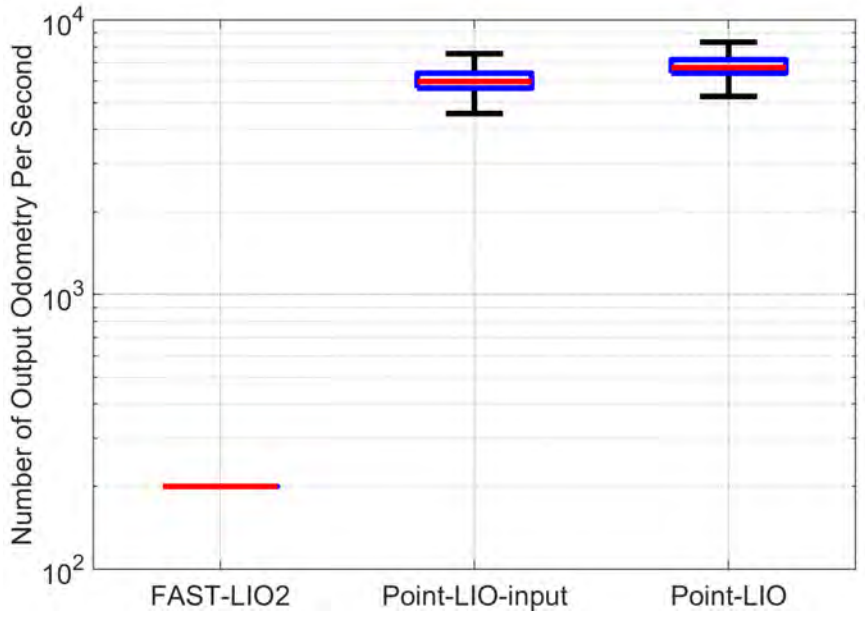


Figure 8: Number of output odometry per second of FAST-LIO2 and Point-LIO-input and Point-LIO.

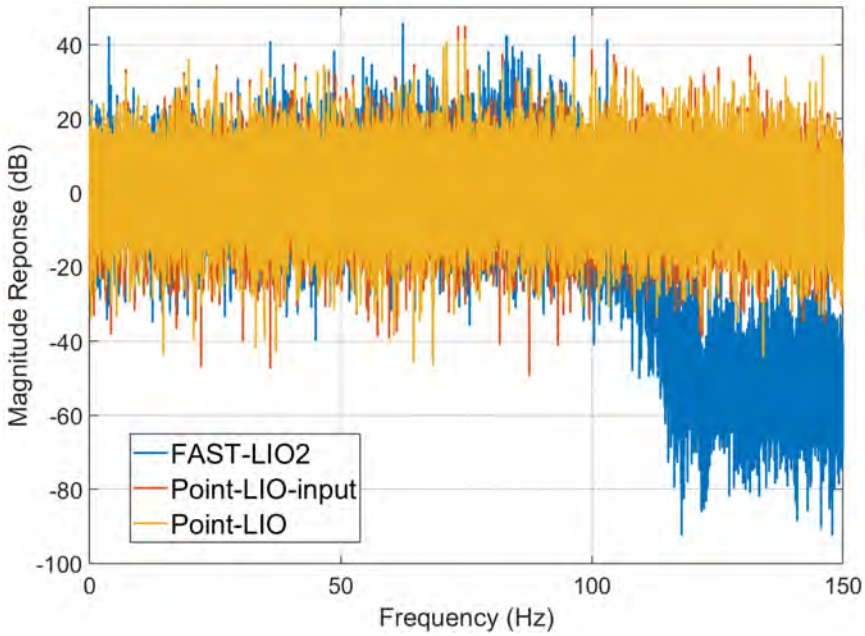


Figure 9: Bandwidth analysis of FAST-LIO2, Point-LIO-input and Point-LIO.

Table 2: Comparison of average frequency of odometry output (Hz) and bandwidth (Hz)

	FAST-LIO2	Point-LIO-input	Point-LIO
Output Odo.:	200	6132	6955
Bandwidth:	100	> 150	> 150

5.5 Extremely aggressive motion with saturated IMU measurements

While the Point-LIO-input and Point-LIO have comparable performances so far, in this section, we show that the Point-LIO is able to track extremely aggressive motions even beyond the IMU measuring ranges.

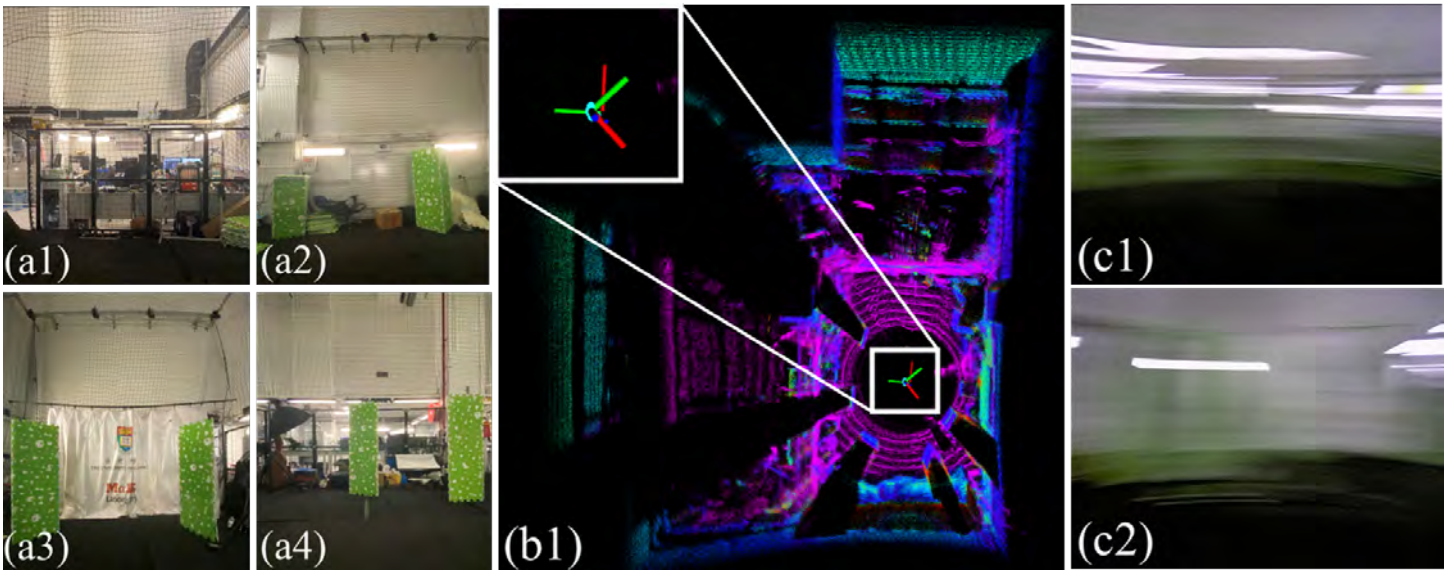
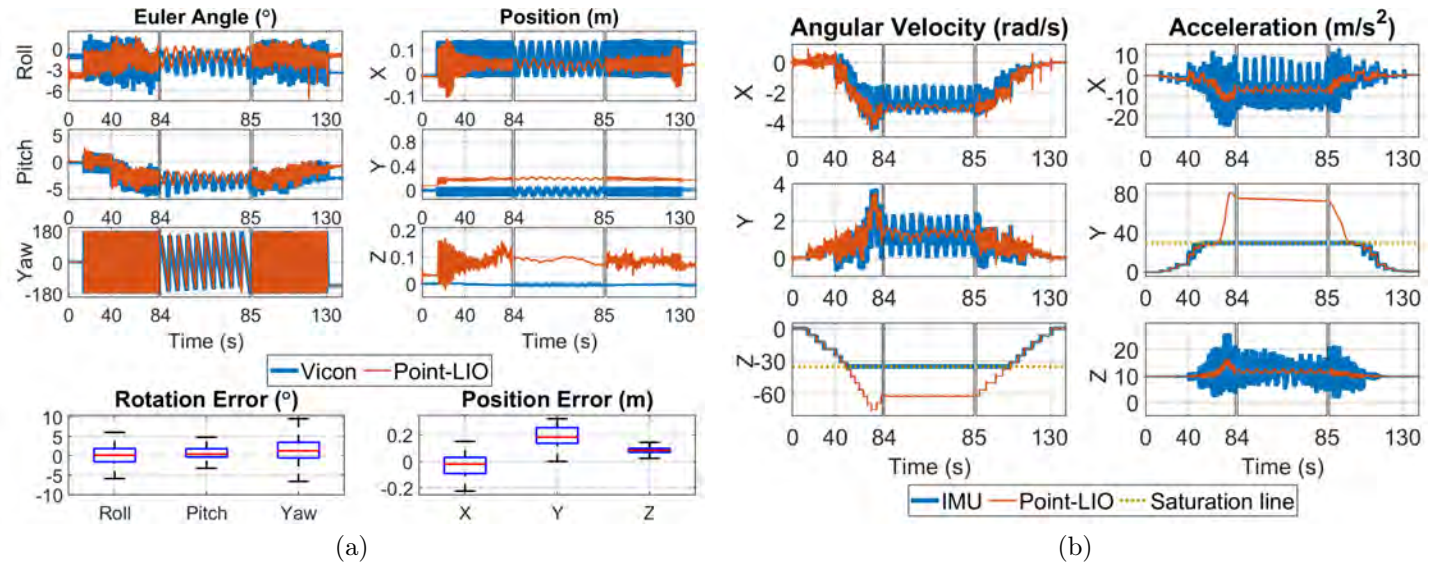


Figure 10: Test environments of the spinning motion experiment and the mapping results of Point-LIO.

Two types of motion are produced in the experiments, one is spinning motion (denoted as "Satu-1") and the other is a circling in the space (denoted as "Satu-2"). Both experiments suffer from IMU saturation after initial stage caused by either the high spinning rate or the large centrifugal forces. To the best of our knowledge, no prior SLAM systems could cope with such aggressive motions or the saturated IMU measurements.

5.5.1 Spinning motion

This experiment is conducted using the rotating platform placed in a cluttered laboratory environment (see Figure 10 (a1)-(a4)). During the experiment, the sensor suite is rotated by the step motor under step angular speed commands, which increases from zero to a peak value step by step and then decreases to zero at the end. The resultant peak angular velocity is 75 rad s^{-1} (in yaw), which far exceeds the IMU measuring range, i.e., 35 rad s^{-1}. The high angular velocity also causes a peak acceleration around 80 m s^{-2}, also far beyond the IMU measuring range, i.e., around 30 m s^{-2}. The onboard FPV images shown in Figure 10 (c1) and (c2) give an illustration of the rotation in process.
The mapping result of Point-LIO is shown in Figure 10 (b1), which shows a fairly consistent mapping of the environment, and the estimated ending position coincides with the starting position very well as shown in Figure 10 (b2). The estimated kinematic state, including rotation in Euler angles and position, are compared with the ground-truth in Figure 11 (a), where the x axis is broken into three segments to zoom in the time period 84-85 s. The continuous rapid change of yaw is due to the continuous rotation driven by the step motor, and the sinusoidal-like fluctuations in positions are caused by the offset between the Vicon marker and the step motor shaft. As can be seen, the estimated yaw angle can closely track the actual ones in the whole process, the overall rotation and translation error (in terms of RMSE) are 4.60 ° and 0.233 m, respectively. The slightly large RMSE of translation is mostly caused by the y-direction, where the constraints in this direction are insufficient from the beginning. Considering the extreme motions in the experiment, this translation error is well acceptable.
Another benefit of our Point-LIO is the capability of estimating angular velocity and acceleration (which are states of our system and can hence be estimated by the Kalman filter) where the IMU are saturated. The estimation versus the IMU measurements are plotted in Figure 11 (b). As can be seen, during the time period 50-106 s, the IMU saturates (z-axis for gyroscope and y-axis for accelerometer), while our Point-LIO can still give a reasonable estimate. Outside this region, the estimation from our Point-LIO is in good agreement with the IMU measurements albeit some high-frequency components are filtered.
We further challenge Point-LIO by starting it at different initial angular speed of the motor. As shown by the map results in Figure 12 and the RMSE of rotation in Table 3, when the initial angular veloc-



(a) (b)

Figure 11: (a) Estimation results of Point-LIO for rotation in Euler angles and position of spinning motion experiment. (b) Comparison of angular velocity and acceleration between estimation of Point-LIO and measurements from IMU. The gray dotted lines indicate the saturation values of the IMU.

ity is below the IMU saturation value, i.e., 35 rad s^{-1}, the Point-LIO is able to survive by constructing a reasonable map and state estimation. The quality of state estimation is slightly degraded when compared to the above case where the sensor starts from a stationary pose. This performance degradation is reasonable since the fast initial angular speed causes Point-LIO to build a biased map at the very beginning, which further misleads the subsequent state estimate. This also causes the RMSE of rotation to increase with the initial angular velocity as seen in Table 3. When the initial angular velocity is beyond the IMU measuring range, Point-LIO fails due to the drastic large initial state estimate (e.g., the initial angular velocity estimate is set to zero while the actual is above 35 rad s^{-1}).

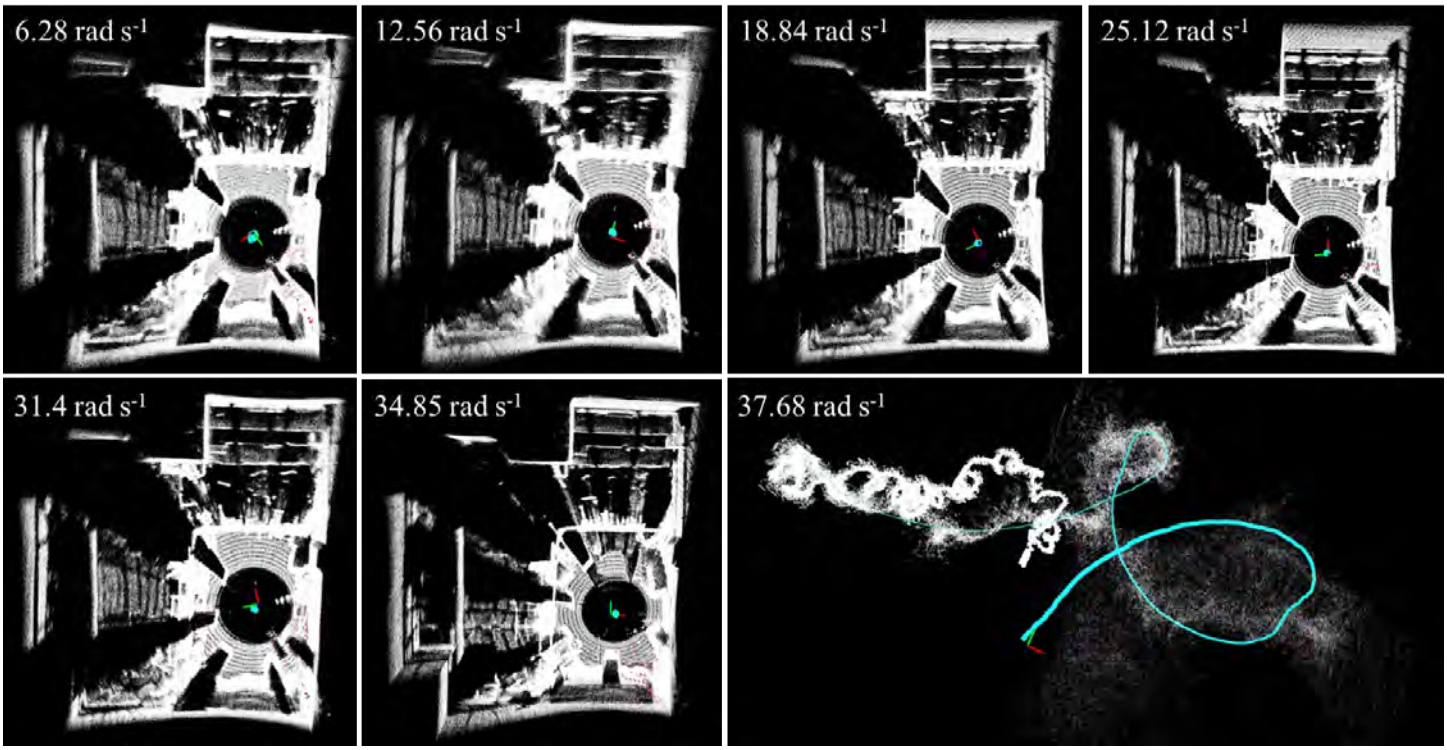


Figure 12: Map results of Point-LIO starting with different angular velocity.

Table 3: Rotation RMSE (°) for Point-LIO starting with different angular velocity (rad s^{-1}).

Angular velocity at start (rad s^{-1}):	6.28	12.56	18.84	25.12	31.40	34.85	37.68
RMSE of rotation (°):	6.9	9.7	13.6	14.6	16.4	18.7	fail

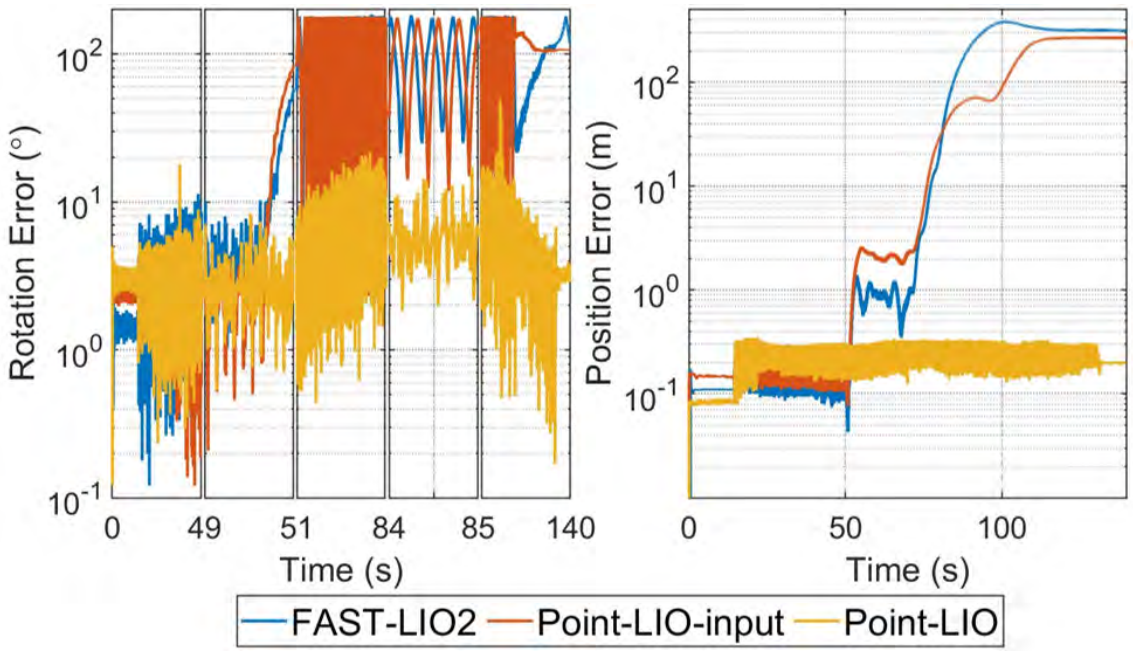


Figure 13: Error comparison of FAST-LIO2, Point-LIO-input and Point-LIO, for rotation and position estimations.

Finally, as acomparison, we run the FAST-LIO2 and Point-LIO-input on the same dataset, and the error comparisons of rotation and position are presented in Figure 13. As can be seen, the estimations of FAST-LIO2, Point-LIO-input and Point-LIO have comparable rotation error during the first 50 s, where IMU works normally. From time 50 s where IMU starts saturating, the estimations of FAST-LIO2 and Point-LIO-input start to diverge immediately for rotation, and the estimation of position also starts to drift and then diverge. In conclusion, the FAST-LIO2 as well as the Point-LIO-input fails to work under saturated IMU measurements while our proposed Point-LIO can survive well if the saturation does not last from the beginning.

5.5.2 Circling motion

This experiment is conducted using the pendulum in the same laboratory environment (see Figure 14 (d)). In this experiment, the sensor suite is tied to one end of a rope that is swing into a circling trajectory in the vertical plane (see Figure 14 (b)). This motion causes an acceleration at the bottom of the circle up to 40 m s^{-2}, which exceeds the IMU measuring range 30 m s^{-2}. FPV images shown in Figure 14 (e) gives an visual illustration of the motion. More details are shown in a video available online https://youtu.be/oS83xUs42Uw.

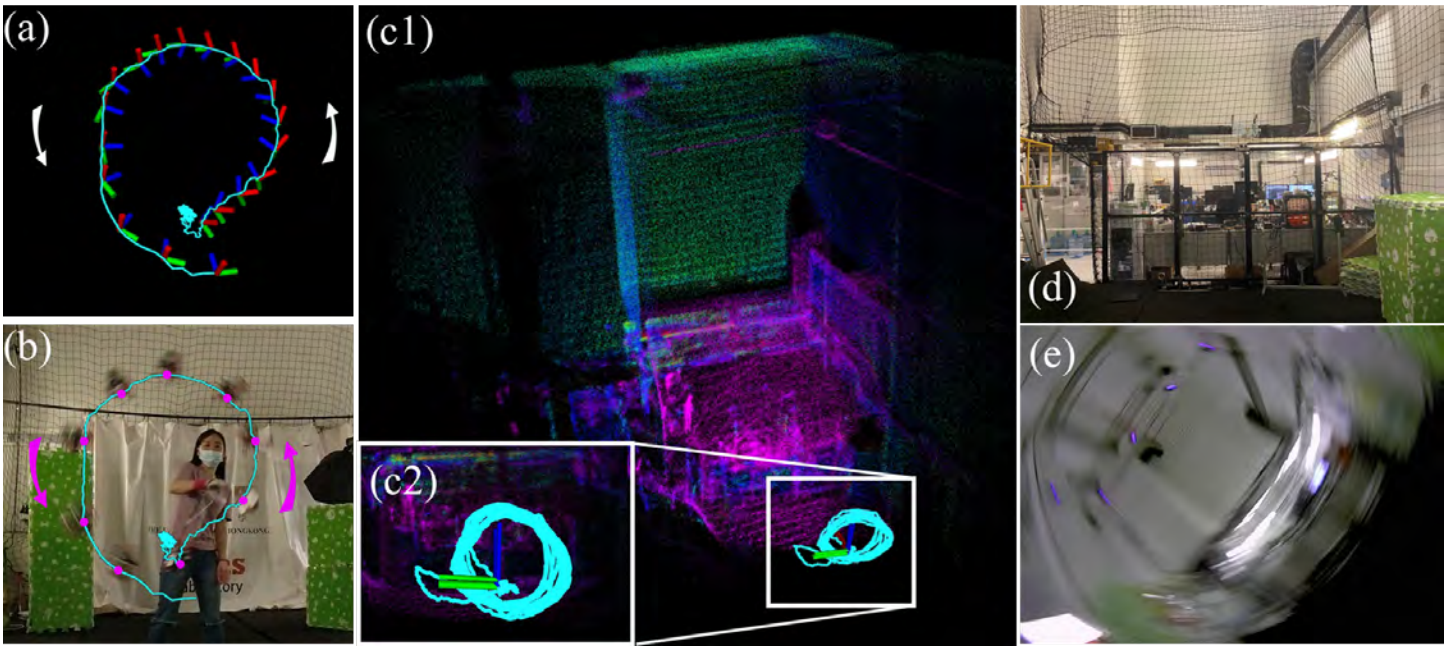
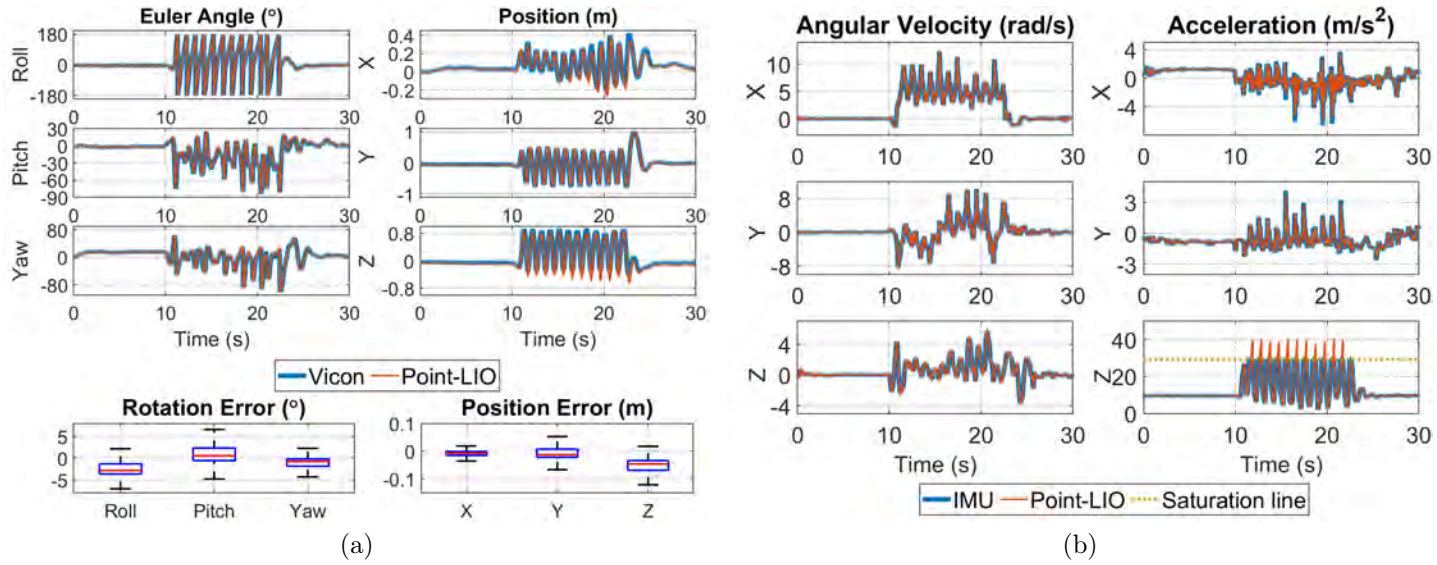


Figure 14: Test environment of the circling motion experiment and the mapping results of Point-LIO.

Qualitatively, Figure 14 (c1) and (c2) show the mapping results of our proposed Point-LIO. Since the LiDAR is facing front with a 70.4 ° × 77.2 ° circular FoV, only one side of the laboratory is mapped. The estimated trajectory is shown in Figure 14 (a), which is in high agreement with the actual sensor path shown in Figure 14 (b).
Quantitatively, the estimated rotation in Euler angles and position are compared with ground-truth measured by Vicon in Figure 15 (a), where both the Euler angle estimation and the position estimation successfully demonstrate the 12 sequential circles. The RMSE of the average rotation and translation errors are 4.42 ° and 0.0990 m, respectively. Figure 15 (b) shows the estimated angular velocity and acceleration by the Point-LIO versus the IMU measurements. As can be seen, the estimations agree with the IMU measurements pretty well when those dynamic states are within the IMU measurements. Moreover, our Point-LIO can give reasonable acceleration estimate even when the IMU saturates. Figure 16 further shows the error comparison among the Point-LIO, the Point-LIO-input, and the frame-based odometry FAST-LIO2. Similar to the last experiment, our Point-LIO achieves consistently lower estimation errors than the other two due to the ability to cope with IMU saturation.



(a) (b)

Figure 15: (a) Estimation results of Point-LIO for rotation in Euler angles and position of the circling motion experiment. (b) Comparison of angular velocity and acceleration between estimation of Point-LIO and measurements from IMU. The gray dotted lines refer to saturation values of the IMU.

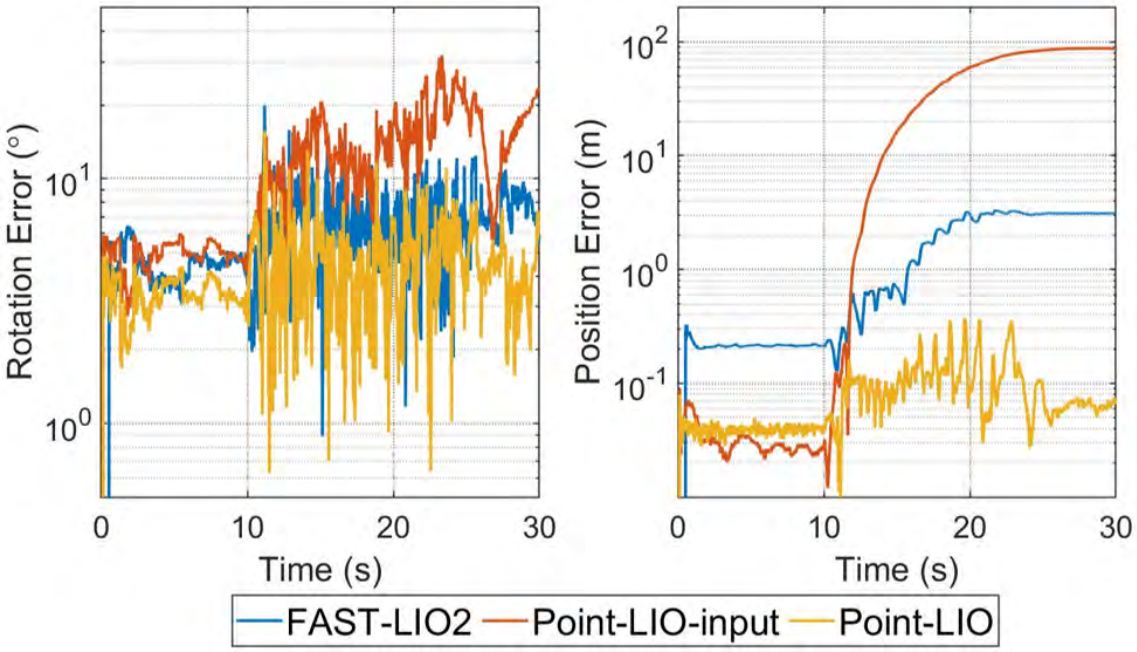


Figure 16: Error comparison of FAST-LIO2, Point-LIO-input and Point-LIO.

5.6 Real-time performance

The average time costs of each step of the Point-LIO-output for processing one scan of LiDAR points are shown in Figure 17, which are tested on an intel i7-based micro-UAV onboard computer, a DJI Manifold 2-C7 with a 1.8 GHz quad-core Intel i7-8550U CPU and 8 GB RAM. The mapping includes searching of nearest points and adding of point to the map, which takes up the largest part of time consumption. Even the system states are updated at each LiDAR point, the time for EKF filtering including state propagation and update are less than 10 ms for 10 Hz sequences and less than 1 ms for 100 Hz sequences. The average total time consumption for one scan is compared among FAST-LIO2, Point-LIO-input and Point-LIO-output in Table 4. × indicates that the LIO fails in those sequences. As can be seen, our Point-LIO, with either the Point-LIO-input (i.e., Point-LIO-input) or the output model (i.e., Point-LIO), have

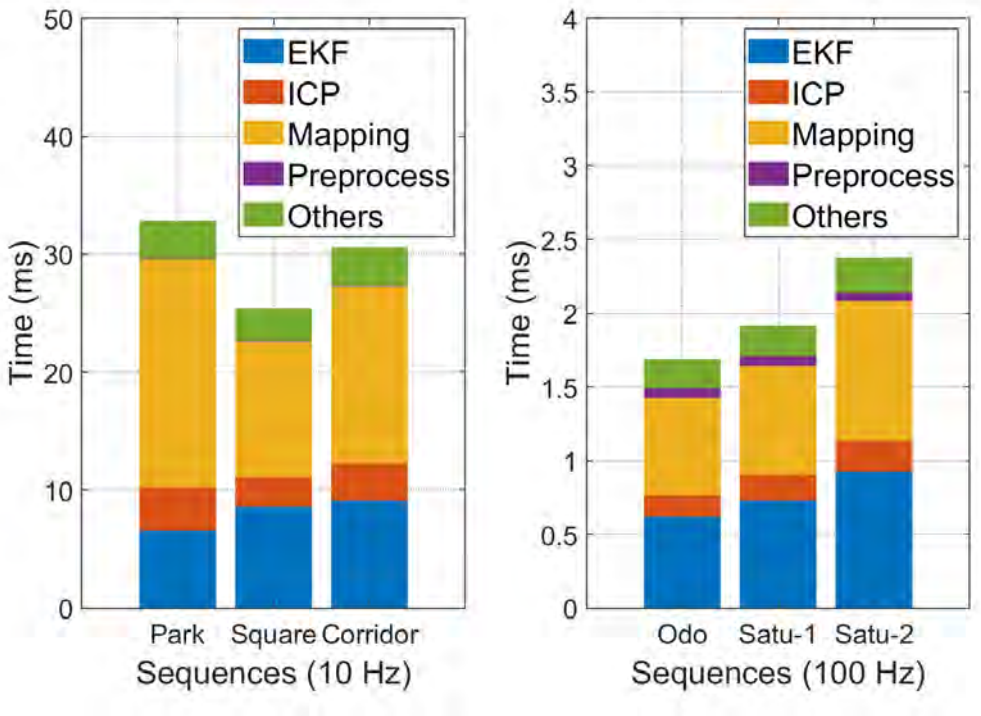


Figure 17: Time usage in each step of Point-LIO.

time consumption comparable to FAST-LIO2, and they all achieve real-time performance, i.e., within 100 ms for 10 Hz sequences and within 10 ms for 100 Hz sequences. Finally, the average number of points processed per second (including those with and without plane correspondences) over all sequences are 33,710, and the average processing time per point is 9 us.

Table 4: Comparison of time consumption (milliseconds) per scan for Point-LIO and FAST-LIO2

		FAST-LIO2	Point-LIO-input	Point-LIO
$10Hz$	Park:	39.84	32.09	32.94
	Square:	21.89	25.20	25.41
	Corridor:	28.36	28.76	30.59
$100Hz$	Odo:	1.18	1.70	1.69
	Satu-1:	×	×	1.92
	Satu-2:	×	×	2.38

6 Benchmark Results

In this section, we test Point-LIO on various public sequences, which have more gentle motion without IMU saturation, and compare it with other state-of-the-art LiDAR-inertial odometry methods, including FAST-LIO2 [29], LILI-OM [11], LIO-SAM [21], and LINS [17]. The computation platform for benchmark comparison is the same lightweight UAV onboard computer as used in FAST-LIO2 [29], which is a DJI Manifold 2-C7 with a 1.8 GHz quad-core Intel i7-8550U CPU and 8 GB RAM, hence the results for FAST-LIO2, LILI-OM, LIO-SAM, and LINS can be directly obtained from the original paper [29]. Since

Point-LIO uses the same mapping structure as FAST-LIO2, for a fair comparison, we set the mapping parameters of Point-LIO to the default values of FAST-LIO2 used in [29], i.e., the local map size $L = 1000$ m, the LiDAR raw points are directly fed into state estimation after a $1:4$ (one out of four LiDAR points) temporal downsampling, and spatial downsample resolution $l = 0.5$ m with re-balancing thresholds of *ikd-Tree* as $\alpha_{\text{bal}} = 0.6$, $\alpha_{\text{del}} = 0.5$ and $N_{\max} = 1500$. For the EKF part of Point-LIO, the LiDAR measurement noise of Kalman filter is set as $\bar{\mathcal{R}}_k = 10^{-2} \cdot \mathbf{I}$ (see (18)). These parameter values are kept the same for all sequences.
We evaluate our method on the same 12 sequences used in FAST-LIO2 [29], which are drawn from 4 different public datasets, i.e., "lili" from LILI-OM [11], "utbm" [59], "ulhk" [60] and "liosam" from the work LIO-SAM [21]. Among them, "lili" uses a solid-state 3D LiDAR, Livox Horizon while the other three datasets use the spinning LiDARs, i.e., Velodyne HDL-32E LiDAR for "utbm" and "ulhk", and VLP-16 LiDAR for "liosam". These LiDARs have different scanning patterns. We refer the readers to [29] for more detailed information about the dataset and the selected sequences.

6.1 Accuracy Evaluation

Similar to [29], two criteria, the root mean square deviation (RMSE) of average translation error (for sequences with good ground-true trajectory) and the end-to-end error (for sequences starting and ending at the same location), are used for the accuracy evaluation.

6.1.1 RMSE benchmark

The RMSEs are computed using the same method of the publication [29] and reported in Table 5. Compared with other LIO methods, our Point-LIO achieves the best performances in 4 out of 5 sequences, especially an obvious improvement on utbm_9, while has a slightly higher RMSE in liosam_1. The overall accuracy of our method is on par to (and most cases better than) the counterparts.

Table 5: Comparison of RMSE (meters) for Point-LIO, FAST-LIO2 and other state-of-art LiDAR-inertial systems on public sequences.

	utbm_8	utbm_9	utbm_10	ulhk_4	liosam_1
Point-LIO	**23.77**	**26.35**	**15.70**	**2.17**	5.24
FAST-LIO2	27.29	51.60	16.80	2.57	**4.58**
LILI-OM	59.48	782.11	17.59	2.29	18.78
LIO-SAM	×	×	×	3.52	4.75
LINS	48.17	54.35	60.48	3.11	880.92

6.1.2 Drift Benchmark

The end-to-end errors are reported in Table 6. The overall trend is similar to the RMSE benchmark results, that our Point-LIO achieves the lowest drifts in 5 out of 7 sequences. The result for sequence lili_8 is worse than LILI-OM and FAST-LIO2, which is because that the LILI-OM has tuned parameters for each of their own sequences "lili", while parameters of Point-LIO are kept the same among all the sequences. And since lili_8 has a much longer trajectory than the other two "lili" sequences, the drift caused by inappropriate parameters would accumulate along the following process and result in more than 10 meters drift worse than FAST-LIO2. Also, Point-LIO shows slightly larger RMSE on sequence ulhk_6 relative to FAST-LIO2 albeit the margin is very small.
As can be seen from the above benchmark results, Point-LIO achieves higher accuracy in most sequences while for the rests, the margin from the best method is not significant. Considering the various types of LiDARs, environments, and moving platforms across all datasets and sequences, this effectively shows the accuracy and robustness of our method on real-world data.

Table 6: Comparison of drifts (meters) for Point-LIO, FAST-LIO2 and other state-of-art LiDAR-inertial systems on public sequences.

	lili_6	lili_7	lili_8	ulhk_5	ulhk_6	liosam_2	liosam_3
Point-LIO	< **0.1**	< **0.1**	28.64	< **0.1**	2.30	< **0.1**	**7.85**
FAST-LIO2	< **0.1**	1.63	17.39	0.39	< **0.1**	< **0.1**	9.50
LILI-OM	0.8	4.13	**15.6**	1.84	7.89	1.95	13.79
LIO-SAM	×	×	×	0.83	2.88	×	8.61
LINS	×	×	×	0.9	6.92	×	29.9

6.2 Processing time evaluation

Table 7 shows the processing time of Point-LIO, FAST-LIO2, LILI-OM, LIO-SAM, and LINS in all the sequences. Both Point-LIO and the FAST-LIO2 integrate odometry and mapping together, where the map is updated immediately at each step the odometry is updated. Therefore, the total time ("Total" presented in Table 7) counts all possible procedures occurring in the odometry, including point-to-map-matching, state estimation, and mapping. On the other hand, LILI-OM, LIO-SAM, and LINS are all based on a separate architecture of odometry (including feature extraction and rough pose estimation) and mapping (such as back-end fusion in LILI-OM [11], incremental smoothing and mapping in LIO-SAM [21], and map-refining in LINS [17]), whose average processing time per LiDAR scan are summed up by those two parts ("Odo." and "Map." respectively presented in Table 7) when ranking the computation time.

Table 7: Comparison of time consumption (milliseconds) per scan for Point-LIO, FAST-LIO2 and other state-of-art LiDAR-inertial systems on public sequences.

	Point-LIO	FAST-LIO2	LILI-OM		LIO-SAM		LINS	
	Total	Total	Odo.	Map.	Odo.	Map.	Odo.	Map.
utbm_8	**19.68**	22.05	65.28	84.76	×	×	37.44	153.92
utbm_9	**19.70**	25.44	68.94	97.90	×	×	38.82	154.06
utbm_10	**19.65**	22.48	66.10	97.29	×	×	33.61	166.12
lili_6	14.19	**12.56**	68.95	58.46	×	×	×	×
lili_7	**11.77**	17.61	40.01	83.71	×	×	×	×
lili_6	**10.57**	15.31	61.80	79.11	×	×	×	×
ulhk_4	20.54	**20.14**	52.40	74.80	39.50	95.29	34.72	93.70
ulhk_5	35.71	**23.90**	53.56	47.68	25.68	127.63	28.01	99.13
ulhk_6	43.01	**31.56**	64.46	70.43	15.16	164.36	41.54	199.96
liosam_1	**10.85**	14.77	48.45	84.28	13.47	135.39	24.13	179.44
liosam_2	24.77	**19.77**	42.58	99.01	13.09	154.69	20.71	160.66
liosam_3	**12.79**	16.64	38.42	64.02	11.32	124.35	40.47	117.25
Average	20.27	**20.19**	55.91	78.45	19.70	133.62	33.27	147.14

As can be seen, our proposed method and FAST-LIO2 achieve the least computation time when compared with other methods. When compared to FAST-LIO2, our method takes less time on 7 out of 12 sequences. The average computation time of these two methods are very close. Note that, due to the frame-based architecture, FAST-LIO2 uses 4 threads to parallelize the nearest point search, while our method, due to the point-wise architecture, has to perform such operations sequentially. Still, our method takes comparable average computation time, suggesting a fewer use of the computation resources, which could be reserved for other modules (e.g., planning, control). This computation efficiency is attributed

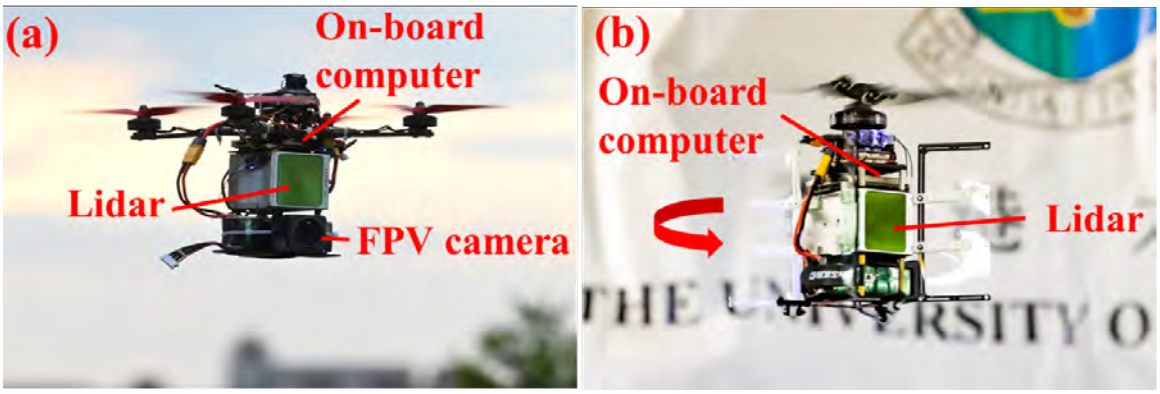


Figure 18: Platforms for application. (a) Racing drone; (b) self-rotating UAV.

to the great sparsity of the system and the elimination of iteration in the Kalman filter as explained in section 4.4.
In summary, Point-LIO has accuracy and computation efficiency comparable to FAST-LIO2, while costing fewer computation resources. At the meantime, the Point-LIO is significantly faster than the current state-of-the-art LIO algorithms, i.e., LILI-OM, LIO-SAM, LINS, while achieving highly competitive or better accuracy.

7 Applications

In this section, we apply our proposed Point-LIO to the state estimation of two unmanned aerial vehicles (UAVs): one is on a racing quadrotor drone shown in Figure 18 (a) which has a thrust to weight ratio up to 5.4. The high thrust weight ratio enables the drone to perform extremely agile motions. The other UAV is on an agile, single-actuated aircraft called self-rotating UAV as shown in Figure 18 (b). The propeller blades are attached to the motor shaft through two passive hinges [61, 62], which, by modulating momentary acceleration and deceleration on the propeller rotation speed, is able to produce the roll and pitch moment necessary to stabilize the UAV's attitude. Due to the uncompensated moment produced by the motor, the UAV will produce a high-rate continuous yaw rotation.

7.1 Racing drone

As shown in Figure 18 (a), the racing drone is mounted with a Livox Avia LiDAR and a FPV camera. The LiDAR FoV is aligned to that of the FPV camera, based on which an expert human pilot manually controls the drone to perform extremely agile flight maneuvers. The flights are conducted above a farmland with vegetation, pond and buildings (see Figure 19 (b), (c)). Several aggressive maneuvers are performed during the flight, including extremely fast rolling flips (see Figure 19 (e1)-(e3)), cliff-diving and lateral accelerations. During the flip, the angular velocities reach 59.37 rad s^{-1}, which exceeds the IMU measuring range 35 rad s^{-1}. Readers can refer to the accompanying video https://youtu.be/oS83xUs42Uw for better visual illustration of the experiment.
We conducted two flights and our method succeeded on both of them. Due to the space limit, we present the results of one flight only. The mapping results are shown in Figure 19 (d1)-(d3), it is seen that the constructed map is consistent with easily distinguishable fine structures on the ground, such as trees and buildings. The estimation of rotation in Euler angles, position and velocity are shown in Figure 20 (a), and the estimation of the angular velocities versus the IMU measurements are shown in Figure 20 (b). From these results, we can see that our method is able to estimate the drone's states even with extremely aggressive motions. As noted, the maximum velocity on one axis reaches 14.63 m s^{-1}, maximum acceleration reaches 30 m s^{-2}, and angular velocity reaches 59 rad s^{-1}. Besides the extremely agile mo-

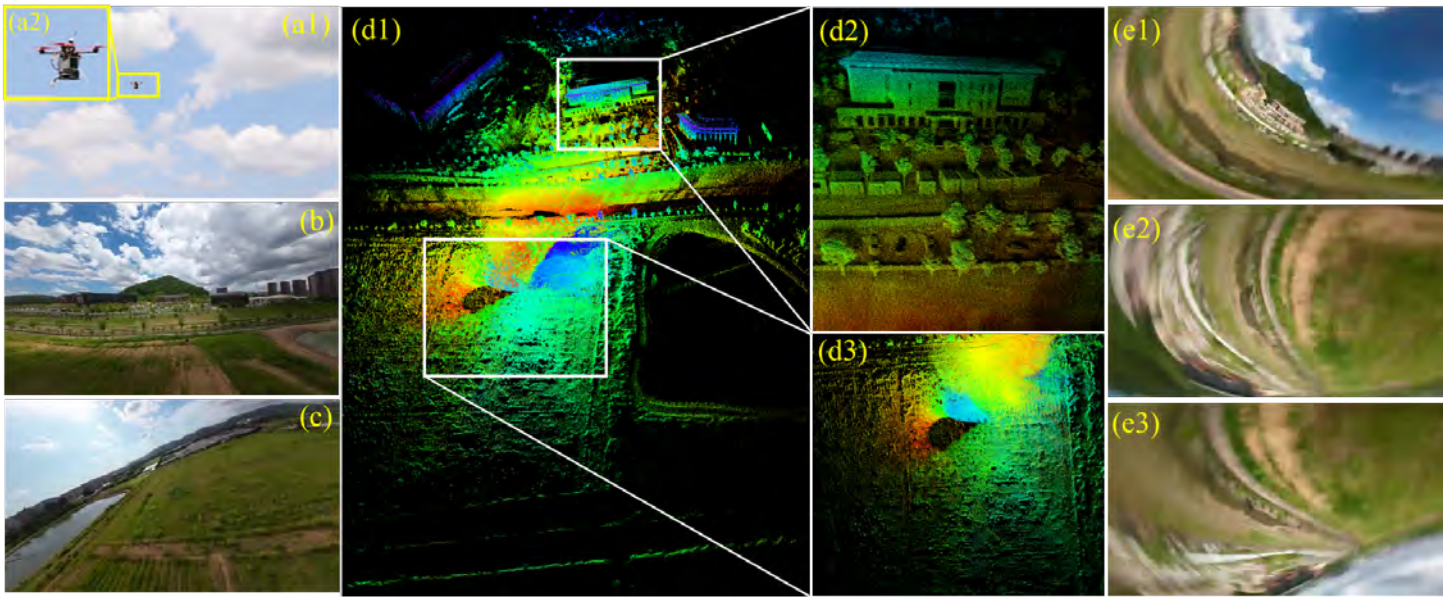
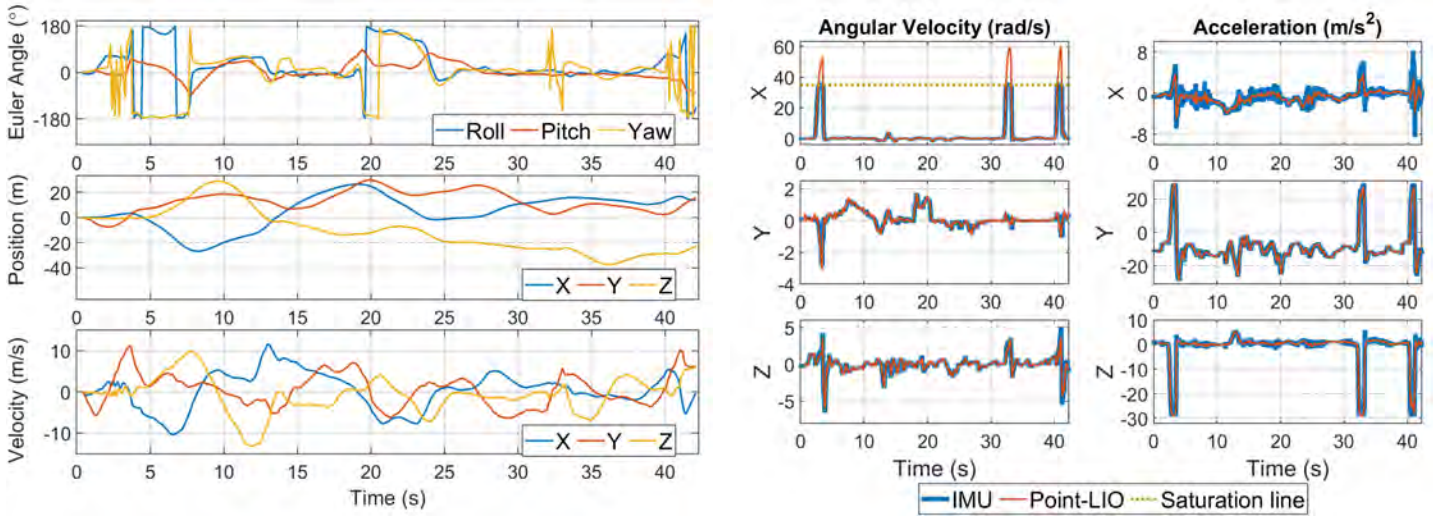


Figure 19: Mapping results of Point-LIO.

tion, the LiDAR also occasionally faces the sky, which causes no LiDAR measurements, during the maneuvering. Still, our method is able to estimate the state stably.



(a) (b)

Figure 20: (a) State estimation of Point-LIO. (b) Comparison of angular velocity and acceleration between estimation from Point-LIO and measurements from IMU. The gray dotted lines indicate saturation values of the IMU.

7.2 Self-rotating UAV

Figure 18 (b) presents the self-rotating UAV. As shown in this figure, the Livox Avia LiDAR is front, leading to a fast FoV change when the UAV undergoes continuous yaw rotation. We use the LiDAR built-in IMU and set the measuring range to 17.5 rad s^{-1}, while the average yaw rate of the UAV is around 25 rad s^{-1}. The UAV is also equipped with an low-power, ARM-based computer, Khadas VIM3 Pro, which has 2.2 GHz quad-core Cortex-A73 CPU and 4 GB RAM. The onboard computer runs our Point-LIO to estimate the UAV state in real-time. The estimated state is then fed to the flight controller, Pixhawk 4 Mini, to do the real-time control task. Two experiments are conducted using this UAV, one is an outdoor experiment outside the Haking Wong building of the University of Hong Kong, as shown in Figure 21 (a1) and (a2), and the other is an indoor experiment in a cluttered laboratory shown in Figure 21

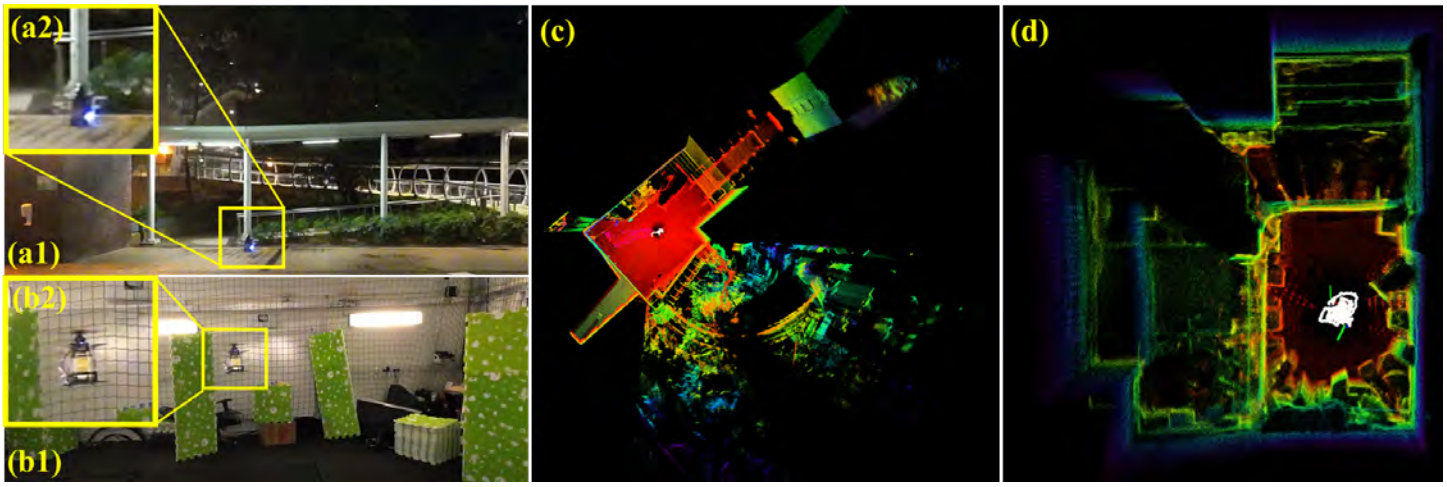
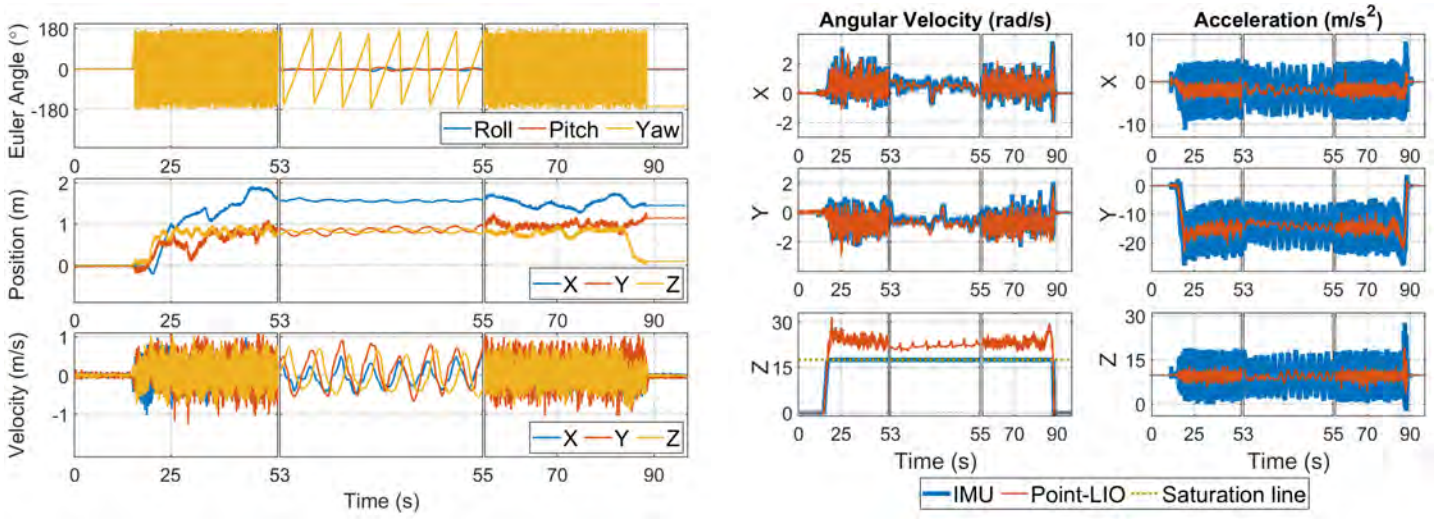


Figure 21: Mapping results of Point-LIO.

(b1) and (b2).

The real-time mapping results of the two experiments are shown in Figure 21, i.e., (c) for the outdoor experiment, and (d) for the indoor experiment. It is seen that our method manages to build maps of both environments without noticeable mismatches. Figure 22 (a) and Figure 22 (b) further show the estimation of the kinematic state (i.e., rotation, position, and velocity), angular velocity and acceleration, respectively. The plots are enlarged during the time interval $53 - 55s$, to better show the estimation results. As can be seen, our method can produce stable state estimate that are in line with the IMU measurements, despite the IMU saturation in the middle and quick FoV changes caused by the fast spinning. The average processing time per LiDAR package of our method is 14.63 ms while the package rate is 50 Hz, ensuring real-time performance. This is also confirmed by the experiments, where the controller is able to perform stable controlled flight with the state feedback from our method.



(a) (b)

Figure 22: (a) State estimation by the Point-LIO. (b) Comparison of angular velocity and acceleration between estimation from Point-LIO and measurements from the IMU. The gray dotted lines indicate the saturation values of the IMU.

8 Conclusion

This paper proposes Point-LIO, a robust and high-bandwidth LIO framework. The framework is based on a novel point-by-point update scheme, which updates the system state at the true sampling time of each point without accumulating points into a frame. The elimination of points accumulation removes the long-standing in-frame motion distortion and allows high odometry output at nearly the point sampling rate (4-8 kHz), which further enables the system to track very fast motions. To further boost the system bandwidth beyond the IMU measuring range, a colored stochastic process is augmented into the kinematic model treating the IMU measurements as system output. The bandwidth, robustness, accuracy and computation efficiency of the developed system are exhaustively tested in real-world experiments with extremely aggressive motions and public datasets with diversified LiDAR types, environments, and motion patterns. In all tests, Point-LIO achieves comparable computation efficiency and odometry accuracy to other state-of-the-art LIO algorithms while significantly boosting the system bandwidth. As an odometry, Point-LIO could be used in various autonomous tasks, such as trajectory planning, control, and perception, especially in cases involving very fast ego-motions (e.g., in the presence of severe vibration and high angular or linear velocity) or requiring high-rate odometry output and mapping (e.g., for high-rate feedback control and perception). Furthermore, in the future, the point-by-point strategy could be explored in the LiDAR systems without inertial measurements, and dynamic object detection systems.

Acknowledgements
This work is supported by Information Science Academy of China Electronics Technology Group Corporation (ISA CETC), and the project number is 200010756.

References

[1] M. Bosse, R. Zlot, *The International Journal of Robotics Research* **2008**, *27*, 6 667.

[2] R. M. Eustice, H. Singh, J. J. Leonard, *IEEE Transactions on Robotics* **2006**, *22*, 6 1100.

[3] F. Gao, W. Wu, W. Gao, S. Shen, *Journal of Field Robotics* **2019**, *36*, 4 710.

[4] F. Kong, W. Xu, Y. Cai, F. Zhang, *IEEE Robotics and Automation Letters* **2021**, *6*, 4 7869.

[5] R. W. Wolcott, R. M. Eustice, *The International Journal of Robotics Research* **2017**, *36*, 3 292.

[6] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, In *2011 IEEE intelligent vehicles symposium (IV)*. IEEE, **2011** 163–168.

[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *Journal of field Robotics* **2006**, *23*, 9 661.

[8] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *Journal of field Robotics* **2008**, *25*, 8 425.

[9] D. Wang, C. Watkins, H. Xie, *Micromachines* **2020**, *11*, 5 456.

[10] Z. Liu, F. Zhang, X. Hong, *IEEE/ASME Transactions on Mechatronics* **2021**, *27*, 1 58.

[11] K. Li, M. Li, U. D. Hanebeck, *IEEE Robotics and Automation Letters* **2021**, *6*, 3 5167.

[12] D. Frossard, S. Da Suo, S. Casas, J. Tu, R. Urtasun, In *Conference on Robot Learning*. PMLR, **2021** 1174–1183.

[13] W. Han, Z. Zhang, B. Caine, B. Yang, C. Sprunk, O. Alsharif, J. Ngiam, V. Vasudevan, J. Shlens, Z. Chen, In *European Conference on Computer Vision*. Springer, **2020** 423–441.

[14] W. Xu, D. He, Y. Cai, F. Zhang, *IEEE Transactions on Control Systems Technology* **2022**.

[15] J. Zhang, S. Singh, In *Robotics: Science and Systems*, volume 2. Berkeley, CA, **2014** 1–9.

[16] T. Shan, B. Englot, In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2018** 4758–4765.

[17] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, M. Liu, In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, **2020** 8899–8906.

[18] H. Ye, Y. Chen, M. Liu, In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, **2019** 3144–3150.

[19] Y. Pan, P. Xiao, Y. He, Z. Shao, Z. Li, In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2021** 11633–11640.

[20] J. Lin, F. Zhang, In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2020** 3126–3131.

[21] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, D. Rus, In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, **2020** 5135–5142.

[22] A. Tagliabue, J. Tordesillas, X. Cai, A. Santamaria-Navarro, J. P. How, L. Carlone, A.-a. Aghamohammadi, In *International Symposium on Experimental Robotics*. Springer, **2020** 380–390.

[23] C. Park, S. Kim, P. Moghadam, C. Fookes, S. Sridharan, In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. **2017** 2418–2426.

[24] J. Behley, C. Stachniss, In *Robotics: Science and Systems*, volume 2018. **2018** 59.

[25] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, C. Stachniss, In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2019** 4530–4537.

[26] J. Quenzel, S. Behnke, In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2021** 5499–5506.

[27] C. Yuan, X. Liu, X. Hong, F. Zhang, *arXiv preprint arXiv:2109.07082* **2021**.

[28] W. Xu, F. Zhang, *IEEE Robotics and Automation Letters* **2021**, *6*, 2 3317.

[29] W. Xu, Y. Cai, D. He, J. Lin, F. Zhang, *IEEE Transactions on Robotics* **2022**.

[30] M. Karimi, M. Oelsch, O. Stengel, E. Babaians, E. Steinbach, *IEEE Robotics and Automation Letters* **2021**, *6*, 2 2248.

[31] C. Qu, S. S. Shivakumar, W. Liu, C. J. Taylor, In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, **2022** 4149–4155.

[32] S. Hong, H. Ko, J. Kim, In *2010 IEEE International Conference on Robotics and Automation*. IEEE, **2010** 1893–1898.

[33] F. Moosmann, C. Stiller, In *2011 ieee intelligent vehicles symposium (iv)*. IEEE, **2011** 393–398.

[34] H. Dong, T. D. Barfoot, In *Field and Service Robotics*. Springer, **2014** 327–342.

[35] S. Anderson, T. D. Barfoot, In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, **2013** 2093–2099.

[36] C. H. Tong, T. D. Barfoot, In *2013 IEEE International Conference on Robotics and Automation*. IEEE, **2013** 5204–5211.

[37] S. Anderson, T. D. Barfoot, In *2013 IEEE International Conference on Robotics and Automation*. IEEE, **2013** 1033–1040.

[38] S. Zhao, Z. Fang, H. Li, S. Scherer, In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2019** 1285–1292.

[39] H. Wang, C. Wang, C.-L. Chen, L. Xie, In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2021** 4390–4396.

[40] R. Zlot, M. Bosse, *Journal of Field Robotics* **2014**, *31*, 5 758.

[41] L. Kaul, R. Zlot, M. Bosse, *Journal of Field Robotics* **2016**, *33*, 1 103.

[42] D. Droeschel, S. Behnke, In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2018** 5000–5007.

[43] S. Anderson, T. D. Barfoot, In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2015** 157–164.

[44] C. Le Gentil, T. Vidal-Calleja, S. Huang, In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, **2019** 6388–6394.

[45] C. Le Gentil, T. Vidal Calleja, S. Huang, *IEEE Transactions on Robotics* **2020**, *37*, 1 275.

[46] Y. Zhang, *IEEE Transactions on Aerospace and Electronic Systems* **2022**, *58*, 4 2649.

[47] J. Tang, Y. Chen, X. Niu, L. Wang, L. Chen, J. Liu, C. Shi, J. Hyyppä, *Sensors* **2015**, *15*, 7 16710.

[48] J. Tang, Y. Chen, A. Jaakkola, J. Liu, J. Hyyppä, H. Hyyppä, *Sensors* **2014**, *14*, 7 11805.

[49] W. Zhen, S. Zeng, S. Soberer, In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, **2017** 6240–6245.

[50] G. Hemann, S. Singh, M. Kaess, In *2016 IEEE/RSJ International conference on intelligent robots and systems (IROS)*. IEEE, **2016** 1659–1666.

[51] J. A. Hesch, F. M. Mirzaei, G. L. Mariottini, S. I. Roumeliotis, In *2010 IEEE International Conference on Robotics and Automation*. IEEE, **2010** 5376–5382.

[52] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, S. Sridharan, In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2018** 1206–1213.

[53] P. Geneva, K. Eckenhoff, Y. Yang, G. Huang, In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2018** 123–130.

[54] C. Forster, L. Carlone, F. Dellaert, D. Scaramuzza, *IEEE Transactions on Robotics* **2016**, *33*, 1 1.

[55] D. He, W. Xu, F. Zhang, *arXiv preprint arXiv:2102.03804* **2021**.

[56] B. M. Bell, F. W. Cathey, *IEEE Transactions on Automatic Control* **1993**, *38*, 2 294.

[57] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *Aaai/iaai* **2002**, *593598*.

[58] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, S. Singh, *Autonomous Robots* **2012**, *33*, 1 189.

[59] Z. Yan, L. Sun, T. Krajník, Y. Ruichek, In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2020** 10697–10704.

[60] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, L.-T. Hsu, In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2020** 2310–2316.

[61] J. Paulos, M. Yim, In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, **2013** 1374–1379.

[62] Y. Qin, N. Chen, Y. Cai, W. Xu, F. Zhang, *IEEE/ASME Transactions on Mechatronics* **2022**.

Dongjiao He received the B.S. degree in Mechanical Engineering, in 2016, from Southeast Univeristy, Nanjing, China, and M.S. degree in Mechanical Engineering, in 2019, from Shanghai Jiao Tong University, Shanghai, China. She is currently a Ph.D. candidate in Department of Mechanical Engineering, University of Hong Kong, Hong Kong, China. Her research interests include mechatronics and robotics, with focus on sensor fusion and LiDAR based navigation.

Wei Xu received the B.S. and M.S. degree in Aerial Vehicle Design from the Beihang University, Beijing, China, in 2015 and 2018 respectively. He is currently a Ph.D. candidate in the Department of Mechanical Engineering, University of Hong Kong, Hong Kong, China. His research interests include advanced aerial robots control, sensor fusion, and LiDAR SLAM.

Nan Chen received the B.Eng. degree in automation and the M.Eng. degree in control theory and control engineering from the Shenzhen University, Shenzhen, China, in 2015 and 2019, respectively. He is currently working toward the Ph.D. degree in UAV design, control, and SLAM application with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong, China. His research interests include design, system identification, and control on unmanned aerial vehicles.

Fanze Kong received the B.S. degree in Flight Vehicle design and engineering from Harbin Institute of Technology, Harbin, China, in 2020. He is currently working toward the Ph.D. degree in Mechanical Engineering with the Department of the University of Hong Kong, Hong Kong, China. His research interests include UAV design, UAV motion planning and autonomous navigation.

Chongjian Yuan received the B.Eng. degree in Automation from Zhejiang University (ZJU), Hangzhou, Zhejiang, China, in 2016. He is currently a Ph.D. student with the University of Hong Kong, Hong Kong and his research interests includes LiDAR slam and sensor calibration.

Fu Zhang received the B.E. in Automation, in 2011, from the University of Science and Technology of China (USTC), China, and the Ph.D. degree in Mechanical Engineering, in 2015, from University of California, Berkeley, USA. He joined the University of Hong Kong, in 2018, where he is currently an assistant professor in Mechanical Engineering. His research interests include mechatronics and robotics, with focus on UAV design, control, and LiDAR-based navigation.

Table of Contents A high-bandwidth LiDAR-inertial system (LIO) is presented. The system updates the state at the sampling time of each LiDAR point or IMU measurement without accumulating a frame. The system is able to output a high-rate (4 kHz - 8 kHz), high-bandwidth (over 150Hz) odometry and handles extremely aggressive motions where IMU saturates.

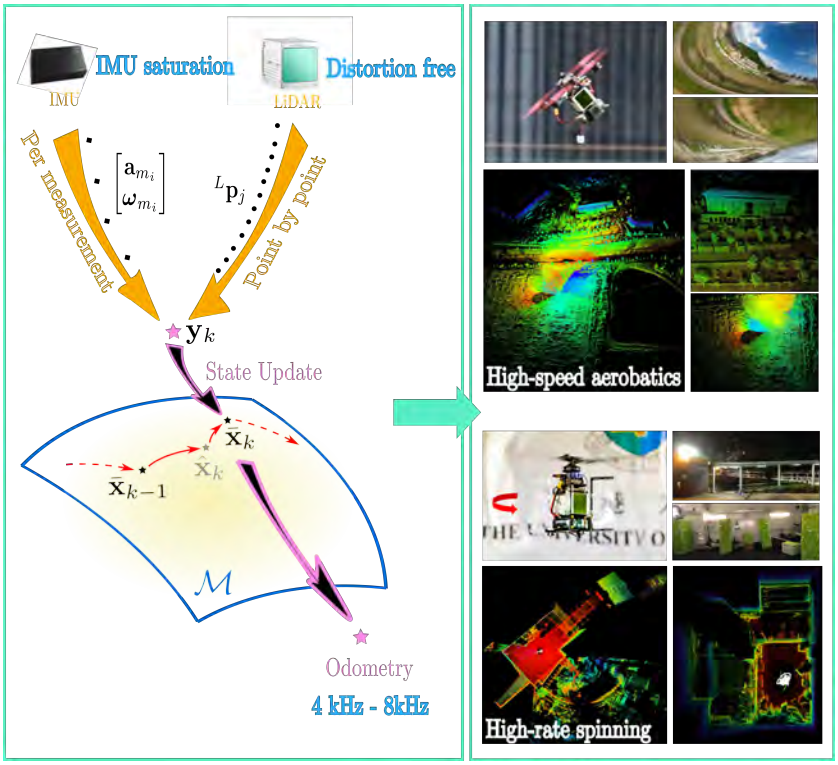


Figure 23: Table of content