FAST-LiDAR-SLAM: A Robust and Real-Time Factor Graph for Urban Scenarios With Unstable GPS Signals

Zhi Rui, Wangtu Xu, and Zhen Feng

***Abstract*— This paper introduces FAST-LiDAR-SLAM: a robust, fast, and versatile LiDAR SLAM framework, which is specifically designed for urban scenarios with unstable GPS signals. The framework establishes a flexible and compact factor graph model that incorporates various relative and absolute measurements (including loop closure and GPS) as factors to optimize state estimation and dense mapping. Two key innovations are presented in this work. Firstly, it adopts a direct point cloud registration and map updating system in contrast to the traditional feature-based SLAM. In this way, the system can effectively exploit the fine structure of the environment and maintain robustness when confronted with non-structural road scenes. Secondly, an improved strong tracking extended Kalman filter (STEKF) is proposed to exclude anomalous GPS data in the back-end processing. Experiments show that FAST-LiDAR-SLAM achieves single-frame pose estimation in 20 *ms* for solid-state LiDAR tests, outperforming Faster-LIO in accuracy. For rotating LiDAR, FAST-LiDAR-SLAM achieves comparable accuracy to LIO-SAM while significantly improving computational efficiency.**

***Index Terms*— Simultaneous localization and mapping (SLAM), LiDAR, autonomous driving, factor graph.**

I. Introduction

AUTONOMOUS vehicles are increasingly being deployed for various functions such as airport transportation, sightseeing, factory patrols, cargo handling, and security monitoring [1]. As technology advances, these vehicles are transitioning from simple closed environments to complex urban road scenarios. However, safety concerns arise due to the accurate and reliable localization being a prerequisite for successful navigation of autonomous vehicles. Currently, widely used localization methods for autonomous vehicles include combining Global Positioning System (GPS) with Inertial Navigation System (INS) and map-aided algorithms [2]. However, the former method is costly and prone to defects in scenarios with unstable GPS signals, such as parking lots, staggered viaducts, tall building occlusions, and indoor environments. In addition, INS encounters challenges related to initialization and cumulative errors, and high-precision inertial navigation elements are expensive [3]. In recent years, the matching and positioning of autonomous vehicles with High Definition Map (HD Map) have developed rapidly [4]. Nonetheless, the construction of HD Maps demands costly mapping systems and complex post-processing steps, such as manual labeling, which restricts the application of autonomous vehicles in general scenarios. Therefore, there is a growing need for an industrial-grade localization and mapping system that is low-cost, reliable, and robust in complex environments to facilitate the development of autonomous driving systems.

To solve the above problems, Simultaneous Localization and Mapping (SLAM) technology, a crucial module of robot navigation, is employed to tackle the localization problem in autonomous driving [5], [6]. Vision-based methods, which utilize monocular or stereo cameras to triangulate consecutive image features and estimate camera motion, are well-suited for place recognition. However, their reliability as standalone solutions for supporting autonomous navigation systems is limited due to their sensitivity to initialization, illumination, and range [7]. In contrast, LiDAR-based methods are less affected by illumination variations. Recent advancements in LiDAR technology have led to the commercialization and mass production of solid-state LiDAR devices with reduced cost, weight, and size. They are comparable to vision cameras in terms of cost, yet offering high performance with centimeter-level accuracy within a range of hundreds of meters [8]. This field has aroused considerable interest of researchers [9], [10], [11]. The advent of such cost-effective solid-state LiDARs, like Livox HAP-144 and Velodyne Velarray H800, enables the direct capture of 3D spatial environment details, which has the potential to benefit emerging LiDAR-based robotic applications. Therefore, this paper will focus on the LiDAR-based state estimation and mapping methods.

Since LiDAR points are usually sampled sequentially during continuous motion, this process brings significant motion distortion. And registration using only tilted point clouds or features will eventually lead to large drift and pose estimation errors. Classical LiDAR-based localization systems for autonomous driving use information from Inertial Measurement Unit (IMU) and wheel speed meter as odometry. Real Time Kinematic (RTK) and point cloud localization are used as global observations. On this basis, the Error State Kalman Filter (ESKF) is used for fusion and output results [2], [3], [12]. However, as the number of sensors increases, the arrangement and combination of different sensor states lead to exponential

Manuscript received 1 September 2023; revised 8 March 2024 and 11 July 2024; accepted 28 August 2024. Date of publication 17 September 2024; date of current version 27 November 2024. The Associate Editor for this article was C. K. Sundarabalan. *(Corresponding author: Wangtu Xu.)*

The authors are with the School of Architecture and Civil Engineering, Xiamen University, Xiamen 361005, China (e-mail: ato1981@xmu.edu.cn).

Digital Object Identifier 10.1109/TITS.2024.3455263

growth in the localization state table, increasing computational pressure. This will hinder the ability of autonomous vehicles to handle high dynamic road scenarios effectively, even though many of them are stable at low speeds. To address these limitations of traditional fusion localization methods and ensure real-time performance, LiDAR Odometry (LO) and LiDAR inertial Odometry (LIO) have been proposed. LO is typically some form of registration between scan-to-scan or scan-to-map. Initially, it was improved based on registration algorithms like ICP [13] and NDT [14]. The emergence of LOAM [15] inspired a series of subsequent works to do registration by extracting geometric features. However, using LO alone cannot remove point cloud distortion adequately and struggles with many degraded scenes. The loosely coupled framework of LIO overcomes distortion by fusing IMU data, which provides initial values for point cloud registration, and incorporates gravity direction observations. It can directly output point cloud registration results or use filters to fuse observations. Nevertheless, the fusion of LiDAR and IMU in this scheme is mainly reflected at the result level, without considering the inherent constraints between the two observations, and still cannot deal with some degraded scenes [15], [16].

In this paper, we present a tightly coupled and efficient LiDAR-inertial-odometry framework that calibrates LiDAR scans using raw IMU measurements. Then we use state-dimension-based Iterated Extended Kalman Filter (IEKF) to output the registration results of global point clouds. In the back-end processing, we incorporate corrected GPS data and loop constraints to jointly optimize the graph and pose estimation. In addition, we perform scan matching at compact local scale and apply sliding windows to build a sub-map (register new keyframes to a fixed-size set of previous sub-keyframes) to enhance computational efficiency. The main contributions of this work can be summarized as follows:

- By tightly coupling IMU and LiDAR data, an IESKF-based motion estimation model is constructed to output accurate odometry information, serving as the initial guess for optimization. This method makes full use of global point clouds and exhibits superior capability in recovering scene details compared to traditional feature extraction method.
- An improved strong tracking extended Kalman filter (STEKF) approach is proposed to optimize GPS data. This method enhances the whole system's ability to handle GPS signal interruptions, particularly in urban scenarios where obstructions, dynamic interference, and multipath effects are prevalent.
- A flexible and robust factor graph model is developed, which is suitable for multi-sensor fusion and global optimization, while maintaining excellent performance compared to state-of-the-art (SOTA) algorithms.
- FAST-LiDAR-SLAM system is extensively tested and validated on different road environments, vehicles, and hardware configurations. It exhibits robustness in LiDAR scanning tasks, even in scenarios lacking salient features, such as small fields of view (FoV) and unstructured environments.

This paper is organized as follows: Section II discusses the related work in the field of LiDAR SLAM. Section III provides an overview of our factor graph system, including the details of factor construction. Section IV introduces the experimental settings, the evaluation results and some discussions. Section V concludes our work.

II. Related Work

Due to the local sparsity of 3D LiDAR points, the large amount of data (hundreds of thousands of points per second) and the noise caused by dynamic objects, traditional registration methods such as ICP [13] or NDT [14] are difficult to meet the efficient computational demands of SLAM algorithm [5]. Therefore, Zhang and Singh [15] proposed LOAM based on the registration of edge and plane features, which improved the odometry accuracy through the loose coupling of two-axis LiDAR and IMU, and compensated the motion distortion of LiDAR. They divide the complex SLAM problem into high-frequency motion estimation module and low-frequency environment mapping module (similar to that tracking and mapping are divided into two threads in visual SLAM since PTAM [17]), realizing low-drift and real-time odometry. Most of the existing 3D LiDAR SLAM studies follow the LOAM structure, including feature extraction based on local smoothness measurement, general odometry, and dense mapping. However, the performance of feature extraction will be limited in large-scale scenes that lack effective structural information (e.g., Bridges, tunnels, single object edging, and circular buildings). This condition is greatly worse if the LiDAR FoV is small, which is common in emerging solid-state LiDARs [18]. Moreover, different LiDAR scanning patterns (e.g., mechanical rotation [15], optical phased array [19], prism vibration [18], MEMS [20]) and point densities introduce variations in point cloud formats and feature extraction methods, necessitating substantial manual efforts to adapt the original SLAM method to different hardware. To overcome these challenges, this paper uses the global point cloud instead of the traditional line-surface features. We refer to this raw point-based registration as a direct approach, which is similar to visual SLAM. By directly registering the raw point cloud to the map, the subtle features in the environment can be fully utilized, thus improving the accuracy while maintaining compatibility with different LiDARs.

Complex urban roads bring challenges for deploying autonomous driving systems, which are characterized by highly dynamic scenarios and unstable GPS signals [3]. The localization module based on the integration of GPS and INS exhibits unreliability in urban environment due to satellite signal obstruction by skyscrapers, tree canopies, and walls. To enhance the perception ability and achieve long-term reliability of mapping, these works [21], [22], [23] focus on fusing multi-sensor data, developing deep learning models, and incorporating semantic priors. However, the increased hardware cost and computational resource overhead will limit the expansion of autonomous vehicles. Recent researches focus on fast point cloud registration methods that identify and remove dynamic points in real-time during the registration process. Inspired by the well-known NDT [14], LiTAMIN [24]

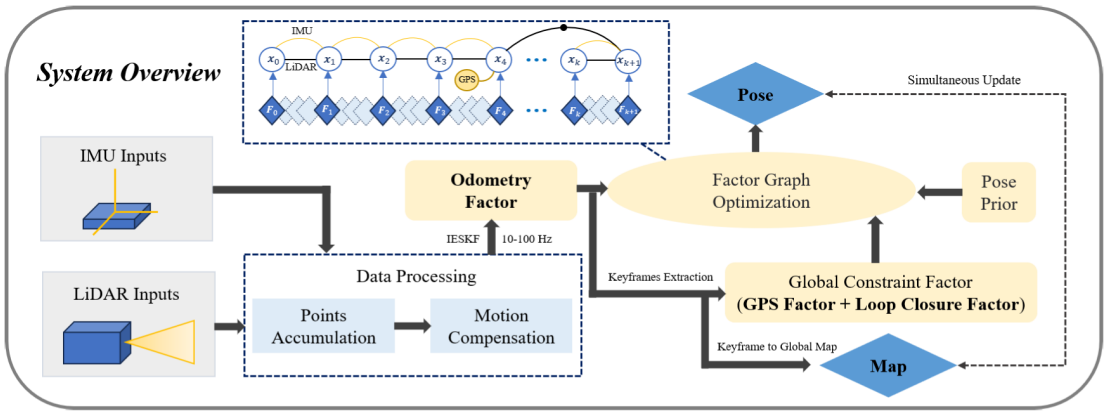


Fig. 1. System overview of FAST-LiDAR-SLAM.

and LiTAMIN2 [25] proposed a fast registration method that reduces the number of registration points and introduces the symmetric Kullback-Leibler divergence into ICP [13]. They first partitioned the points into individual voxels and then performed a normal distribution transformation in each voxel. Ye et al. [26] proposed a tightly coupled LIO framework, adding point cloud information to the dead reckoning module of traditional positioning. Common open-source solutions include LIO-SAM [27], FAST-LIO [10], FAST-LIO2 [11], LiLi-OM [8] and Faster-LIO [9], which are all based on tightly coupled LIO schemes. In these schemes, IMU data is first used to recover the distorted LiDAR point, then both are jointly optimized using a global state estimation model to minimize overall errors and estimate final state variables, such as position, velocity, orientation, bias, and gravity. Experiments show that the tightly coupled LIO system performs better than traditional LO systems in degraded scenes [9], [10], [11], [27], [28]. However, the estimation of the gravity direction of the LIO system may be coupled with the IMU bias, leading to global attitude instability during long-term operation. Therefore, learning from the traditional ESKF navigation scheme, we improve GPS and loop observations and integrate them into the constructed factor graph, making the LIO system more robust.

LiDAR usually has a measurement range of hundreds of meters, but the density of line bundles in a single scan is relatively low compared to a complete 3D space. As a result, a large and dense map is required to register these sparse points. Since registration methods usually rely on k-nearest neighbor search in point clouds, registration efficiency can be improved by exploiting faster k-NN data structures [28]. The main advantage of the k-d tree is its ability to provide rigorous k-NN and range search results by conditionally splitting the high-dimensional space with hyperplanes [29]. To reduce the computation of FAST-LiDAR-SLAM, we follow the ikd-Tree design of FAST-LIO2 [11] and downsample the map points at the specified resolution. This data structure supports incremental map updates (i.e., point insertions, upsampling, downsampling, and point deletions) and dynamic balancing of the global map with minimal computational cost. Based on this, we only need to build a sparse map of key points and guarantee that they fall in the FoV. Since the factor graph optimization framework is used in this paper, joint estimation of state variables at multiple moments is possible for higher accuracy. However, traditional factor graph framework suffers from low computational efficiency, and early SLAM based on factor graph cannot guarantee real-time performance [30]. LIOM [27] is one of the important works that apply sliding window optimization to LIO, which constructs the local map and the constraint relationship between adjacent frames respectively to form a complete sliding window, and finally realizes reasonable resource consumption. In FAST-LiDAR-SLAM, the keyframe is adopted to further improve computational speed, which is widely used in visual SLAM [31]. Instead of matching all scans to a global map, keyframes are extracted based on major pose changes, and old LiDAR scans are marginalized for pose optimization. As a result, FAST-LiDAR-SLAM achieves faster computational efficiency while maintaining the same accuracy as other advanced multi-sensor fusion SLAM systems.

III. FAST-LiDAR-SLAM

A. System Overview

The framework of FAST-LiDAR-SLAM is shown in Fig. 1. This system receives sensor data from LiDAR, IMU, and GPS to estimate the state and trajectory of the autonomous vehicle. We use factor graphs to model this problem because they are more suitable for large-scale tasks than traditional Bayesian networks [27]. Moreover, the proposed multi-sensor fusion framework is not limited to the optimization based on a certain class of hardware configurations, but focuses on providing a general paradigm for fusing more sensors (depth camera, millimeter wave radar, wheel speed meter, etc.) to remain robust in complex environments. We introduce three types of factors and one type of variable to build the factor graph. The variable represents the vehicle's pose at a specific time and is attributed to the nodes of the graph, while the edges of the graph represent constraints between poses.

Since processing each LiDAR frame is computationally unreasonable, we created a keyframe selection module. This module selects the current LiDAR frame as a keyframe and associates it with a new vehicle state node in the factor graph if the pose transformation between the current frame and the

previous frame exceeds a user-set threshold. Subsequently, a new state node will be added to the graph by incremental smoothing and Bayesian tree mapping [32], while several LiDAR frames between two keyframes are discarded. In this work, the position and rotation change thresholds for adding new keyframes are set to 1 m and 12°, and the registration of keyframes to the map uses ICP [13] with a threshold of 0.3 m (scene complexity, motion characteristics, hardware constraints, and dataset-specific attributes are considered). Adding keyframes helps with efficient nonlinear optimization tasks because it ensures a sparse factor graph. Subsequently, we keep track of all map points within a cube region of a fixed size and use it in the state estimation module to compute the residuals (assuming the region is smooth enough). If the current LiDAR FoV range crosses the map boundary, the corner points of the bounding box need recalculation and the historical points corresponding to the map area farthest from the sensor location are deleted from the ikd-Tree. This approach ensures that the map maintains relevant points and is responsive to changes in the LiDAR FoV.

In the final step of the system, the new state and covariance obtained from the optimization are used to update the information of all variable nodes in the factor graph. This update includes the pose of all historical keyframes, effectively correcting the odometry trajectory. For the point cloud that needed to be added to the ikd-Tree, we decide whether to downsample the points based on the distance between the point and the center of the bounding box: If a point is farther from the center in X, Y, and Z dimensions than the half axis of the map grid, downsampling is not necessary. We refer to this step as the extension of ikd-Tree.

The factors include Odometry Factor, GPS Factor, and Loop Closure Factor. Each of them plays an important role in the optimization process and contributes to the accuracy and reliability of the state estimation and mapping results. The specific process of generating these factors is described detailly in the paper, providing a comprehensive understanding of their functions and contributions to the FAST-LiDAR-SLAM framework. The mathematical notations used throughout the paper are defined and summarized in Table I.

TABLE I
EXPLANATION OF IMPORTANT SYMBOLS

Symbols	Meaning
I	The IMU frame.
W	The world frame (take the first IMU frame as the origin).
L	The LiDAR frame.
${}^{L}\mathbf{T}_{I}$	The extrinsic parameter matrix describing LiDAR to IMU.
$\lfloor \boldsymbol{a} \rfloor_{\tau}$	The antisymmetric matrix of the vector $\boldsymbol{a} \in \mathbb{R}^3$, which maps the cross product operation.
${}^{W}\mathbf{p}_{I}, {}^{W}\mathbf{R}_{I}$	The IMU position and attitude in the world frame.
${}^{w}\mathbf{g}$	The gravity vector in the world frame.
$\mathbf{a}_m$, $\boldsymbol{\omega}_m$	The linear acceleration obtained from the accelerometer and the angular velocity obtained from the gyroscope.
$\mathbf{n}_a, \mathbf{n}_\omega$	The Gaussian noise of $\boldsymbol{a}_m$ and $\boldsymbol{\omega}_m$ measurements.
$\mathbf{b}_a$, $\mathbf{b}_\omega$	The acceleration drift and the angular velocity drift, which are modeled as IMU bias of a random walk driven by $\boldsymbol{n}_{b_a}$ and $\boldsymbol{n}_{b_\omega}$. The derivatives of and satisfy Gaussian distributions.
Δt	The sampling period of the IMU.
ρ_j	The sampling time of the first point in the LiDAR scan.
i	The index of the IMU measurement.
k	The index of the LiDAR scan frame.
$\mathbf{x}, \hat{\mathbf{x}}, \bar{\mathbf{x}}$	The truth value of vector $\mathbf{x}$, the propagation value during filtering, and the optimal value after filtering.
$\widetilde{\mathbf{x}}$	The error between $\mathbf{x}$ and optimal state estimation $\bar{\mathbf{x}}$, which is the main input of IESKF.
$\mathbf{P}$	The covariance matrix with respect to $\widetilde{\mathbf{x}}$, which is used to represent the confidence of the error state propagation.
${}^{L}\mathbf{p}_j$	The j-th LiDAR point in a scan.
${}^{L}\mathbf{n}_j$	The ranging and beam-guiding noise of ${}^{L}\mathbf{p}_j$.
κ	The current number of iterations of IESKF.

B. Odometry Factor

This factor accounts for the relative pose transformation between consecutive keyframes based on IMU and LiDAR information. Due to the high sampling rate of raw LiDAR points (e.g., 200 kHz), processing them individually upon receipt is not practical. A more feasible approach is to accumulate these points over a certain time and process them as a single unit, referred to as a scan [10]. We refer to the processing time as t_k. However, this method of accumulating points will result in motion distortion, potentially causing registration failure when trying to align the sampled points with the global map. Combined with the scanning characteristics of solid-state LiDAR, this paper proposes an odometry that tightly couples LiDAR and IMU measurements. We use the propagation of IMU to calibrate the point ρ_j collected at different times during the scanning process, as shown in Fig. 2 (a). The IMU propagation compensates for motion distortion, enabling more accurate and reliable registration for the global point cloud map.

By defining the generalized update operations $\oplus$ and $\ominus$ (see Appendix A), we can map the changes in the state of an iteration to the changes in the true state. Following the linear optimization, we obtain the optimal state increment, and then update the current state in the form of $\mathbf{x}_k = \hat{\mathbf{x}}_k^{\kappa} \oplus \widetilde{\mathbf{x}}_k^{\kappa}$. This iterative operation is repeated until the system converges. Then we can obtain the required optimal solution.

1) Data Preprocessing: To remove noise from LiDAR measurements, we can use filtering techniques such as statistical outlier removal or voxel grid downsampling. We downsample the scan frames to eliminate duplicate points falling in the same grid cell, and the resolution was set to 0.4 m. Measurement noise is inevitable, thus, removing this noise will result in the true point location ${}^{L}\mathbf{p}_j^{gt}$ in L:

$$ {}^{L}\mathbf{p}_j^{gt} = {}^{L}\mathbf{p}_j - {}^{L}\mathbf{n}_j. \tag{1} $$

After projecting the point from the local frame to the world frame using the corresponding LiDAR pose ${}^{W}\mathbf{T}_{I_k} = \left({}^{W}\mathbf{R}_{I_k}, {}^{W}\mathbf{p}_{I_k}\right)$ and external parameter ${}^{L}\mathbf{T}_{I}$, the true points should be located exactly on the local small plane in the map [29] (see Fig. 2 (b)):

$$ \mathbf{0} = {}^{W}\mathbf{u}_j^{T} \left({}^{W}\mathbf{T}_{I_k} {}^{L}\mathbf{T}_{I} \left({}^{L}\mathbf{p}_j - {}^{L}\mathbf{n}_j \right) - {}^{G}\mathbf{q}_j \right). \tag{2} $$

In the above equation, ${}^{W}\mathbf{u}_j$ is the normal vector of the corresponding plane and ${}^{W}\mathbf{q}_j$ is a point set lying on the plane. Furthermore, (2) can be summarized as a more compact implicit measurement model:

$$ \mathbf{0} = \mathbf{h}_j \left(\mathbf{x}_k \, , {}^{L}\mathbf{n}_j \right). \tag{3} $$

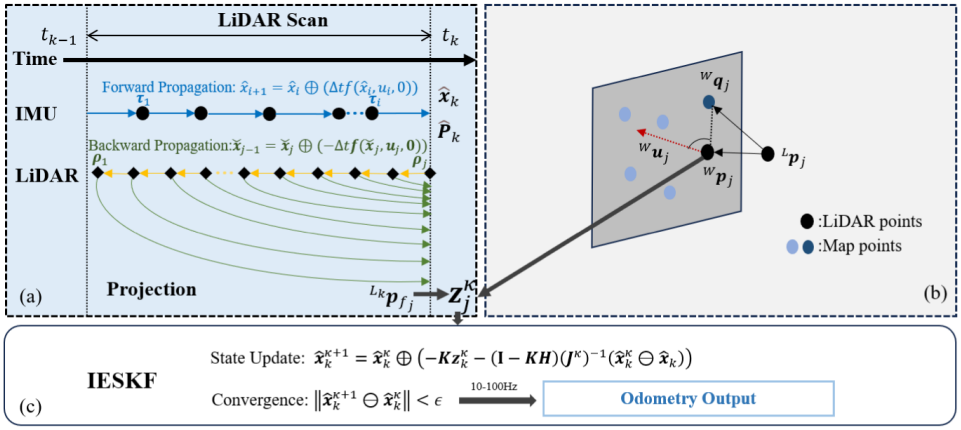


Fig. 2. System overview of odometry. (a) Propagation process. (b) The measurement model. (c) The filtering workflow.

2) Kinematic Model: Assuming that the IMU is rigidly connected to LiDAR, we can obtain the external parameter matrix ${}^{L}\mathbf{T}_I = ({}^{L}\mathbf{R}_I, {}^{L}\mathbf{p}_I)$ through physical measurements and online calibration, and the kinematic model can be established:

$$\begin{aligned} {}^{W}\dot{\mathbf{p}}_I &= {}^{W}\mathbf{V}_I, {}^{W}\dot{\mathbf{R}}_I = {}^{W}\mathbf{R}_I \left\lfloor {}^{I}\omega \right\rfloor_{\tau}, \\ {}^{W}\dot{\mathbf{V}}_I &= {}^{W}\mathbf{R}_I (\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + {}^{w}\mathbf{g}. \\ \dot{\mathbf{b}}_{\omega} &= \mathbf{n}_{b_{\omega}}, \ \dot{\mathbf{b}}_a = \mathbf{n}_{b_a}, {}^{I}\boldsymbol{\omega} = \boldsymbol{\omega}_m - \mathbf{b}_{\omega} - \mathbf{n}_{\omega}, \\ {}^{L}\dot{\mathbf{R}}_I &= \mathbf{0}, \ {}^{L}\dot{\mathbf{p}}_I = \mathbf{0}, {}^{w}\dot{\mathbf{g}} = \mathbf{0}. \end{aligned} \tag{4}$$

Based on the theoretical derivation of Xu et al. [10], given adjacent point cloud frames, the state of position, velocity and direction of a point in the world frame can be propagated by high-frequency inertial measurements. Therefore, we discretize the continuous kinematic model (4) to Δt:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \oplus (\Delta t \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)). \tag{5}$$

3) State Estimation: Based on the measurement model (4) and the state model (5) derived above, we adopt IESKF [10], [33], which operates directly on $\mathcal{M}$, and estimate the covariance in the tangent space of the state estimation. It consists of two key steps: propagation of high-frequency IMU measurements and iterative update of LiDAR frames.

Suppose the optimal state estimate for the last LiDAR frame (at $\mathbf{t}_k - 1$) is $\bar{\mathbf{x}}_{k-1}$ with a covariance matrix $\bar{\mathbf{P}}_{k-1}$. $\bar{\mathbf{P}}_{k-1}$ describes the confidence level of the following error state:

$$\begin{aligned} \widetilde{\mathbf{x}}_{k-1} &\doteq \mathbf{x}_{k-1} \ominus \bar{\mathbf{x}}_{k-1} \\ &= \left[\delta\boldsymbol{\theta}^T \ {}^{W}\widetilde{\mathbf{p}}_I^T \ {}^{W}\widetilde{\mathbf{v}}_I^T \ \widetilde{\mathbf{b}}_{\omega}^T \ \widetilde{\mathbf{b}}_a^T \ {}^{w}\widetilde{\mathbf{g}}^T \ {}^{L}\widetilde{\mathbf{R}}_I^T \ {}^{L}\widetilde{\mathbf{p}}_I^T\right]^T, \end{aligned} \tag{6}$$

where $\delta\boldsymbol{\theta} = \log({}^{W}\bar{\mathbf{R}}_I^T, {}^{W}\mathbf{R}_I)$ is the attitude error, which describes the small deviation between the true pose and the estimated pose. Given that the pose information is a 3D spatial vector, the main advantage of this definition is allowing us to use the smallest form of the covariance matrix (3×3) to characterize the pose uncertainty. The traditional Kalman filter uses quaternions (4D) or rotation matrices (9D), or representations with singularities (Euler angles). By adding the optimal estimation error quantity with the state quantity, the optimal result of the filter-based method ($\mathbf{x}_k = \hat{\mathbf{x}}_k^{\kappa} \oplus \widetilde{\mathbf{x}}_k^{\kappa}$) can be obtained.

a) Forward propagation: The forward propagation is performed when the IMU measurement reaches and continues until the end time of the new scan (i.e., k-th). We set the process noise $\mathbf{w}_i$ to $\mathbf{0}$ and propagate the state and covariance according to (5), resulting in:

$$\begin{aligned} \hat{\mathbf{x}}_{i+1} &= \hat{\mathbf{x}}_i \oplus \left(\Delta t \mathbf{f}\left(\hat{\mathbf{x}}_i, \ \mathbf{u}_i, \mathbf{0}\right)\right), \\ \hat{\mathbf{P}}_{i+1} &= \mathbf{F}_{\widetilde{\mathbf{x}}_i} \hat{\mathbf{P}}_i \mathbf{F}_{\widetilde{\mathbf{x}}_i}^T + \mathbf{F}_{\mathbf{w}_i} \mathbf{Q}_i \mathbf{F}_{\mathbf{w}_i}^T, \end{aligned} \tag{7}$$

where $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_0 = \bar{\mathbf{P}}_{k-1}$, $\mathbf{Q}_i$ is the covariance of the white noise $\mathbf{w}_i$, and the matrices $\mathbf{F}_{\widetilde{\mathbf{x}}_i}$ and $\mathbf{F}_{\mathbf{w}_i}$ are calculated in the Appendix C. To propagate the covariance, we build the error model of the motion equation:

$$\begin{aligned} \widetilde{\mathbf{x}}_{i+1} &= \mathbf{x}_{i+1} \ominus \hat{\mathbf{x}}_{i+1} \\ &= (\mathbf{x}_i \oplus \Delta t \mathbf{f}(\mathbf{x}_i, \ \mathbf{u}_i, \ \mathbf{w}_i)) \ominus \left(\hat{\mathbf{x}}_i \oplus \Delta t \mathbf{f}\left(\hat{\mathbf{x}}_i, \ \mathbf{u}_i, \ \mathbf{0}\right)\right) \\ &\simeq \mathbf{F}_{\tilde{x}} \widetilde{\mathbf{x}}_i + \mathbf{F}_{\mathbf{w}} \mathbf{w}_i. \end{aligned} \tag{8}$$

Since IMU measurements are usually obtained at a higher frequency than LiDAR scans, it is common to perform multiple propagation steps of IMU measurements before updating, see Fig. 2 (a). The forward propagation computes a rough state quantity $\hat{\mathbf{x}}_k$ by IMU integration, which will be used in the subsequent backward propagation and the prior of the IESKF, while output the error quantity $\widetilde{\mathbf{x}}_k$ and the covariance matrix $\mathbf{P}_i$.

b) Backward propagation: LiDAR usually samples point by point, and when the point cloud accumulation time interval reaches t_k, the new scan point should be fused with the propagation state $\mathbf{x}_k$ and covariance matrix $\mathbf{P}_k$ to generate the optimal state. To compensate for the motion $\check{\mathbf{x}}_{j-1} = \check{\mathbf{x}}_j \oplus \left(-\Delta t \mathbf{f}\left(\check{\mathbf{x}}_j, \mathbf{u}_j, \mathbf{0}\right)\right)$. Applying this method allows all scan points to be viewed as points sampled simultaneously at t_k.

Backward propagation is performed based on the frequency of the sampling points, which is usually much higher than the IMU measurement frequency (for a common 16-line Velodyne LiDAR, the point cloud is sampled at 10 Hz, whereas each laser beam usually has 2000 points, so the point acquisition frequency is 320 kHz). Therefore, for multiple points sampled between two IMU frames, we use the value at t_{k-1} as input for backward propagation. Moreover, since the last three rows

of $\mathbf{f}\left(\mathbf{x}_j, \mathbf{u}_j, \mathbf{0}\right)$ have zero elements, the backward propagation can be simplified as follows:

$$\begin{aligned} {}^{I_k}\check{\mathbf{p}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{p}}_{I_j} - {}^{I_k}\check{\mathbf{v}}_{I_j}\Delta t, \\ {}^{I_k}\check{\mathbf{v}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{v}}_{I_j} - {}^{I_k}\check{\mathbf{R}}_{I_j}\left(\boldsymbol{a}_{m_{i-1}} - \hat{b}_{\boldsymbol{a}_k}\right)\Delta t - {}^{I_k}\hat{\mathbf{g}}_k\Delta t, \\ {}^{I_k}\check{\mathbf{R}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{R}}_{I_j} Exp\left(\left(\hat{\mathbf{b}}_{\omega_k} - \boldsymbol{\omega}_{m_{i-1}}\right)\Delta t\right), \end{aligned} \tag{9}$$

where $\Delta t = \rho_j - \rho_{j-1}$. Backward propagation will produce a relative pose transformation between ρ_j and t_k :${}^{I_k}\check{\mathbf{T}}_{I_j} = \left({}^{I_k}\check{\mathbf{R}}_{I_j}, {}^{I_k}\check{\mathbf{p}}_{I_j}\right)$. This relative pose allows us to project the measurement ${}^{L_j}\mathbf{p}_{f_j}$ at one time into the measurement ${}^{L_k}\mathbf{p}_{f_j}$ at the end of the scan time t_k: ${}^{L_k}\mathbf{p}_{f_j} = {}^{L}\mathbf{T}_I^{-1}\,{}^{I_k}\check{\mathbf{T}}_{I_j}\,{}^{L}\mathbf{T}_I\,{}^{L_j}\mathbf{p}_{f_j}$. Subsequently, ${}^{L_k}\mathbf{p}_{f_j}$ will be used to build the residual in the next section.

c) Residual calculation: Suppose the estimate of state $\mathbf{x}_k$ at the current iteration update is $\hat{\mathbf{x}}_k^\kappa$. When $\kappa = 0$ (i.e., before the first iteration), $\hat{\mathbf{x}}_k^\kappa = \hat{\mathbf{x}}_k$. We project the compensated LiDAR points ${}^{L_k}\mathbf{p}_{f_j}$ into the world frame and search for its nearest five points in the map represented by the ikd-Tree [11], and then use the found nearest neighbors to fit the local small plane, see Fig. 2 (b). $\mathbf{z}_j^\kappa$ is the distance between ${}^{W}\hat{\mathbf{p}}_{f_j}^\kappa$ and the local plane. We refer to it as the residual:

$$\begin{aligned} \mathbf{z}_j^\kappa &= \mathbf{h}_j\left(\hat{\mathbf{x}}_k^\kappa\ ,\ {}^{L}\mathbf{n}_j\right) = \mathbf{u}_j^T\left({}^{W}\hat{\mathbf{T}}_{I_k}^\kappa\,{}^{L}\hat{\mathbf{T}}_{I_k}^\kappa\,{}^{L_k}\mathbf{p}_{f_j} - {}^{G}\mathbf{q}_j\right) \\ &= \mathbf{u}_j^T\left({}^{W}\hat{\mathbf{p}}_{f_j}^\kappa - {}^{W}\mathbf{q}_j\right). \end{aligned} \tag{10}$$

It is important to note that we only consider residuals whose norm is below a certain threshold, while those above the threshold are treated as outliers or new observations. In order to fuse the state predictor $\hat{\mathbf{x}}_k$ and covariance $\hat{\mathbf{P}}_k$ obtained by IMU propagation with the residual $\mathbf{z}_j^\kappa$, it is necessary to linearize the observation equation. Since the state quantity of IESKF is numerically small, close to the origin, and far away from the singular point, its linearization approximation ability is good, allowing the second-order variables to be relatively neglected [33]. By performing a first-order Taylor expansion at $\hat{\mathbf{x}}_k^\kappa$, we approximate (3) as follows:

$$\begin{aligned} \mathbf{0} &= \mathbf{h}_j\left(\mathbf{x}_k, {}^{L}\mathbf{n}_j\right) \simeq \mathbf{h}_j\left(\hat{\mathbf{x}}_k^\kappa, 0\right) + \mathbf{H}_k^\kappa\tilde{\mathbf{x}}_k^\kappa + \mathbf{v}_j \\ &= \mathbf{z}_j^\kappa + \mathbf{H}_j^\kappa\tilde{\mathbf{x}}_k^\kappa + \mathbf{v}_j, \end{aligned} \tag{11}$$

where $\tilde{\mathbf{x}}_k^\kappa = \mathbf{x}_k \ominus \hat{\mathbf{x}}_k^\kappa$, $\mathbf{v}_j \in N\left(\mathbf{0}, \mathbf{R}_j\right)$ is due to the original measurement noise. $\mathbf{H}_j^\kappa$ is the Jacobian matrix of the first-order Taylor expansion of $\mathbf{h}_j\left(\mathbf{x}_k, {}^{L}\mathbf{n}_j\right)$ with respect to $\tilde{\mathbf{x}}_k^\kappa$ at $\mathbf{0}$. In this section, the superscript κ represents the state given by the Kalman filter, and without it is the state estimated by forward propagation.

d) Update iteratively: The forward propagation state $\hat{\mathbf{x}}_k$ and covariance $\hat{\mathbf{P}}_k$ impose a prior Gaussian distribution on the unknown state $\mathbf{x}_k$. According to (6), $\hat{\mathbf{P}}_k$ denotes the covariance of the following error states:

$$\begin{aligned} \mathbf{x}_k \ominus \hat{\mathbf{x}}_k &= \left(\hat{\mathbf{x}}_k^\kappa \oplus \tilde{\mathbf{x}}_k^\kappa\right) \ominus \hat{\mathbf{x}}_k \\ &= \hat{\mathbf{x}}_k^\kappa \ominus \hat{\mathbf{x}}_k + \mathbf{J}^\kappa\tilde{\mathbf{x}}_k^\kappa \sim N\left(\mathbf{0}, \hat{\mathbf{P}}_k\right), \end{aligned} \tag{12}$$

where $\mathbf{J}^\kappa$ is the partial differential matrix of $\left(\hat{\mathbf{x}}_k^\kappa \oplus \tilde{\mathbf{x}}_k^\kappa\right) \ominus \hat{\mathbf{x}}_k$ at $\tilde{\mathbf{x}}_k^\kappa = \mathbf{0}$:

$$\mathbf{J}^\kappa = \begin{bmatrix} \mathbf{A}\left(\varphi^W\boldsymbol{\theta}_{I_k}\right)^{-T} & \mathbf{0}_{3\times15} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{15\times3} & \mathbf{I}_{15\times15} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times15} & \mathbf{A}\left(\varphi^L\boldsymbol{\theta}_{I_k}\right)^{-T} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times15} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}. \tag{13}$$

In this expression, $\varphi^W\boldsymbol{\theta}_{I_k} = {}^{W}\hat{\mathbf{R}}_{I_k}^\kappa \ominus {}^{W}\hat{\mathbf{R}}_{I_k}$, $\varphi^L\boldsymbol{\theta}_{I_k} = {}^{L}\hat{\mathbf{R}}_{I_k}^\kappa \ominus {}^{L}\hat{\mathbf{R}}_{I_k}$. The definition of $\mathbf{A}$ is

$$\begin{aligned} \mathbf{A}\left(\mathbf{u}\right)^{-1} &= \mathbf{I} - \frac{1}{2}\left\lfloor\mathbf{u}\right\rfloor_\tau + \left(1 - \alpha\left(\|\mathbf{u}\|\right)\right)\frac{\left\lfloor\mathbf{u}\right\rfloor_\tau^2}{\|\mathbf{u}\|^2}, \\ \alpha(m) &= \frac{m}{2}\cot\left(\frac{m}{2}\right) = \frac{m}{2}\frac{\cos\left(m/2\right)}{\sin\left(m/2\right)}. \end{aligned} \tag{14}$$

For the first iteration, the estimate of the Kalman filter should be equal to the estimate of the forward propagation, which is $\hat{\mathbf{x}}_k^\kappa = \hat{\mathbf{x}}_k$, $\mathbf{J}^\kappa = \mathbf{I}$. In addition to the prior distribution, we have a state distribution due to the measurement equation (10):

$$-\mathbf{v}_j = \mathbf{z}_j^\kappa + \mathbf{H}_j^\kappa\tilde{\mathbf{x}}_k^\kappa \sim N\left(\mathbf{0}, \mathbf{R}_j\right). \tag{15}$$

Combining the prior distribution in (12) with (15) can obtain a posterior distribution for state $\mathbf{x}_k$, which we will equivalently represent as $\tilde{\mathbf{x}}_k^\kappa$ and its Maximum A Posteriori (MAP) estimate:

$$\min_{\tilde{\mathbf{x}}_k^\kappa}\left(\left\|\mathbf{x}_k \ominus \hat{\mathbf{x}}_k\right\|_{\hat{\mathbf{P}}_k}^2 + \sum_{j=1}^{m}\left\|\mathbf{z}_j^\kappa + \mathbf{H}_j^\kappa\tilde{\mathbf{x}}_k^\kappa\right\|_{\mathbf{R}_j}^2\right), \tag{16}$$

where $\|\mathbf{x}\|_{\mathbf{M}}^2 = \mathbf{x}^T\mathbf{M}^{-1}\mathbf{x}$. Substituting the linearization of the prior in (12) into (16) and optimizing the final quadratic cost, the standard form and iterative formula of Kalman gain can be obtained as follows:

$$\begin{aligned} \mathbf{K} &= \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1}, \\ \hat{\mathbf{x}}_k^{\kappa+1} &= \hat{\mathbf{x}}_k^\kappa \oplus \left(-\mathbf{K}\mathbf{z}_k^\kappa - \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\left(\mathbf{J}^\kappa\right)^{-1}\left(\hat{\mathbf{x}}_k^\kappa \ominus \hat{\mathbf{x}}_k\right)\right). \\ \mathbf{H} &= \left[\mathbf{H}_1^{\kappa^T}, \cdots, \mathbf{H}_m^{\kappa^T}\right]^T, \mathbf{R} = diag\left(\mathbf{R}_1, \cdots, \mathbf{R}_m\right), \\ \mathbf{z}_k^\kappa &= \left[\mathbf{z}_1^{\kappa^T}, \cdots, \mathbf{z}_m^{\kappa^T}\right]^T \end{aligned} \tag{17}$$

$\tilde{\mathbf{x}}$ will change after each iteration, and the covariance matrix is theoretically not the original $\mathbf{P}_k$, so we update the covariance through $\mathbf{P} = \left(\mathbf{J}^\kappa\right)^{-1}\hat{\mathbf{P}}_k\left(\mathbf{J}^\kappa\right)^{-T}$ during the iteration. In fact, it is close to the identity matrix. Iterate $\tilde{\mathbf{x}}_k^\kappa$ to minimize (16) and find an error value $\tilde{\mathbf{x}}$ and the corresponding covariance. Repeat the process until convergence: $\left\|\hat{\mathbf{x}}_k^{\kappa+1} \ominus \hat{\mathbf{x}}_k^\kappa\right\| < \epsilon$ (Since IESKF is proven to be equivalent to Gauss-Newton method, global convergence is guaranteed [33]). After convergence, the optimal state and covariance estimates are: $\bar{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{\kappa+1}$, $\bar{\mathbf{P}}_k = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{P}$.

The above form requires the inversion of the matrix $\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)$ in the measurement dimension (its dimension is $m \times m$, because of $\mathbf{H}_{m\times21}, \mathbf{P}_{21\times21}, \mathbf{R}_{m\times m}$), and the number of LiDAR point clouds (m) is huge. The operation of inversion will bring great computational load and time consumption, which is not allowed in real-world tasks. Therefore, the existing works [31], [34] only use a small number of measurement

points. However, Shan et al. [27] believe that the cost function is built on the state quantity, so the computational complexity should depend on the size of the state dimension. Based on this, we transform the standard Kalman gain:

$$\begin{aligned}\mathbf{K} &= \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)^{-1}\\ &= \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}-\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}+\mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)^{-1}\\ &= \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}-\mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)^{-1}\left(\mathbf{H}\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\right)\end{aligned} \tag{18}$$

Notice that $\mathbf{H}\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} = \left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)\mathbf{R}^{-1}-\mathbf{I}$, then the above equation can be simplified as $\mathbf{H}^T\mathbf{R}^{-1}\left[\mathbf{P}-\mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)^{-1}\mathbf{H}\mathbf{P}\right]$. According to the Sherman-Morrison-Woodbury formula [35]: $\mathbf{P}-\mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right)^{-1}\mathbf{H}\mathbf{P} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}+\mathbf{P}^{-1}\right)^{-1}$, then $\mathbf{K}$ can be rewritten as follows:

$$\mathbf{K} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}+\mathbf{P}^{-1}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}. \tag{19}$$

The new formulation takes advantage of the independence of LiDAR measurements and the diagonal property of the covariance matrix $\mathbf{R}$. This allows for a more efficient computation as it only requires the inverse of the two matrices in the state dimension, rather than the measurement dimension. Since the state dimension (e.g., 21 dimensions in this case) is typically much lower than the number of measurements (e.g., more than 10,000 valid points at a 10 Hz scanning rate), significant computational time is saved. This optimization greatly improves the computational efficiency of the LIO framework, making it more suitable for real-time applications and large-scale SLAM tasks. Finally, by updating state $\bar{\mathbf{x}}_k$, the relative transformation $\Delta\mathbf{T}$ between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ can be obtained, which is the odometry factor connecting these two poses. Using this relative transformation as an initial value, we match the current frame with the global map (global matching is only performed on keyframes. ${}^L\mathbf{p}_j$ is converted to world frame and inserted into the map represented by the ikd-Tree:

$${}^W\bar{\mathbf{p}}_{f_j} = {}^W\bar{\mathbf{T}}_{I_k}{}^L\bar{\mathbf{T}}_{I_k}{}^{L_k}\mathbf{p}_{f_j},\ j = l,\ \cdots,\ m. \tag{20}$$

C. GPS Factor

To address the prevalent issue of GPS signal interruptions in urban autonomous driving scenarios, this study proposes enhancements to the existing STEKF. The modifications aim to facilitate a rapid return to system stability for GPS signal restoration. The fundamental concept of STEKF involves employing a time-varying factor to gradually attenuate past data, thereby diminishing the influence of outdated data on the current filtering predictions. This can be accomplished by altering the covariance matrix and subsequently adjusting the gain matrix. The specific approach involves incorporating a time factor into the conventional EKF model [36]:

$$\mathbf{P}_{k+1|k} = \omega_{k+1}\mathbf{J}_{k+1,k}\mathbf{P}_k\mathbf{J}_{k+1,k}^T+\mathbf{Q}_k. \tag{21}$$

$\mathbf{Q}_k$ represents the noise covariance matrix of the state model. While STEKF exhibits a rapid response capability to sudden changes, this comes at the expense of sacrificing a certain level of filtering accuracy. This is due to the fact that the standard EKF is designed to determine the optimal gain matrix based on fulfilling the minimum variance criterion, as showed in (16). In most autonomous driving tasks, choosing to sacrifice overall navigation accuracy in exchange for rapid local response appears to be unjustifiable. Thus, we propose adjusting the fixed time factor to support the seamless switch between GNSS-affluent regions and GNSS-denied regions. In the following algorithm, this study will determine the decay time factor by comparing the relationship between the residual estimate of the system and the conditional expectation. The ratio of the current residual's deviation from the expected residual is:

$$\begin{aligned}\gamma_0 &= \left(\mathbf{Z}_{k+1}^{*T}\mathbf{Z}_{k+1}^{*}\right)/\left(\mathbf{tr}E\left(\mathbf{Z}_{k+1}^{*}\mathbf{Z}_{k+1}^{*T}\right)\right),\\ \mathbf{Z}_{k+1}^{*} &= \mathbf{Z}_{k+1}-\mathbf{H}_{k+1}\hat{\mathbf{X}}_{k+1|k}\\ &= \mathbf{H}_{k+1}\left(\mathbf{X}_{k+1}-\hat{\mathbf{X}}_{k+1|k}\right)+\mathbf{V}_{k+1}\\ &= \mathbf{H}_{k+1}\mathbf{X}_{k+1}^{*}+\mathbf{V}_{k+1},\end{aligned} \tag{22}$$

where $E\left(\mathbf{Z}_{k+1}^{*}\mathbf{Z}_{k+1}^{*T}\right)$ represents the expectation of the residual, and $\mathbf{V}_{k+1}$ is the independent Zero-mean white noise sequences. Notice that $E\left(\mathbf{X}_{k+1}^{*}\mathbf{V}_{k+1}^{T}\right) = E\left(\mathbf{V}_{k+1}\mathbf{X}_{k+1}^{*\,T}\right) = 0$, $E\left(\mathbf{X}_{k+1}^{*}\mathbf{X}_{k+1}^{*\,T}\right) = \mathbf{P}_{k+1|k}$, $E\left(\mathbf{V}_{k+1}\mathbf{V}_{k+1}^{T}\right) = \mathbf{R}_{k+1}$, $\mathbf{R}_k$ represents the noise covariance matrix of the measurement model, thus

$$\begin{aligned}E\left(\mathbf{Z}_{k+1}^{*}\mathbf{Z}_{k+1}^{*\,T}\right) &= \mathbf{H}_{k+1}\left(\mathbf{J}_{k+1,k}\mathbf{P}_k\mathbf{J}_{k+1,k}^T\right)\\ &\quad+\mathbf{H}_{k+1}\mathbf{Q}_k\mathbf{H}_{k+1}^T+\mathbf{R}_{k+1}.\end{aligned} \tag{23}$$

After the calculation of γ_0, let $\gamma_1 = \max(\gamma_{min}, \gamma_0)$, and set 1 as the critical residual parameter of γ_{min}. When $\gamma_1 \le \gamma_{min}$, we regard the residual is small and GPS data is not interrupted. Based on these inference, let it be the subjection function:

$$g(\gamma_1) = \omega_{k+1}\exp\left(-\frac{\gamma_{min}\ln\omega_{k+1}}{\gamma_1}\right), \tag{24}$$

the original covariance formula (21) becomes:

$$\mathbf{P}_{k+1|k} = g(\gamma_1)\mathbf{J}_{k+1,k}\mathbf{P}_k\mathbf{J}_{k+1,k}^T+\mathbf{Q}_k. \tag{25}$$

Above is the outlined improvement scheme, which we refer to as the fundamental form of fuzzy STEKF. Intuitively, when $\gamma_0 \le \gamma_{min}$, it can be considered that the estimated residual is within an acceptable range. This implies that there has been no abrupt change in GPS measurements at this moment ($g(\gamma_1) = \omega_{k+1}\exp\left(-\frac{\gamma_{min}\ln\omega_{k+1}}{\gamma_{min}}\right) = 1$). When the computed residual is significantly larger than the expected residual, indicating a substantial change in measurements, at this point, $\gamma_1 \to +\infty$, $g(\gamma_1) \to \omega_{k+1}$. The filter demonstrates strong tracking characteristics.

The advantage of our algorithm lies in leveraging the subjection function from fuzzy logic, allowing for a synergistic integration of both EKF and STEKF. In most cases, when the system is stable and GPS measurements are normal, EKF is predominantly utilized to ensure filtering accuracy. Conversely, in the event of GPS signal failure, STEKF is employed to a

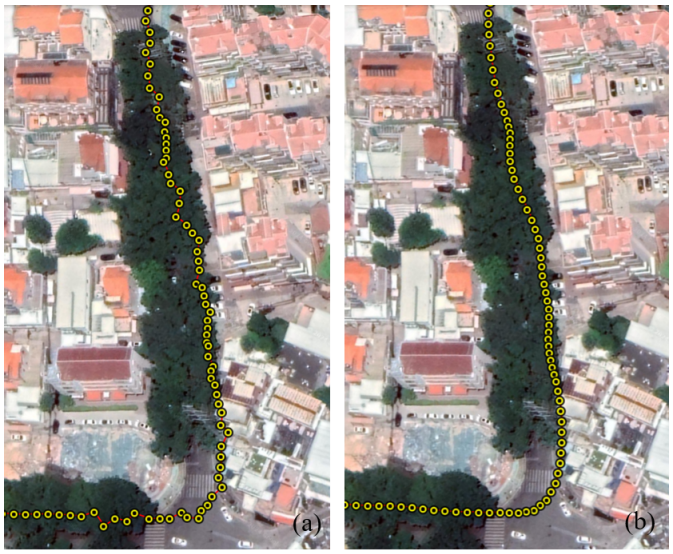


Fig. 3. Comparison of the filtered GPS. (a) Before GPS data filtering. (b) After GPS data filtering.

greater extent. In order to prove the efficacy of the proposed Fuzzy STEKF in providing accurate estimations of systems, we conducted a mathematical proof (see Appendix D).

Subsequently, we conducted tests in real-world scenarios, particularly focusing on the urban areas occluded by dense buildings and trees, where fluctuations were observed in GPS data. Fig. 3 illustrates the GPS trajectories before and after the application of our filtering method. It is evident that prior to filtering, there was significant fluctuation, whereas the fluctuations decreased after the implementation of Fuzzy STEKF. In practice, the Kalman filter updates the state vector and covariance matrix based on pseudorange residuals to refine the position estimate, and similarly updates the state vector and covariance matrix based on Doppler residuals to enhance the accuracy of the velocity estimate.

D. Loop Closure Factor

Due to the construction of factor graphs, loop detection module can be smoothly integrated into our system. This paper describes and implements a loop closure detection method based on Scan context [37]. Subsequently, loop observations need to be added to the factor graph framework in the form of global constraints. When a new vehicle state $\mathbf{x}_{k+1}$ is added to the factor graph, we first search the graph to find the prior state that is closest to $\mathbf{x}_{k+1}$ in the space of descriptors. For example, as shown in Fig. 1, $\mathbf{x}_4$ using ICP [13] and obtain the relative transformation $\Delta\mathbf{T}_{4,k+1}$. It is finally added to the graph as a loop closure factor. Note that $\mathbf{F}_{k+1}$ is converted to the world frame before scan matching. In order to avoid the problem of excessive calculation caused by frequent queries, we set the frequency of loop detection to 4 Hz and the minimum interval time to 30 seconds to filter out close matches, allocating a separate thread for it.

The original Scan Context determines and excludes similar but distant point clouds by setting a fixed threshold. However, when the threshold is set too high, it may lead to numerous false loop closure detection, resulting in some deflected edges during graph optimization. To address this issue, we introduce an adaptive distance threshold to replace the conventional threshold. When the system obtains loop closure detection results, we use the poses ${}^{W}\mathbf{T}_i$ corresponding to the loop closure frame and ${}^{W}\mathbf{T}_{loop}$ to calculate the distance between two points:

$$
\begin{aligned}
{}^{loop}\mathbf{T}_i &= {}^{W}\mathbf{T}_{loop}^{-1}\,{}^{W}\mathbf{T}_i, \\
d &= \sqrt{{}^{loop}\mathbf{T}_i \cdot x^2 + {}^{loop}\mathbf{T}_i \cdot y^2 + {}^{loop}\mathbf{T}_i \cdot z^2},
\end{aligned} \tag{26}
$$

where ${}^{loop}\mathbf{T}_i$ represents the pose transformation between the current pose ${}^{W}\mathbf{T}_i$ obtained from odometry and the loop closure pose ${}^{W}\mathbf{T}_{loop}$. The components ${}^{loop}\mathbf{T}_i \cdot x$, ${}^{loop}\mathbf{T}_i \cdot y$, ${}^{loop}\mathbf{T}_i \cdot z$ denote the vector connecting the current frame and the loop closure frame in the x, y, and z directions, and d represents the magnitude of this vector. If d exceeds the threshold d_{th}, it is considered that the point cloud at that location does not form a loop.

The cumulative error of the system increases with the unmanned vehicle operation. Therefore, d_{th} is designed as a function associated with the odometry keyframes: $d_{th} = 15 + k/n$. Here, k represents the number of keyframes, and n is a parameter designed based on the cumulative error from odometry. 15 is an experimentally determined optimal threshold. The use of d_{th} can reduce false loop closure detections, consequently improving the results of factor graph optimization and the overall efficiency of the SLAM algorithm.

In the tests, we found that adding loop closure factors has great potential in correcting height errors, which is particularly important because the absolute elevation of GPS is imprecise and even jumps in numerical values. However, in some complex road scenarios, a large number of similar objects are inevitable, such as pedestrians, buildings, traffic signs, trees, etc., which increases the difficulty of judging loop candidates. We suggest readers to replace this module with a more advanced loop closure detection system.

IV. Experiments

The hardware device for this experiment is a Dell G15 5511 laptop computer with 11th Gen Intel(R) Core(TM) i7-11800H (8 cores, 16 threads, 2.30 GHz) processor. The experiment uses Ubuntu18.04 based on Linux kernel as the running environment of SLAM, and is built on ROS Melodic. It relies on several open-source library files, including Eigen3.3.7, Ceres-solver1.13.0, PCL1.8 and Gtsam4.0.0, to handle various mathematical and optimization tasks. Due to the heavy CPU parallel computing pressure, the performance of algorithms is significantly affected by the computer's CPU. During the experiments, we set the number of threads to the maximum that CPU can hold.

The input datasets used in this experiment are from AVIA [9], LIO-SAM [27], KITTI [38], and UTBM robocar [39], which are represented by lowercase letters in figures and tables. $AVIA$ dataset is designed to support Faster-LIO [9] system, and contains solid-state LiDAR point clouds. Specifically, the $avia\_3$ sequence provides high-frequency IMU data at 100 Hz, which represents extreme motion

scenarios. $LIO\text{-}SAM$ dataset is provided by Shan et al. [27] and serves to evaluate the robustness of the algorithm. It covers point cloud data in different formats acquired by multi-line rotating Velodyne LiDAR and solid-state Livox Horizon LiDAR. $UTBM$ robocar dataset is collected by cars driving at medium or low speeds, and it integrates different weather, environment, and seasonal changes, which is crucial for the long-term stability of autonomous driving. Due to the similarities of some sequences in $UTBM$, we selected only a subset of them. Finally, we evaluate the performance of the algorithm on the $KITTI$ odometry dataset, which is characterized by long distance and highly dynamic road scenarios. These datasets provide information from IMU, LiDAR and have good ground-truth trajectories, often collected using RTK techniques. To evaluate the robustness of FAST-LiDAR-SLAM when GPS is suddenly interrupted, we manually disabled the GPS signal at a certain frequency (see IV.B).

A. Mapping Results

In order to verify the mapping effectiveness and robustness of FAST-LiDAR-SLAM with different hardware configurations, we conducted tests on the $AVIA$ dataset (solid-state LiDAR) and $liosam\_2$ sequence (mechanical rotary LiDAR). They contain complex trajectories and challenging road environments, which presents obstacles to build a 3D dense map. As shown in Fig. 4 and Fig. 5, FAST-LiDAR-SLAM demonstrates excellent performance in reconstructing detailed environments although different point cloud formats are input. Its ability to achieve intensive mapping results is attributed to the feature-free approach, enabling natural adaptation to multiple types of LiDARs.

To further investigate the applicability of FAST-LiDAR-SLAM in terms of the emerging solid-state LiDAR, we compared it with LIO-SAM [27] and LiLi-OM [8] on the $AVIA$ dataset (see Fig. 6 and Fig. 7). LIO-SAM [27] is known for its dense mapping capabilities. LiLi-OM [8] designs a novel feature extraction method and lightweight front-end based on irregular scanning LiDAR. In this experiment, the unmanned platform starts from the left top and makes a counterclockwise circle around the industrial park at an average speed of 1.54 m/s and then returns to the starting point. The driving track is a closed loop with a total distance of 540 m. Testing roads are mostly level with occasional bumps and slopes less than 4°. It is equipped with the Livox Avia solid-state LiDAR, which has a built-in IMU, a circular FoV, and a non-repetitive scanning pattern.

Fig. 6 corresponds to the top view of maps generated by different algorithm. It can be found that the points of LIO-SAM are diverse and chaotic, and the map has certain offset and ghosting caused by irregular scanning. As a result, the road contours are unclear, making it unable to apply to autonomous driving tasks. LiLi-OM [8] algorithm has higher mapping efficiency than LIO-SAM [27], but it has problems such as rough clustering segmentation, incorrect deletion of moving objects, sparse valid points and unclear reconstruction contours. These limitations arise from the fact that LiLi-OM's feature extraction module is designed for the specific scanning pattern (Livox Horizon), leading to a weak generalization ability. In contrast, our work shows several advantages over both LIO-SAM [27] and LiLi-OM [8]. The 3D reconstruction result is clear, and there are no noticeable ghosting or offset phenomena.

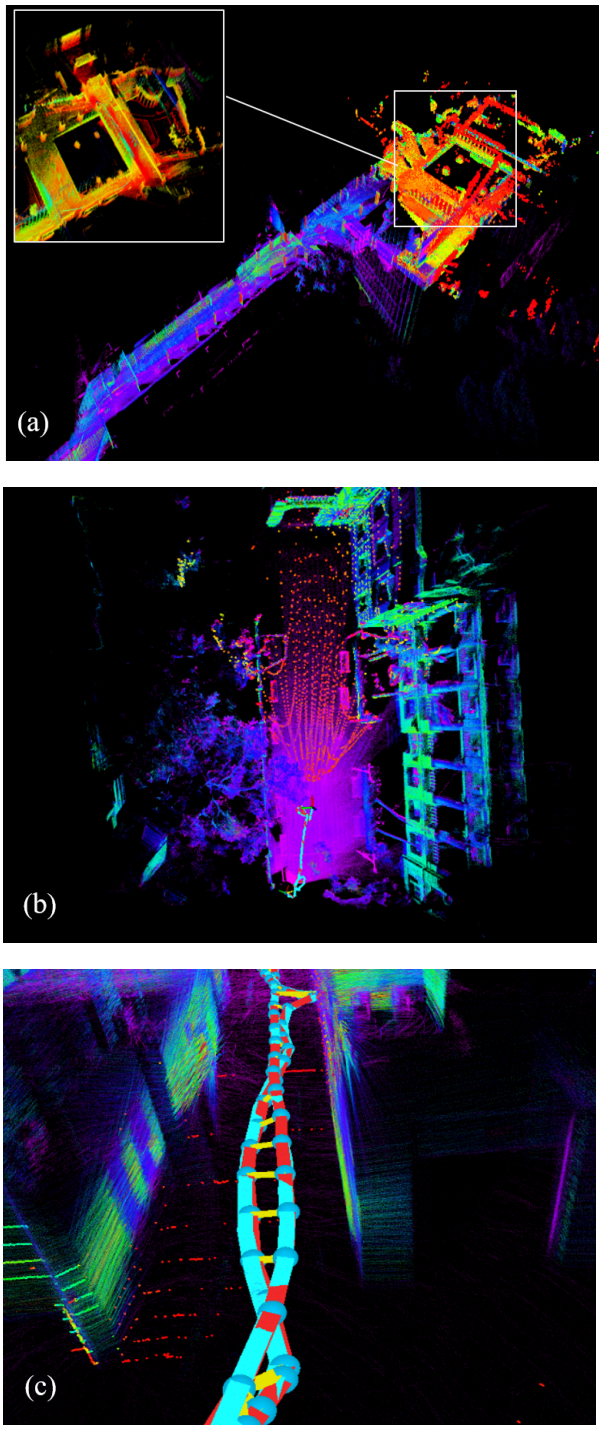


Fig. 4. Dense mapping result on $AVIA$ dataset. (a) Dense mapping result on $avia\_1$ sequence. (b) Fine restoration of buildings and trees. (c) Visualization of loop detection.

In Fig. 7, it is found that a certain ghosting arises in the Z-axis direction of LIO-SAM, which is caused by the repeated features extracted between multiple LiDAR frames. On the other hand, LiLi-OM algorithm simplifies the feature extraction process. The lack of global constraints leads

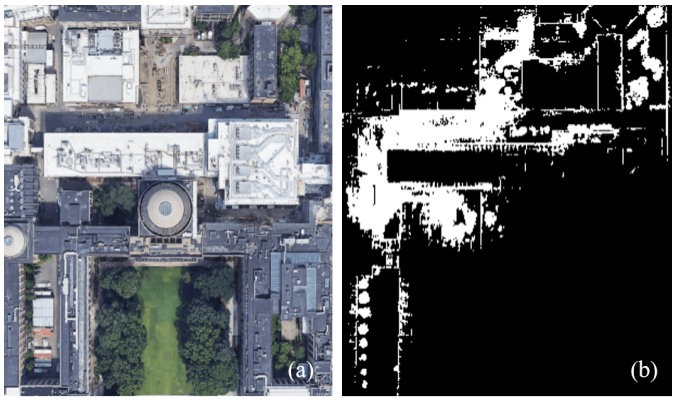


Fig. 5. Mapping result on *liosam_2* sequence. (a) Real environment. (b) Mapping result of FAST-LiDAR-SLAM. Our work basically reconstructs the correct road structure and important obstacles in the environment, which is consistent with the Google Earth map provided in [28]. Unlike the dense reconstruction achieved by LIO-SAM, the map generated by FAST-LiDAR-SLAM is sparser, which is attributed to the lightweight front-end. The number of valid points in the map is 5598804 (about half of that generated by LIO-SAM [27]).

to poor mapping results, sketchy structure restoration, and obvious drift of the map. In contrast, FAST-LiDAR-SLAM algorithm directly registers the original LiDAR points to the map and utilizes the subtle features in the environment, leading to improved mapping accuracy. From the perspective of 3D reconstruction, the method of directly registering point clouds suppresses the elevation error and point cloud ghosting, demonstrating robust mapping performance in complex scenes.

In addition, we conduct an ablation test for FAST-LiDAR-SLAM, which is designed to show the benefits of introducing GPS and loop closure factors. To do this, we deliberately forbid the insertion of GPS and loop closure factors in the trajectory graph. When both GPS and loop closure factors are disabled, our method is called $odometry\_only$, which utilizes only the tightly coupled odometry of IMU and LiDAR. When loop closure factor is used, our method is called $odometry + loop$, which uses the odometry factor and loop closure factor for mapping. FAST-LiDAR-SLAM uses all available factors. $Groundtruth$ represents the reference truth obtained from the full RTK measurements. As shown in the Fig. 8, the trajectory drift of $odometry\_only$ is obvious. In the case of $odometry\_only$, the overall trajectory error converges to the true value, but there is a small drift on the right side. Therefore, adding the loop observation to the graph can significantly reduce the cumulative error of the odometry, especially over a long time. When all factors are available, its estimated trajectory can be well aligned with the true trajectory, with only an absolute error of 0.83 m over 540 meters test.

B. Accuracy

1) GPS Uninterrupted Experiment: This section compares FAST-LiDAR-SLAM with other SOTA SLAM systems, including Faster-LIO [9], FAST-LIO [10], LIO-SAM [27], and LiLi-OM [8]. Absolute Trajectory Error (ATE) was used as the accuracy index for the whole trajectory, and the systematic cumulative drift assessment was performed by calculating the

Fig. 6. The top view of 3D mapping results.

(translational) Relative Pose Error (RPE) for every 100 meters. The results in Table II show that FAST-LiDAR-SLAM outperforms the other SLAM systems in terms of ATE and RPE. The bold numbers indicate the best performance on both accuracy indices. The parameter Settings of LIO-SAM and LiLi-OM were not adjusted in this experiment, as this may cause the old algorithm not working perfectly in the new dataset. Since LIO-SAM only accepts 9-axis IMU measurements as input, it was not tested on the $UTBM$ dataset, denoted as -. Similarly, the $LIO\text{-}SAM$ dataset does not produce the pose format required by LiLi-OM and is denoted as -.

The evaluation on the $AVIA$ dataset and the long-term $UTBM$ dataset demonstrates our work's strong performance. The algorithm achieves only 0.22% trajectory deviation and

TABLE II
ATE AND RPE RESULTS OF FAST-LiDAR-SLAM, FAST-LIO, FASTER-LIO, LIO-SAM, AND LiLi-OM USING DIFFERENT DATASETS

Dataset	Ours		FAST-LIO		Faster-LIO		LIO-SAM		LiLi-OM		Time	Distance
	ATE(m)	RPE(%)	ATE(m)	RPE(%)	ATE(m)	RPE(%)	ATE(m)	RPE(%)	ATE(m)	RPE(%)	(s)	(km)
$avia\_1$	**0.83**	**0.87**	1.24	1.31	1.15	1.17	2.73	2.57	5.23	2.46	351	0.54
$avia\_2$	**0.74**	**0.22**	0.88	0.35	0.81	0.36	1.17	0.79	2.78	0.77	260	0.10
$avia\_3$	**1.63**	1.40	2.43	1.97	1.76	**1.33**	2.57	2.04	4.42	2.34	49	0.01
$avia\_ave$	**1.07**	**0.83**	1.52	1.21	1.24	0.95	2.16	1.80	4.14	1.86	220	0.22
$utbm\_1$	**11.90**	**0.51**	14.48	0.71	12.73	0.66	-	-	63.18	2.26	1019	5.03
$utbm\_2$	**12.29**	**0.57**	13.37	0.84	15.13	0.84	-	-	82.07	1.20	959	4.99
$utbm\_3$	**13.32**	**0.79**	14.60	1.18	14.84	1.01	-	-	102.32	6.72	959	4.99
$utbm\_4$	**6.75**	**1.34**	7.77	1.80	7.22	1.54	-	-	48.01	2.32	999	5.00
$utbm\_ave$	**11.07**	**0.82**	12.56	1.13	12.48	1.01	-	-	73.90	3.13	984	5.00
$liosam\_1$	4.21	**0.59**	4.98	0.99	4.83	1.06	**3.83**	0.61	-	-	986	1.44
$liosam\_2$	**3.34**	**0.49**	3.71	0.73	4.46	0.90	3.40	0.52	-	-	655	0.81
$liosam\_3$	2.69	0.40	2.97	0.71	3.73	0.83	**2.61**	**0.37**	-	-	358	0.46
$liosam\_4$	**1.85**	0.78	2.22	1.18	3.34	1.62	1.87	**0.71**	-	-	551	0.66
$liosam\_ave$	3.02	0.57	3.47	0.90	4.09	1.10	**2.93**	**0.55**	-	-	638	0.84

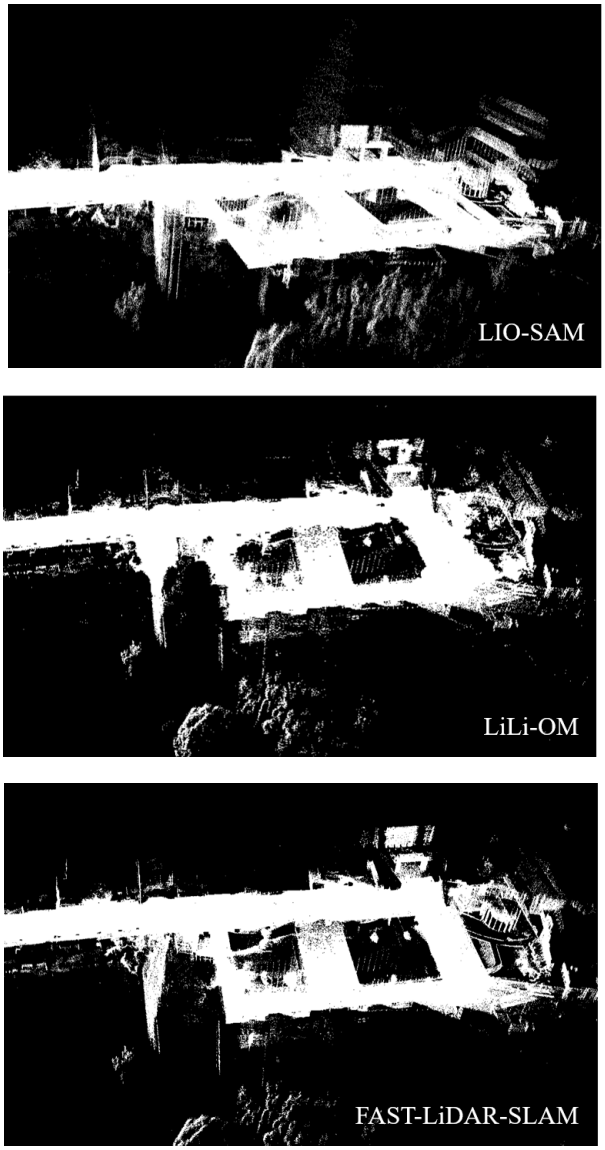


Fig. 7. The main view of 3D mapping results.

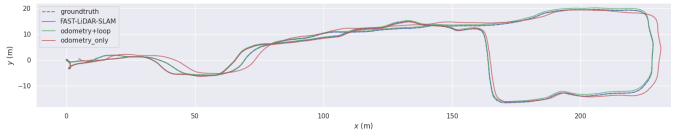


Fig. 8. Trajectory comparison on $avia\_1$ sequence.

0.85% odometry horizontal drift under long-distance operation, achieving centimeter-level accuracy. In comparison to Faster-LIO [9], the average APE of FAST-LiDAR-SLAM is reduced by 13.71%, and the average RPE is reduced by 12.63%. As the map size increases, the overall accuracy of FAST-LiDAR-SLAM is improved because the factor graph model effectively handles revisits to old sites by obtaining the registration of the new scan with the old historical points, providing pose priors and opportunities to speed up operations. However, the accuracy increment becomes difficult to maintain linear growth, as mismatches with original map points occur due to error accumulation. It is worth noting that LiLi-OM [8] exhibits a very large drift on the $UTBM$ dataset, which is due to the failure of the sliding window optimization thread as the number of LiDAR frames increase sharply.

Fig. 9 presents a comparison of the mapping trajectories of Faster-LIO [9], LIO-SAM [27], LiLi-OM [8], and our work ($optimized\_pose$) on $avia\_1$ sequence. The black dotted line in the figure represents the true GNSS trajectory. The results indicate that $optimized\_pose$ is the closest to the true trajectory. Faster-LIO [9] is an incremental voxel-based LIO method known for its SOTA computational performance. However, due to the lack of global constraints, drift occurs in scenes with rapid changes in LiDAR viewpoints, as observed in the square enlarged area in Fig. 9 (a). LIO-SAM faces challenges in adapting to the scanning pattern of solid-state LiDAR (with a narrower FoV), leading to inferior trajectory accuracy compared to Faster-LIO. Since LiLi-OM strictly follows the specific LiDAR model, it requires the autonomous platform moving at a low speed to achieve complete convergence of the point cloud registration. If the platform moves at high speeds, it may result in a large trajectory drift.

Based on Table II and Table V, FAST-LiDAR-SLAM achieves comparable accuracy (typically 0.5-1.0% translation drift and less than 0.5% trajectory error) to LIO-SAM [27] at a faster rate. Considering that the mechanical rotation LiDAR

TABLE III
RELATIVE POSE ERROR OF FAST-LIDAR-SLAM, LOAM, SA-LOAM, FASTER-LIO, LIO-SAM, AND SUMA USING *KITTI* DATASET

Seq	Ours		LOAM*		SA-LOAM*		Faster-LIO		LIO-SAM*		SUMA*		Time	Distance
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	(s)	(km)
00	1.13	0.85	0.78	0.53	**0.59**	0.25	1.47	0.72	0.78	0.52	0.68	**0.23**	471	3.72
01	1.08	0.59	1.43	0.55	1.89	0.48	1.36	0.47	**0.67**	**0.23**	1.70	0.54	114	2.45
02	**0.74**	**0.25**	0.92	0.55	0.77	0.28	1.52	0.71	0.86	0.61	1.20	0.48	483	5.07
03	0.94	**0.44**	0.86	0.65	0.87	0.46	1.03	0.66	1.76	**0.44**	**0.74**	0.50	84	0.56
04	1.23	0.89	0.71	0.50	0.59	0.35	0.51	0.65	**0.37**	0.50	0.44	**0.27**	28	0.39
05	0.67	0.68	0.57	0.38	0.45	0.24	1.04	0.69	0.65	0.67	**0.43**	**0.20**	288	2.21
06	0.38	**0.24**	0.65	0.39	0.52	0.25	0.71	0.50	**0.27**	0.52	0.54	0.30	114	1.23
07	0.59	0.48	0.63	0.50	**0.41**	**0.22**	1.70	0.89	0.80	0.45	0.74	0.54	114	0.69
08	1.08	0.52	1.12	0.44	**0.85**	**0.27**	2.12	0.77	1.32	0.55	1.20	0.38	423	3.22
09	**0.57**	0.31	0.77	0.48	0.68	0.28	1.37	0.58	0.79	0.76	0.62	**0.22**	165	1.71
10	0.73	0.49	0.79	0.57	0.78	0.35	1.80	0.93	0.98	0.62	**0.72**	**0.32**	124	0.92
Average	0.83	0.52	0.84	0.50	**0.76**	**0.31**	1.33	0.69	0.84	0.53	0.82	0.36	219	2.02

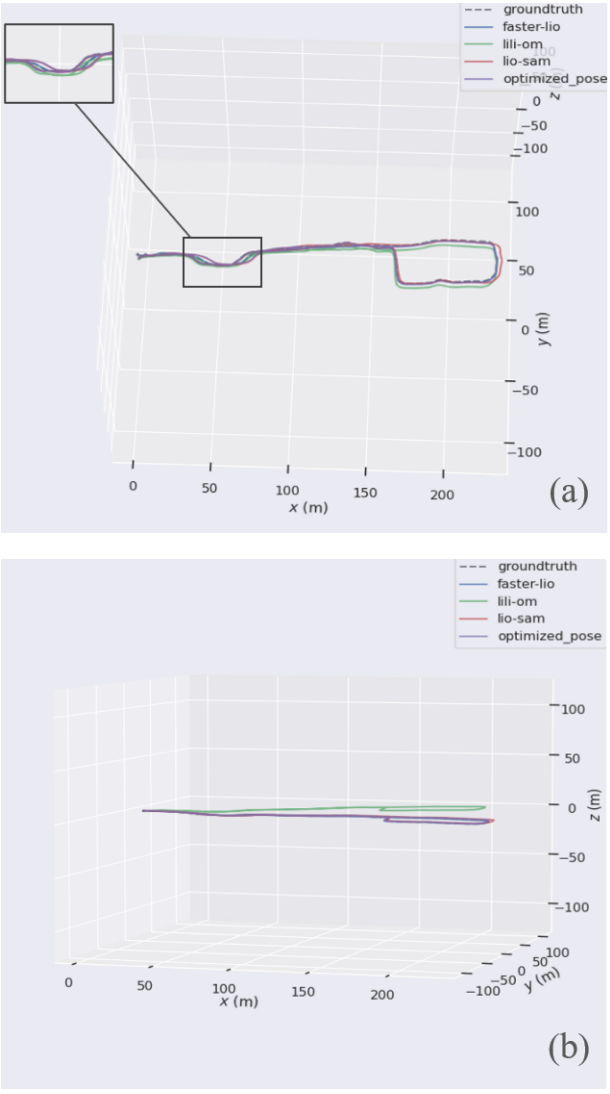


Fig. 9. Trajectories of different algorithms. (a) The main view of odometry trajectories. (b) The Z-axis view of odometry trajectories.

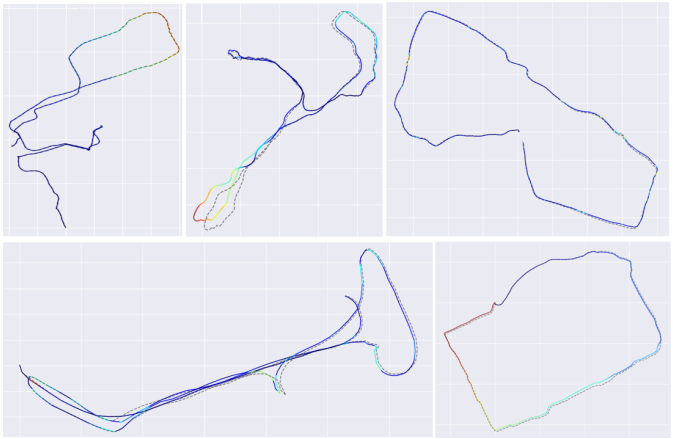


Fig. 10. True and estimated trajectory of *AVIA* and *LIO-SAM* datasets: dotted lines represent the reference truth trajectories obtained from RTK, and colored solid lines represent the estimated results. The colors in the figure indicate the error of the estimated trajectory: red is large, blue is small.

TABLE IV
RELATIVE POSE ERROR OF FAST-LIDAR-SLAM, LIO-SAM, AND OL-SLAM USING *KITTI* DATASET FOR GPS INTERRUPTED EXPERIMENTS

Seq	Ours		LIO-SAM		OL-SLAM	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	1.14	0.85	**0.89**	0.53	0.97	**0.48**
01	1.10	**0.59**	**0.98**	0.66	0.99	0.64
02	**0.74**	**0.26**	0.93	0.61	1.24	0.75
03	**0.94**	**0.44**	1.76	0.44	1.53	0.53
04	1.23	0.89	**0.37**	0.50	0.79	**0.46**
05	**0.69**	0.68	0.69	**0.67**	0.81	0.70
06	**0.40**	**0.24**	0.59	0.53	0.64	0.43
07	**0.63**	0.48	0.98	**0.45**	0.96	0.50
08	**1.08**	**0.52**	1.41	0.55	1.04	0.61
09	**0.58**	**0.31**	1.29	0.76	0.97	0.59
10	**0.76**	0.49	1.45	0.64	0.79	**0.46**
Average	**0.84**	**0.52**	1.03	0.58	0.96	0.56

dataset is extracted directly from *LIO-SAM*, the algorithm is already well-tuned for the data. FAST-LiDAR-SLAM does not adjust the initial parameter Settings when testing 11 sequences, which shows the robustness to deal with different scenes and hardware configurations. Moreover, in most sequences, the performance of FAST-LIO [10], which is based on line-surface features, is similar to the directly Faster-LIO method.

Furthermore, we compare FAST-LiDAR-SLAM with the advanced SLAM systems on the *KITTI* dataset. These results

TABLE V
CUMULATIVE ERRORS OF FAST-LiDAR-SLAM, LIO-SAM, AND OL-SLAM USING $avia\_2$ SEQUENCE

Scheme	Horizontal			Vertical			Global pose error
	x/(m)	y/(m)	yaw/(°)	z/(m)	roll/(°)	pitch/(°)	(m)
OL-SLAM	0.105	0.122	1.230	**0.101**	1.291	1.206	0.190
LIO-SAM	0.421	0.459	5.977	0.565	6.142	6.004	0.841
Ours	**0.077**	**0.085**	**0.921**	0.103	**0.908**	**1.013**	**0.154**

TABLE VI
SINGLE-FRAME PROCESSING TIME OF FAST-LiDAR-SLAM, FAST-LIO, FASTER-LIO, LIO-SAM, AND LiLi-OM USING DIFFERENT DATASETS

Dataset	Ours		FAST-LIO		Faster-LIO		LIO-SAM		LiLi-OM	
	PRE(ms)	OPT(ms)	PRE(ms)	OPT(ms)	PRE(ms)	OPT(ms)	PRE(ms)	OPT(ms)	PRE(ms)	OPT(ms)
$avia\_1$	**1.48**	15.57	2.95	6.98	1.53	**5.58**	4.91	43.74	6.93	7.98
$avia\_2$	2.31	15.09	2.03	6.21	**1.26**	**3.94**	3.39	34.91	5.20	6.75
$avia\_3$	**0.64**	13.83	1.74	5.87	0.72	**2.21**	2.07	34.47	4.02	7.49
$avia\_ave$	1.81	14.83	2.24	6.35	**1.50**	**3.91**	3.46	37.71	5.38	7.41
$utbm\_1$	3.01	15.04	7.31	9.75	**2.08**	**5.47**	-	-	13.07	17.12
$utbm\_2$	**3.74**	15.57	7.93	9.98	3.90	**5.45**	-	-	15.66	17.79
$utbm\_3$	4.12	15.01	7.28	10.17	**3.90**	**5.54**	-	-	15.03	18.43
$utbm\_4$	**4.08**	15.92	7.60	10.93	4.13	**6.06**	-	-	13.96	17.43
$utbm\_ave$	3.74	15.39	7.53	10.21	**3.50**	**5.63**	-	-	14.43	17.69
$liosam\_1$	**2.09**	14.77	2.59	6.53	2.32	**5.15**	4.80	43.67	-	-
$liosam\_2$	2.13	14.37	4.85	10.11	**1.97**	**4.04**	3.82	36.91	-	-
$liosam\_3$	**1.72**	14.66	2.39	6.44	1.98	**5.10**	3.63	35.04	-	-
$liosam\_4$	**1.95**	15.14	2.58	6.82	2.04	**4.96**	3.84	36.88	-	-
$liosam\_ave$	**1.97**	14.74	3.10	7.48	2.08	**4.81**	4.02	38.13	-	-
Average	2.47	15.27	4.47	7.90	**2.34**	**4.86**	-	-	-	-

are summarized in Table III. Most of the data were extracted from their original papers (marked with *). Bold numbers indicate the best performance in terms of translation error and rotation error. t_{rel} denotes the mean translation RMS error (%) and r_{rel} denotes the mean rotation RMS error (°/100 m). It can be found that semantically augmented LiDAR SLAM such as SUMA [40], SA-LOAM [41], obtain more accurate results. However, they usually rely on supervised training and senseless learning, which brings expensive cost to the real-world deployment. In contrast, the lightweight and efficient nature of FAST-LiDAR-SLAM makes it a viable option for achieving comparable accuracy, which is suitable for practical applications in large-scale autonomous driving. In addition, it was observed that FAST-LiDAR-SLAM achieved similar accuracy as LOAM [15] and LIO-SAM [27], indicating that our work is well compatible with traditional rotating LiDAR and meets the accuracy requirements at the centimeter level.

2) GPS Interrupted Experiment: In this section, we compare FAST-LiDAR-SLAM with LIO-SAM [27] and OL-SLAM [42] on the $KITTI$ dataset. To simulate the sudden GPS interruption in the real world, we manually interrupt the GPS signals for approximately 5 seconds. Specifically, interruptions starting from the 40th and 80th second were applied to the 01, 05, 06, 07, 09 and 10 sequences. Interruptions starting from the 100th, 200th, 300th and 400th were applied to the 00, 02 and 08 sequences. The test results are summarized in Table IV. For the GPS interrupted experiments, FAST-LiDAR-SLAM has a slightly higher minimum ATE than LIO-SAM, but it outperforms LIO-SAM and OL-SLAM in other statistical indicators, with decrements of 18.45% and 12.5% RPE respectively. In terms of mean ATE value, the proposed algorithm achieves 1.39 m on testing sequences, which is only 78.09% of LIO-SAM.

Furthermore, we compare the horizontal and vertical pose errors of LIO-SAM [27], OL-SLAM [42] and our work on $avia\_2$ sequence. This is a challenging scene with a maximum velocity up to 7 m/s and angular velocity varying around ± 100 $^\circ/s$. Similarly, GPS signals starting from the 100th second are interrupted for approximately 30 seconds. The evaluation metric used in this section is the difference between the end point and the starting point pose coordinates. The horizontal pose deviation is evaluated using [x, y, yaw], and the vertical pose deviation is evaluated using [z, roll, pitch]. From Table V, it can be observed that FAST-LiDAR-SLAM benefits from consistent calibration brought by robust GPS, resulting in elevation error and horizontal offset in Z-axis at the centimeter level. This is superior to OL-SLAM, and higher than the decimeter level of LIO-SAM. In terms of the global pose estimation, our work shows improvement over OL-SLAM. It also outperforms LIO-SAM by 81.69%. Regarding the attitude angle control, FAST-LiDAR-SLAM maintains a tiny error of about 1°, which is better than the performance of OL-SLAM. In contrast, LIO-SAM exhibits larger errors at around 6°. In scenes with violent rotation, the LIO-SAM map diverges at multiple locations, inducing errors in trajectory estimation.

C. Efficiency

The comparison offers insights into the computational efficiency of FAST-LiDAR-SLAM and other SOTA SLAM systems. In Table VI, PRE denotes the total time consumed by preprocessing, dedistortion, and downsampling, which are

TABLE VII
AVERAGE PROCESSING TIME OF EACH MODULE FOR A LIDAR SCAN

Module	FAST-LIO (ms)	Faster-LIO (ms)	LIO-SAM (ms)	Ours (ms)
Preprocessing	0.48	0.47	1.33	0.48
Motion compensation	0.65	0.71	1.07	0.65
Downsampling	0.35	0.35	0.81	0.35
Feature extraction	1.47	0.00	1.70	0.00
Feature matching	1.12	0.00	3.02	0.00
Residual calculation	1.96	1.62	5.72	1.62
State estimation	3.90	3.96	3.35	4.10
Optimization	0.00	0.00	31.65	8.05
Sum	9.93	7.11	48.65	17.05

required to prepare the reasonable data format for further processing. OPT represents the time taken for pose calculation, predominantly driven by IESKF, encompassing tasks such as point cloud residual computation, feature matching, nearest neighbor search, plane fitting, state value iteration, and incremental optimization. In instances where the algorithm fails due to significant drift or incompatible data formats, a hyphen is used to indicate the failure in the respective box. Notably, LIO-SAM [27] and LiLi-OM [8] utilize keyframes and distributed ROS nodes for point cloud feature processing instead of sequential point cloud registration, thus, PRE and OPT times corresponding to each keyframe are calculated separately for these algorithms.

Given the evident instability of scanning points from solid-state LiDAR, especially under conditions of severe rotation or road bumps, we increase the frequency of loop detection on specific sections of the dataset. This approach helps prevent the collapse of odometry. It is observed that our work achieves competitive computational efficiency in the PRE module compared to Faster-LIO. Although the addition of pose consistency and global optimization results in a slight increase in OPT module time, FAST-LiDAR-SLAM still demonstrates a significant speed improvement over LIO-SAM [27]. Generally, the capability of FAST-LiDAR-SLAM to achieve single-frame processing within 20 ms is promising, fulfilling the operational requirements of low or medium speed (60 km/h) autonomous vehicles [5]. This illustrates that our work gets a balance between computational efficiency and mapping accuracy, making it well-suited for urban autonomous driving applications.

Table VII details the time consumption of each module. FAST-LIO [10] is a feature-based method that retrieves map points in the current FOV and performs kNN search by building new static kd-Trees, which is the time saved by our work. LIO-SAM [27] is similar to FAST-LIO, but with the added prediction of IMU values, which brings a slightly longer processing time. Since Faster-LIO and FAST-LiDAR-SLAM both use point cloud direct registration, the running time of PRE module is very close (1.48 ms for our work, and 1.53 ms for Faster-LIO [9]). Overall, the preprocessing speed of our work is notably improved compared to the feature-based methods. This underscores quick start capability of the system, which is suitable for dynamic environment mapping.

V. CONCLUSION

The main goal of LiDAR SLAM is to achieve accurate and real-time state estimation using limited on-board computing resources. Therefore, this paper proposes FAST-LiDAR-SLAM: a flexible and scalable LiDAR SLAM framework for urban autonomous driving, which is characterized by real-time state estimation and robust mapping. It is particularly suitable for multi-sensor fusion and urban mapping, as additional sensor factors can be smoothly incorporated into the graph, contributing global pose constraints to the system. In this paper, the effect of fusing the loop closure factor and the GPS factor is tested, which can eliminate the drift during long time odometry accumulation or in extreme motion scenarios. To improve the system stability in the case of GPS signal interruptions, we build a Fuzzy STEKF model to flexibly adjust the optimization strategy. Additionally, it adopts the efficient ikd-Tree data structure to organize the map, so that the pose estimation of a single-frame can be completed within 20 ms. The performance of FAST-LiDAR-SLAM is thoroughly evaluated on autonomous driving datasets with challenging scenarios and different hardware configurations. These results show that FAST-LiDAR-SLAM can achieve better accuracy or efficiency than the SOTA SLAM systems. Future work will involve testing this system in actual urban autonomous driving scenarios.

APPENDIX A
EXPLANATION OF OPERATORS

Let $\mathcal{M}$ be an n-dimensional manifold. By definition, the manifold is locally homeomorphic to the n-dimensional linear space $\mathbb{R}^n$. Thus, we can build an invertible mapping from $\mathcal{M}$'s local neighborhood to its tangent space by two encapsulated operators $\ominus$ and $\oplus$:

$$\begin{aligned}
&\ominus : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M},\\
&\oplus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n,\\
&\mathcal{M} = SO(3) : \mathbf{R} \oplus \mathbf{r} = \mathbf{R}Exp(\mathbf{r}), \mathbf{R}_1 \ominus \mathbf{R}_2 = \log\left(\mathbf{R}_2^T \mathbf{R}_1\right),\\
&\mathcal{M} = \mathbb{R}^n : \mathbf{a} \oplus \mathbf{b} = \mathbf{a} + \mathbf{b}, \mathbf{a} \ominus \mathbf{b} = \mathbf{a} - \mathbf{b}.
\end{aligned} \tag{27}$$

For a composite manifold: $\mathcal{M} = SO(3) \times \mathbb{R}^n$, there is:

$$\begin{aligned}
&\begin{bmatrix}\mathbf{R}_1\\ \mathbf{a}\end{bmatrix} \oplus \begin{bmatrix}\mathbf{R}_2\\ \mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{R}_1 \oplus \mathbf{R}_2\\ \mathbf{a}+\mathbf{b}\end{bmatrix}; \begin{bmatrix}\mathbf{R}_1\\ \mathbf{a}\end{bmatrix} \ominus \begin{bmatrix}\mathbf{R}_2\\ \mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{R}_1 \ominus \mathbf{R}_2\\ \mathbf{a}-\mathbf{b}\end{bmatrix},\\
&(\mathbf{x} \oplus \mathbf{u}) \ominus \mathbf{x} = \mathbf{u}, \mathbf{x} \oplus (\mathbf{y} \ominus \mathbf{x}) = \mathbf{y},\\
&\quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{M}, \forall \mathbf{u} \in \mathbb{R}^n,
\end{aligned} \tag{28}$$

where $Exp(\mathbf{r}) = \mathbf{I} + \frac{\mathbf{r}}{\|\mathbf{r}\|}\sin(\|\mathbf{r}\|) + \frac{\mathbf{r}^2}{\|\mathbf{r}\|^2}(1 - \cos(\|\mathbf{r}\|))$ is an exponential mapping according to Rodriguez's formula [43], $\log(\bullet)$ is the reverse mapping to transform a Lie group into a Lie algebra.

APPENDIX B
DEFINITION OF $\mathbf{f}$, $\mathbf{x}$, $\mathbf{u}$, $\mathbf{w}$ IN (5)

$$\begin{aligned}
&\mathcal{M} \triangleq SO(3) \times \mathbb{R}^{15} \times SO(3) \times \mathbb{R}^3,\\
&\mathbf{x} \triangleq \left[{}^W\mathbf{R}_I^T \ {}^W\mathbf{p}_I^T \ {}^W\mathbf{v}_I^T \ \mathbf{b}_\omega^T \ \mathbf{b}_a^T \ {}^w\mathbf{g}^T \ {}^L\mathbf{R}_I^T \ {}^L\mathbf{p}_I^T\right]^T \in \mathcal{M},\\
&\mathbf{u} \triangleq \begin{bmatrix}\boldsymbol{\omega}_m^T \ \mathbf{a}_m^T\end{bmatrix}^T, \ \mathbf{w} \triangleq \left[\mathbf{n}_\omega^T \ \mathbf{n}_a^T \ \mathbf{n}_{b_\omega}^T \ \mathbf{n}_{b_a}^T\right]^T
\end{aligned} \tag{29}$$

$$\mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{w}) = \begin{bmatrix} \boldsymbol{\omega}_m - \mathbf{b}_\omega - \mathbf{n}_\omega \\ {}^W\mathbf{V}_I + \frac{1}{2}\left({}^W_I\mathbf{R}\left(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a\right) + {}^w\mathbf{g}\right)\Delta t \\ {}^W_I\mathbf{R}\left(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a\right) \\ \mathbf{n}_{b\omega} \\ \mathbf{n}_{ba} \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{30}$$

APPENDIX C
COMPUTATION OF $\mathbf{F}_{\tilde{\mathbf{x}}_i}$ AND $\mathbf{F}_{\mathbf{w}_i}$

Define $\mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i) = \mathbf{f}(\mathbf{x_i}, \mathbf{u_i}, \mathbf{w_i})\Delta t$, then $\tilde{\mathbf{x}}_{i+1} = \mathbf{x}_{i+1} \ominus \hat{\mathbf{x}}_{i+1} = \left[\left(\hat{\mathbf{x}}_i \oplus \hat{\mathbf{x}}_i\right) \oplus \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i)\right] \ominus \left(\hat{\mathbf{x}}_i \oplus \mathbf{g}(\mathbf{0},\mathbf{0})\right)$, we replace the above equation by $\mathbf{G}(\tilde{\mathbf{x}}_i, \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i))$, and obtain:

$$\begin{aligned} \mathbf{F}_{\mathbf{w}_i} &= \left.\frac{\partial\left(\mathbf{x}_{i+1} \ominus \hat{\mathbf{x}}_{i+1}\right)}{\partial\left(\mathbf{w}_i\right)}\right|_{\mathbf{w}_i=\mathbf{0}} \\ &= \left.\frac{\partial \mathbf{G}(\mathbf{0}, \mathbf{g}(\mathbf{0}, \mathbf{w}_i))}{\mathbf{g}(\mathbf{0}, \mathbf{w}_i)} \frac{\mathbf{g}(\mathbf{0}, \mathbf{w}_i)}{\partial(\mathbf{w}_i)}\right|_{\mathbf{w}_i=\mathbf{0}} \end{aligned} \tag{31}$$

$$\begin{aligned} \mathbf{F}_{\tilde{\mathbf{x}}_i} &= \left.\frac{\partial\left(\mathbf{x}_{i+1} \ominus \hat{\mathbf{x}}_{i+1}\right)}{\partial(\tilde{\mathbf{x}}_i)}\right|_{\tilde{\mathbf{x}}_i=\mathbf{0}} \\ &= \left.\frac{\partial \mathbf{G}(\tilde{\mathbf{x}}_i, \mathbf{g}(\mathbf{0},\mathbf{0}))}{\partial(\tilde{\mathbf{x}}_i)} + \frac{\partial \mathbf{G}(\mathbf{0}, \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{0}))}{\mathbf{g}\left(\hat{\mathbf{x}}_i, \mathbf{0}\right)} \frac{\mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{0})}{\partial(\tilde{\mathbf{x}}_i)}\right|_{\tilde{\mathbf{x}}_i=\mathbf{0}}. \end{aligned} \tag{32}$$

Furthermore, by substituting the expression of $\mathbf{f}(\mathbf{x_i}, \mathbf{u_i}, \mathbf{w_i})$ (6) into the above equation, we can obtain the specific calculation form.

APPENDIX D
CONVERGENCE PROOF OF FUZZY STEKF

To prove the efficacy of the proposed Fuzzy STEKF in providing accurate estimations of the system state, the following assumptions and theorem are presented.

Assumption: The system state model (Stochastic discrete nonlinear system) is accurate, and the actual error covariance matrix of standard EKF has a finite upper bound.

Theorem: For the Fuzzy STEKF composed of equations (21) to (25), if the above assumption holds true, the estimation error covariance matrix for the improved algorithm is actually given by: $\mathbf{P}^r_{k+1} = E\left((\mathbf{X}^r_{k+1} - \hat{\mathbf{X}}_{k+1}) \cdot (\mathbf{X}^r_{k+1} - \hat{\mathbf{X}}^T_{k+1})\right)$. It has a finite upper bound:

$$\sup\left(\mathbf{P}^{r^*}_{k+1}\right) = \sup\left(\mathbf{P}^r_{k+1}\right) + \Delta\mathbf{P}, \tag{33}$$

where $\mathbf{X}^r_{k+1}$ is the actual state, $\mathbf{P}^r_{k+1}$ is the mean square error matrix of the standard Kalman filter algorithm, $\Delta\mathbf{P}$ is the additional mean square error matrix introduced by Equation (25), and the superscript r represents the mean square error matrix.

Proof: When the model is correct, the true estimation mean square error matrix is

$$\begin{aligned} \mathbf{P}^r_{k+1} &= \left(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\right)\mathbf{P}^r_{k+1|k}\left(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\right)^T \\ &\quad + \mathbf{K}_{k+1}\mathbf{R}_{k+1}\mathbf{K}^T_{k+1}, \\ \mathbf{P}^r_{k+1|k} &= \mathbf{J}_{k+1,k}\mathbf{P}^r_{k-1}\mathbf{J}^T_{k+1,k} + \mathbf{Q}_k. \end{aligned} \tag{34}$$

Due to the presence of the fuzzy fading function $g(\gamma_1)$ in (25), the filtering gain will exhibit a bias, $\mathbf{K}^*_{k+1} = \mathbf{K}_{k+1} + \Delta\mathbf{K}$, and according to the designed fuzzy rules, it is known that $\Delta\mathbf{K}$ is bounded. As a result of this, the actual mean squared error will deviate from its origin:

$$\begin{aligned} \mathbf{P}^r_{k+1} + \Delta\mathbf{P} = &\left[\mathbf{I} - \left(\mathbf{K}_{k+1} + \Delta\mathbf{K}\right)\mathbf{H}_{k+1}\right]\mathbf{P}^r_{k+1|k} \cdot \\ &\left[\mathbf{I} - \left(\mathbf{K}_{k+1} + \Delta\mathbf{K}\right)\mathbf{H}_{k+1}\right]^T \\ &+ \left(\mathbf{K}_{k+1} + \Delta\mathbf{K}\right)\mathbf{R}_{k+1}\left(\mathbf{K}_{k+1} + \Delta\mathbf{K}\right)^T. \end{aligned} \tag{35}$$

Substitute (32) into (33) and notice that: $\mathbf{H}_{k+1}\mathbf{P}_{k+1|k} - \left(\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1} + \mathbf{R}_{k+1}\right)\mathbf{K}^T_{k+1} = \mathbf{0}$, thus

$$\Delta\mathbf{P} = \Delta\mathbf{K}\left(\mathbf{H}_{k+1}\mathbf{P}^r_{k+1|k}\mathbf{H}^T_{k+1} + \mathbf{R}_{k+1}\right)\Delta\mathbf{K}^T. \tag{36}$$

Since $\Delta\mathbf{K}$ is bounded, and $\mathbf{P}^r_{k+1|k}$ and $\mathbf{R}_{k+1}$ are positive definite matrix, $\Delta\mathbf{P}$ has a finite upper bound. This concludes the proof. It can be inferred that as the iteration progresses, the errors of the Fuzzy STEKF proposed in this study will converge to finite values.

REFERENCES

[1] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[2] L. Chang, X. Niu, T. Liu, J. Tang, and C. Qian, "GNSS/INS/LiDAR-SLAM integrated navigation system based on graph optimization," *Remote Sens.*, vol. 11, no. 9, p. 1009, Apr. 2019.

[3] J. Zhang, W. Wen, F. Huang, Y. Wang, X. Chen, and L.-T. Hsu, "GNSS-RTK adaptively integrated with LiDAR/IMU odometry for continuously global positioning in urban canyons," *Appl. Sci.*, vol. 12, no. 10, p. 5193, May 2022.

[4] L. Ma, Y. Li, J. Li, W. Tan, Y. Yu, and M. A. Chapman, "Multi-scale point-wise convolutional neural networks for 3D object segmentation from LiDAR point clouds in large-scale environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 821–836, Feb. 2021.

[5] M. Elhousni and X. Huang, "A survey on 3D LiDAR localization for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1879–1884.

[6] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A comparative analysis of LiDAR SLAM-based indoor navigation for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6907–6921, Jul. 2022.

[7] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, Apr. 2020.

[8] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-LiDAR-inertial odometry and mapping," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5167–5174, Jul. 2021.

[9] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4861–4868, Apr. 2022.

[10] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[11] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

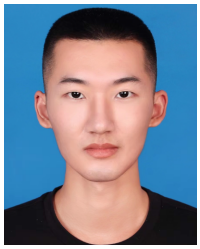
[12] G. Wan et al., "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4670–4677.

[13] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. 3rd Int. Conf. 3-D Digit. Imag. Modeling*, May 2001, pp. 145–152.

[14] E. Einhorn and H.-M. Gross, "Generic NDT mapping in dynamic environments and its application for lifelong SLAM," *Robot. Auto. Syst.*, vol. 69, pp. 28–39, Jul. 2015.
[15] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. Conf.*, vol. 2, no. 9, Berkeley, CA, USA, Jul. 2014, pp. 1–9.
[16] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4758–4765.
[17] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. 8th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2009, pp. 83–86.
[18] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3126–3131.
[19] C.-P. Hsu et al., "A review and perspective on optical phased array for automotive LiDAR," *IEEE J. Sel. Topics Quantum Electron.*, vol. 27, no. 1, pp. 1–16, Jan. 2021.
[20] R. Li, J. Liu, L. Zhang, and Y. Hang, "LiDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments," in *Proc. DGON Inertial Sensors Syst. (ISS)*, Sep. 2014, pp. 1–15.
[21] O. Kechagias-Stamatis, N. Aouf, V. Dubanchet, and M. A. Richardson, "DeepLO: Multi-projection deep LiDAR odometry for space orbital robotics rendezvous relative navigation," *Acta Astronautica*, vol. 177, pp. 270–285, Dec. 2020.
[22] M. Feng, S. Hu, G. Lee, and M. Ang, "Towards precise vehicle-free point cloud mapping: An on-vehicle system with deep vehicle detection and tracking," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 1288–1293.
[23] A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak, "Integrating deep semantic segmentation into 3-D point cloud registration," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2942–2949, Oct. 2018.
[24] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN: LiDAR-based tracking and mapping by stabilized ICP for geometry approximation with normal distributions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5143–5150.
[25] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN2: Ultra light LiDAR-based SLAM using geometric approximation applied with KL-divergence," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11619–11625.
[26] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D LiDAR inertial odometry and mapping," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3144–3150.
[27] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.
[28] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot. Sci. Syst. (RSS)*, vol. 2, no. 4, Seattle, WA, USA, 2009, p. 435.
[29] Y. Cai, W. Xu, and F. Zhang, "Ikd-tree: An incremental K-D tree for robotic applications," 2021, *arXiv:2102.10808*.
[30] D. Wilbers, C. Merfels, and C. Stachniss, "Localization with sliding window factor graphs on third-party maps for automated driving," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5951–5957.
[31] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.
[32] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, Feb. 2012.
[33] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," 2021, *arXiv:2102.03804*.
[34] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A LiDAR-inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8899–8906.
[35] M. A. Akgün, J. H. Garcelon, and R. T. Haftka, "Fast exact linear and non-linear structural reanalysis and the Sherman–Morrison–Woodbury formulas," *Int. J. Numer. Methods Eng.*, vol. 50, no. 7, pp. 1587–1606, Mar. 2001.
[36] Z. Yin, G. Li, Y. Zhang, and J. Liu, "Symmetric-strong-tracking-extended-Kalman-filter-based sensorless control of induction motor drives for modeling error reduction," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 650–662, Feb. 2019.
[37] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4802–4809.
[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[39] Z. Yan, L. Sun, T. Krajník, and Y. Ruichek, "EU long-term dataset with multiple sensors for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10697–10704.
[40] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot. Sci. Syst. (RSS)*, 2018, p. 59.
[41] L. Li et al., "SA-LOAM: Semantic-aided LiDAR SLAM with loop closure," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 7627–7634.
[42] C. Chen et al., "OL-SLAM: A robust and versatile system of object localization and SLAM," *Sensors*, vol. 23, no. 2, p. 801, Jan. 2023.
[43] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, Jan. 2013.

Zhi Rui received the bachelor's degree in urban underground space engineering from Hefei University of Technology in 2021. He is currently pursuing the degree with the School of Architecture and Civil Engineering, Xiamen University, China. His research interests include LiDAR SLAM.

Wangtu Xu received the Ph.D. degree from Beijing Jiaotong University, China, in 2010. He is currently a Professor with the School of Architecture and Civil Engineering, Xiamen University, China. His research interests include autonomous driving, deep learning, and transportation planning and management.



Zhen Feng received the bachelor's degree from Xi'an University of Architecture and Technology in 2021. He is currently pursuing the degree with the School of Architecture and Civil Engineering, Xiamen University. His research interests include autonomous driving technology and SLAM.