2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802314

SR-LIO: LiDAR-Inertial Odometry with Sweep Reconstruction

Zikang Yuan[1], Fengtian Lang[2], Tianle Xu[2] and Xin Yang[2]

***Abstract*— This paper proposes a novel LiDAR-Inertial odometry (LIO), named SR-LIO, based on an error state iterated Kalman filter (ESIKF) framework. We adapt the sweep reconstruction method, which segments and reconstructs raw input sweeps from spinning LiDAR to obtain reconstructed sweeps with higher frequency. We found that such method can effectively reduce the time interval for each iterated state update, improving the state estimation accuracy and enabling the usage of ESIKF framework for fusing high-frequency IMU and low-frequency LiDAR. To prevent inaccurate trajectory caused by multiple distortion correction to a particular point, we further propose to perform distortion correction for each segment. Experimental results on four public datasets demonstrate that our SR-LIO outperforms all existing state-of-the-art methods on accuracy, and reducing the time interval of iterated state update via the proposed sweep reconstruction can improve the accuracy and frequency of estimated states. The source code of SR-LIO is publicly available for the development of the community.**

I. Introduction

Three-dimension light detection and ranging (LiDAR) can directly capture accurate and dense scene structure information in a large range and thus has become one of the mainstream sensors in outdoor robots and autonomous driving fields. An odometry utilizing only 3D LiDAR [1], [6], [7], [16], [19], [26], [27] has the ability to estimate accurate pose in most scenarios, and transform the point clouds collected at different times to a unified coordinate system. However, there are still two main problems in LiDAR-only odometry: 1) Most existing LiDAR odometry rely on the Iterative Closest Point (ICP) algorithm for pose estimation while an inaccurate initial motion value can largely increase the time consumption. 2) For scenes without rich geometric structure information, the commonly used point-to-plane ICP algorithm usually fails due to lack of sufficiently reliable constraints for pose estimation. Introducing Inertial Measurement Unit (IMU) as an additional sensor is a promising solution to address the above two problems with little memory and time consumption.

The tightly-coupled framework also uses IMU measurements to provide motion constraints together with ICP, so as to achieve accurate and robust state estimation. The tightly-coupled LIO framework can be mainly categorized into two types: error state iterated Kalman filter (ESIKF) and optimization. The performance of all these two types both depends on the time interval of integrating IMU measurements. Specifically, for optimization framework, a long integration time will lead to a large accumulative error of pre-integration. For ESIKF framework, a long integration time will reduce the frequency of state update. In general, the less accumulative error in the predicted state, the more accurate and robust result can be estimated. However, the time interval between two consecutive sweeps of spinning LiDAR is 100 ms (i.e., 10 Hz), which makes IMU pre-integration or the predicted state have accumulative error. In addition, the low sweep rate also results in limited output frequency. The limited output frequency will cause a delay in the odometry equal to a full sweep duration [9], and put an unnecessary upper bound for the odometry bandwidth due to the Nyquist–Shannon sampling theorem [11], [15].

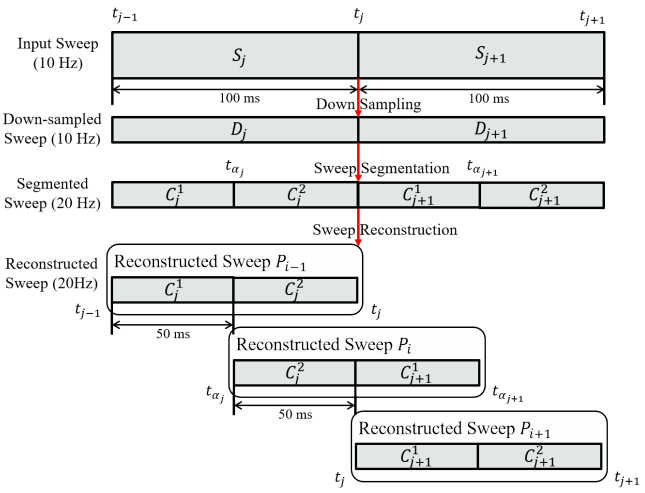


Fig. 1. Illustration of sweep reconstruction method, which splits the original sweep packet into continuous point cloud data segments, and then re-packages point cloud data streams in a multiplexing way to obtain sweeps with higher frequency.

In this paper, we found that our previous proposed sweep reconstruction method [25] can reduce the accumulative error of predicted state by reducing the time interval of IMU measurements integration for ESIKF based LIO systems, and in turn achieve more accurate and robust state estimation results. Specifically, the sweep reconstruction method uses the characteristics of continuous scanning of spinning LiDAR to segment and reconstruct raw input sweeps from spinning LiDAR to obtain reconstructed sweeps with higher frequency (as shown in Fig. 1). The increased frequency shortens

This work was supported by National Natural Science Foundation of China (62122029, U20B200007).

[1]Zikang Yuan is with Institute of Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, 430074, China. (E-mail: yzk2020@hust.edu.cn)

[2]Fengtian Lang, Tianle Xu and Xin Yang are with the Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. (E-mail: M202372913@hust.edu.cn; tianlexu@hust.edu.cn; xinyang2014@hust.edu.cn)

the time interval between two consecutive sweeps, thus reduces the time interval of IMU measurements integration and increasing the frequency of state update. Therefore, the sweep reconstruction can not only increase the frequency of output pose, but also improve the accuracy of state estimation of ESIKF based LIO. In addition, to prevent inaccurate trajectory caused by multiple inconsistent distortion correction to a particular point, we further propose to perform distortion correction for each segment, which ensured the accuracy of estimated trajectory and map.

We integrate the sweep reconstruction method and the corresponding distortion correction method into our ESIKF based LIO system to derived our SR-LIO. Experimental results on four public datasets demonstrate that: 1) our SR-LIO outperforms all existing state-of-the-art methods on accuracy and achieves higher pose output frequency; 2) reducing the time interval of iterated state update via the proposed sweep reconstruction can improve the accuracy of estimated states; 3) the distortion correction for segments can better ensure the accuracy of estimated trajectory.

To summarize, the main contributions of this work are three folds: 1) We embed the previous proposed sweep reconstruction method into our newly designed ESIKF based LIO system and achieve the state-of-the-art accuracy; 2) For reconstructed sweeps, we proposed an undistort strategy which performs distortion correction for each segment. 3) We have released the source code of this work for the development of the community[1].

Although our previous work on the LiDAR-vision system [25] introduced the concept of sweep reconstruction, the framework complexity in [25] hindered its real time performance when applying sweep reconstruction. Therefore, the improvement of output frequency by sweep reconstruction in [25] remain largely theoretical. In contrast, this work adopts sweep reconstruction method as its cornerstone, developing a LIO framework from scratch. Throughout the development process, we dedicated significant efforts to optimizing and testing, ensuring it maintains real time performance even with the inclusion of sweep reconstruction. This work marks a departure from the theoretical confines of sweep reconstruction to actualize the dual benefits in both output frequency and accuracy. In addition, the released code of this work is noted for its exceptional clarity and reproducibility, earning it 465 Stars and 64 Forks on github.

The rest of this paper is structured as follows. In Sec. II, we briefly discuss the relevant literature. Sec. III provides preliminaries. Then Sec. IV presents details of our system SR-LIO. Sec. V provides experimental evaluation. Finally, we conclude the paper in Sec. VI.

II. Related Work

The tightly-coupled framework [4], [12], [13], [17], [21], [22], [24], [28] uses IMU measurements to provide motion constraints together with ICP, so as to achieve accurate and robust state estimation. According to the type of LiDAR-inertial state estimation, the tightly-coupled system can be further divided into ESIKF based framework [4], [13], [21], [22] and optimization based framework [12], [17], [24]. LINs [13] firstly fuses 6-axis IMU and 3D LiDAR in an EKF based framework, where an ESIKF is designed to correct the estimated state recursively by generating new feature correspondences in each iteration, and to keep the system computationally tractable. Based on the mathematical derivation, Fast-LIO [22] adapts a technique of solving Kalman gain [18] to avoid the calculation of the high-order matrix inversion, and in turn greatly reduce the computational burden. Based on Fast-LIO, Fast-LIO2 [21] proposes an ikd-tree algorithm [2]. Compared with the original kd-tree, ikd-tree reduces time cost in building a tree, traversing a tree, removing elements and other operations. Point-LIO [9] proposes a point-by-point LIO framework that updates the state at each LiDAR point measurement, which allows an extremely high-frequency output. DLIO [4] proposes to retain a 3-order minimum in state prediction and point distortion calibration to obtain more accurate pose estimation result. IG-LIO [5] integrates the generalized-ICP (GICP) constraints and inertial constraints into a unified estimation framework. LIO-SAM [17] firstly formulates LIO odometry as a factor graph. Such formulation allows a multitude of relative and absolute measurements, including loop closures, to be incorporated from different sources as factors into the system. [24] firstly fuses 6-axis IMU and 3D LiDAR in an optimization based framework. Besides, to obtain more reliable poses estimation, a rotation-constrained refinement algorithm is proposed to further align the pose with the global map. LiLi-OM [12] selects the key-sweeps from solid-state LiDAR data, and performs multi-key-sweep joint LIO-optimization. However, when the type of LiDAR changes from solid-state to spinning, the time interval between two consecutive key-sweeps becomes longer than 100 ms, then the error of IMU constraints in LiLi-OM is larger than that of [24].

III. Preliminary

A. Coordinate Systems

We denote $(\cdot)^w$, $(\cdot)^l$ and $(\cdot)^b$ as a 3D point in the world coordinates, the LiDAR coordinates and the IMU coordinates respectively. The world coordinate is coinciding with $(\cdot)^b$ at the starting position.

We denote the LiDAR coordinates for taking the i_{th} sweep at time t_i as l_i and the corresponding IMU coordinates at t_i as b_i, then the transformation matrix (i.e., external parameters) from l_i to b_i is denoted as $\mathbf{T}_{l_i}^{b_i} \in SE(3)$, which consists of a rotation matrix $\mathbf{R}_{l_i}^{b_i} \in SO(3)$ and a translation vector $\mathbf{t}_{l_i}^{b_i} \in \mathbb{R}^3$. The external parameters are usually calibrated once offline and remain constant during online pose estimation. Therefore, we can represent $\mathbf{T}_{l_i}^{b_i}$ using $\mathbf{T}_l^b$ for simplicity. The pose from the IMU coordinate $(\cdot)^{b_i}$ to the world coordinate $(\cdot)^w$ is strictly defined as $\mathbf{T}_{b_i}^w$.

[1]https://github.com/ZikangYuan/sr_lio

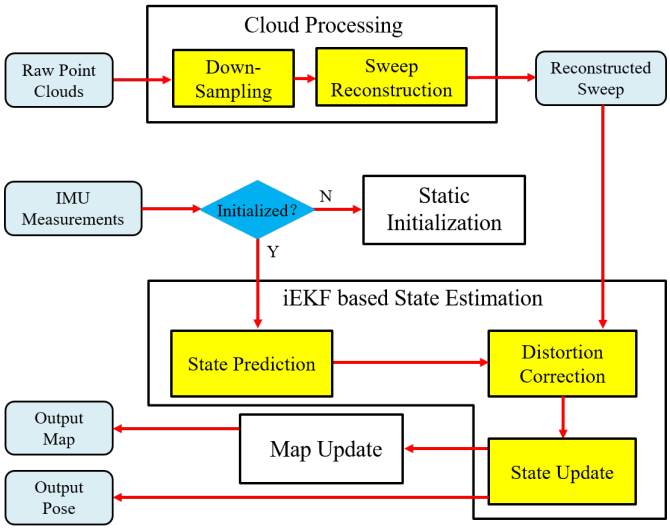


Fig. 2. Overview of our SR-LIO which consists of four main modules: a cloud processing module, a static initialization module, an ESIKF based State Estimation module and a map update module.

In addition to pose, we also estimate the velocity $\mathbf{v}$, the accelerometer bias $\mathbf{b_a}$, the gyroscope bias $\mathbf{b_\omega}$ and the gravitational acceleration $\mathbf{g}^w$, which are represented uniformly by a state vector:

$$\boldsymbol{x} = \left[\mathbf{t}^T, \mathbf{q}^T, \mathbf{v}^T, \mathbf{b_a}^T, \mathbf{b_\omega}^T, \mathbf{g}^{wT}\right]^T \tag{1}$$

where $\mathbf{q}$ is the quaternion form of the rotation matrix $\mathbf{R}$.

B. IMU Measurement Model

An IMU consists of an accelerometer and a gyroscope. The raw accelerometer and gyroscope measurements from IMU, $\hat{\mathbf{a}}_t$ and $\hat{\boldsymbol{\omega}}_t$, are given by:

$$\begin{aligned} \hat{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_{\mathbf{a}_t} + \mathbf{R}_w^t \mathbf{g}^w + \mathbf{n_a} \\ \hat{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_{\boldsymbol{\omega}_t} + \mathbf{n}_{\boldsymbol{\omega}} \end{aligned} \tag{2}$$

IMU measurements, which are measured in the IMU coordinates, combine the force for countering gravity and the platform dynamics, and are affected by acceleration bias $\mathbf{b}_{\mathbf{a}_t}$, gyroscope bias $\mathbf{b}_{\boldsymbol{\omega}_t}$, and additive noise. As mentioned in VINs-Mono [14], the additive noise in acceleration and gyroscope measurements can be modeled as Gaussian white noise, $\mathbf{n_a} \sim N\left(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{a}}^2\right)$, $\mathbf{n}_{\boldsymbol{\omega}} \sim N\left(\mathbf{0}, \boldsymbol{\sigma}_{\boldsymbol{\omega}}^2\right)$. Acceleration bias and gyroscope bias are modeled as random walk, whose derivatives are Gaussian, $\dot{\mathbf{b}}_{\mathbf{a}_t} = \mathbf{n}_{\mathbf{b_a}} \sim N\left(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{b_a}}^2\right)$, $\dot{\mathbf{b}}_{\boldsymbol{\omega}_t} = \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}} \sim N\left(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{b}_{\boldsymbol{\omega}}}^2\right)$.

IV. Our System SR-LIO

A. Overview

Fig. 2 illustrates the framework of our SR-LIO which consists of four main modules: cloud processing, static initialization, ESIKF based state estimation and map update. The cloud processing module down-samples the 10 Hz input sweep, then segments and reconstructs the 10 Hz down-sampled sweep to obtain a reconstructed sweep at 20 Hz. The static initialization module utilizes the IMU measurements to estimate some state parameters such as gravitational acceleration, accelerometer bias, gyroscope bias, and initial velocity. The ESIKF based state estimation module perform state estimation in real time, where the frequency of state update is equal to the frequency of reconstructed sweep. Finally, we add the point clouds to the map and delete the point clouds that are far away. For map management, we utilized the Hash voxel map, which is the same as CT-ICP [6].

B. Cloud Processing

1) Down-Sampling: Due to huge number of 3D point clouds to be processed, the computational burden of the whole system is heavy. In order to reduce the computational burden, we down-sample the input point clouds (i.e., S_j and S_{j+1} in Fig. 1). Firstly, we perform the quantitative down-sampling strategy, which keeps only one out of every four points. Then, we put the quantitative down-sampled points into a volume with $0.5 \times 0.5 \times 0.5$ (unit: m) voxel size, and make each voxel contain only one point, which is the same as CT-ICP [6].

2) Sweep Reconstruction: Sweep reconstruction aims to derive a 20 Hz reconstructed sweep P from the 10 Hz original input point cloud S. In theory, sweep reconstruction can increase the 10 Hz input sweep to any frequency. However, we only derive 2X reconstruction to ensure the state estimation module can handle reconstructed sweep in real time. Fig. 1 illustrates the core idea of sweep reconstruction, which is proposed in our previous work [25]. Given the last sweep S_j which begins at t_{j-1} and ends at t_j, and the current input sweep S_{j+1} which begins at t_j and ends at t_{j+1}, we assume the lengths of time intervals $[t_{j-1}, t_j]$ and $[t_j, t_{j+1}]$ are both 100 ms. Based on the characteristics of continuous acquisition over a period of time of LiDAR, we can split the original sweep packet into continuous point cloud data streams, and then re-package point cloud data streams in a multiplexing way to obtain sweeps with higher frequency. Specifically, we first calculate two equal points of the time interval $[t_{j-1}, t_j]$ and $[t_j, t_{j+1}]$ (i.e., t_{α_j} and $t_{\alpha_{j+1}}$), and put all time stamps in a set:

$$T = \left\{t_{j-1}, t_{\alpha_j}, t_j, t_{\alpha_{j+1}}, t_{j+1}\right\} \tag{3}$$

Then we take each element of T (i.e., $T[k]$) as the begin time stamp and take $T[k+2]$ as the end time stamp. We re-packet the point cloud data streams during $[T[k], T[k+2]]$ to obtain the reconstructed sweep. For instance, we packet the point cloud data streams during $\left[t_{\alpha_j}, t_{\alpha_{j+1}}\right]$ to obtain the reconstructed sweep P_i. By this way, the original sweeps S_{j+1} can be re-packed to obtain two reconstructed sweeps (e.g., P_i and P_{i+1}). Although the duration of P_i is still 100 ms, the time interval between two reconstructed sweeps (e.g., P_i and P_{i+1}) decreases from 100 ms to 50 ms. Therefore, the sweep reconstruction can increase the frequency of sweep from 10 Hz to 20 Hz.

C. Static Initialization

We adopt static initialization [8] in our system to estimate some necessary variables including initial velocity, gravita-

tional acceleration, accelerometer bias and gyroscope bias. Please refer to [8] for more details.

D. ESIKF based State Estimation

The same as Fast-LIO [22], we utilize the error state iterated Kalman filter (ESIKF) to perform state estimation. We set the error state

$$\delta \boldsymbol{x} = [\delta \mathbf{t}, \delta \boldsymbol{\theta}, \delta \mathbf{v}, \delta \mathbf{b_a}, \delta \mathbf{b_\omega}, \delta \mathbf{g}]^T \tag{4}$$

as the state variable of the filter to derive the prediction and update formula. It is necessary to note that $\delta\boldsymbol{\theta} \in so(3)$, which is the Lie algebra of rotation. $\delta \mathbf{t}$, $\delta \mathbf{v}$, $\delta \mathbf{b_a}$, $\delta \mathbf{b_\omega} \in \mathbb{R}^3$, $\delta \mathbf{g} \in S^2$ due to the fixed length of gravitational acceleration. The estimated error state would be added to the optimal state (Eq. 1) during each iteration of state update.

1) State Prediction: The state prediction is performed once receiving an IMU input (i.e., $\hat{\boldsymbol{\omega}}_{n+1}$ and $\hat{\mathbf{a}}_{n+1}$), while the optimal state $\boldsymbol{x}^w_{n+1}$ (i.e., $\mathbf{t}^w_{n+1}$, $\mathbf{R}^w_{n+1}$, $\mathbf{v}^w_{n+1}$, $\mathbf{b}_{\mathbf{a}_{n+1}}$, $\mathbf{b}_{\boldsymbol{\omega}_{n+1}}$, $\mathbf{g}^w_{n+1}$) is calculated by:

$$\begin{aligned}
\mathbf{R}^w_{n+1} &= \mathbf{R}^w_n Exp\left(\left(\frac{\hat{\boldsymbol{\omega}}_n + \hat{\boldsymbol{\omega}}_{n+1}}{2} - \mathbf{b}_{\boldsymbol{\omega}_n}\right)\Delta t\right) \\
\mathbf{v}^w_{n+1} &= \mathbf{v}^w_n + \mathbf{R}^w_n\left(\frac{\hat{\mathbf{a}}_n + \hat{\mathbf{a}}_{n+1}}{2} - \mathbf{b}_{\mathbf{a}_n} - \mathbf{R}^n_w \mathbf{g}^w_n\right)\Delta t \\
\mathbf{t}^w_{n+1} &= \mathbf{t}^w_n + \mathbf{v}^w_n \Delta t + \\
&\frac{1}{2}\mathbf{R}^w_n\left(\frac{\hat{\mathbf{a}}_n + \hat{\mathbf{a}}_{n+1}}{2} - \mathbf{b}_{\mathbf{a}_n} - \mathbf{R}^n_w \mathbf{g}^w_n\right)\Delta t^2 \\
\mathbf{b}_{\mathbf{a}_{n+1}} &= \mathbf{b}_{\mathbf{a}_n}, \mathbf{b}_{\boldsymbol{\omega}_{n+1}} = \mathbf{b}_{\boldsymbol{\omega}_n}, \mathbf{g}^w_{n+1} = \mathbf{g}^w_n
\end{aligned} \tag{5}$$

The error state $\delta \boldsymbol{x}_{n+1}$ and covariance P_{n+1} is propagated as:

$$\begin{aligned}
\delta \boldsymbol{x}_{n+1} &= \mathbf{F}_{\boldsymbol{x}} \delta \boldsymbol{x}_n \\
\mathbf{P}_{n+1} &= \mathbf{F}_{\boldsymbol{x}} \mathbf{P}_n {\mathbf{F}_{\boldsymbol{x}}}^T + \mathbf{F}_{\boldsymbol{w}} \mathbf{Q} {\mathbf{F}_{\boldsymbol{w}}}^T
\end{aligned} \tag{6}$$

where $\mathbf{Q}$ is the diagonal covariance matrix of noise ($\boldsymbol{\sigma}^2_{\mathbf{a}}$, $\boldsymbol{\sigma}^2_{\boldsymbol{\omega}}$, $\boldsymbol{\sigma}^2_{\mathbf{b_a}}$, $\boldsymbol{\sigma}^2_{\mathbf{b_\omega}}$). Δt is the time interval between two consecutive IMU measurements. $\mathbf{F}_{\boldsymbol{x}}$ is expressed as:

$$\mathbf{F}_{\boldsymbol{x}} = \begin{bmatrix}
\mathbf{I} & 0 & \mathbf{I}\Delta t & 0 & 0 & 0 \\
0 & f_{11} & 0 & 0 & -\mathbf{I}\Delta t & 0 \\
0 & f_{21} & \mathbf{I} & -\mathbf{R}^n_w \Delta t & 0 & f_{25} \\
0 & 0 & 0 & \mathbf{I} & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{I} & 0 \\
0 & 0 & 0 & 0 & 0 & f_{55}
\end{bmatrix} \tag{7}$$

$$f_{11} = \mathbf{I} - \left(\frac{\hat{\boldsymbol{\omega}}_n + \hat{\boldsymbol{\omega}}_{n+1}}{2} - \mathbf{b}_{\boldsymbol{\omega}_n}\right)^\wedge \Delta t \tag{8}$$

$$f_{21} = -\mathbf{R}^n_w \left(\frac{\hat{\mathbf{a}}_n + \hat{\mathbf{a}}_{n+1}}{2} - \mathbf{b}_{\mathbf{a}_n}\right)^\wedge \Delta t \tag{9}$$

$$f_{25} = {\mathbf{g}^w_n}^\wedge \mathbf{B}\left(\mathbf{g}^w_n\right) \Delta t \tag{10}$$

$$f_{55} = -\frac{1}{\left\|\mathbf{g}^w_n\right\|^2} \mathbf{B}\left(\mathbf{g}^w_n\right)^T {\mathbf{g}^w_n}^\wedge {\mathbf{g}^w_n}^\wedge \mathbf{B}\left(\mathbf{g}^w_n\right) \tag{11}$$

$$\mathbf{B}(\mathbf{g}) = \begin{bmatrix}
1 - \frac{\bar{\mathbf{g}}_x^2}{1+\bar{\mathbf{g}}_z} & -\frac{\bar{\mathbf{g}}_x \bar{\mathbf{g}}_y}{1+\bar{\mathbf{g}}_z} \\
-\frac{\bar{\mathbf{g}}_x \bar{\mathbf{g}}_y}{1+\bar{\mathbf{g}}_z} & 1 - \frac{\bar{\mathbf{g}}_y}{1+\bar{\mathbf{g}}_z} \\
-\bar{\mathbf{g}}_x & -\bar{\mathbf{g}}_y
\end{bmatrix} \tag{12}$$

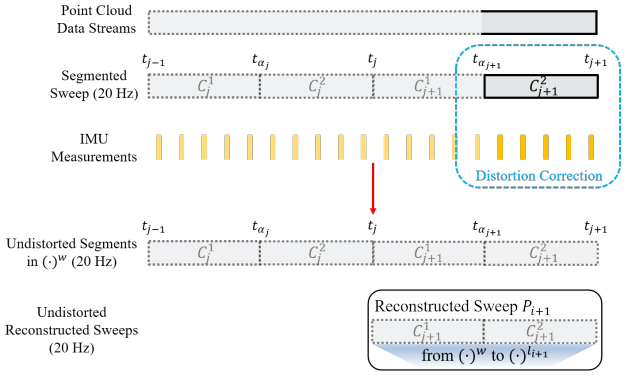


Fig. 3. Illustration of distortion correction. We perform distortion correction for each segment, but not an entire reconstructed sweep. This strategy makes the specific point to be undistorted only once, to ensure the accuracy of estimated trajectory.

where $\bar{(\cdot)}$ indicates normalization for a specific element and $(\cdot)^\wedge$ indicates the skew symmetric matrix corresponding to a vector. $\mathbf{F}_{\boldsymbol{w}}$ is expressed as:

$$\mathbf{F}_{\boldsymbol{w}} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & -\mathbf{I}\Delta t & 0 & 0 \\
-\mathbf{R}^n_w \Delta t & 0 & 0 & 0 \\
0 & 0 & -\mathbf{I}\Delta t & 0 \\
0 & 0 & 0 & -\mathbf{I}\Delta t \\
0 & 0 & 0 & 0
\end{bmatrix} \tag{13}$$

It is not difficult to see from Eq. 6 that: the uncertainty (expressed by covariance $\mathbf{P}$) of the predicted state increases with the increase of IMU measurements being integrated. Therefore, the longer time interval between two state update is, the larger accumulative error exists in the predicted state.

2) Distortion Correction: There are two options to perform distortion correction: 1) Performing distortion correction for each reconstructed sweep; 2) Performing distortion correction for each segment. The option 1) causes the point cloud in a specific period to be undistorted multi times. Since the pose used for each distortion correction are different, for a specific point, the coordinate in $(\cdot)^w$ after two corrections will not be the same. This problem would result in the inaccuracy of estimated trajectory. To solve this problem, we propose to perform distortion correction for each segment, but not an entire sweep (as shown in Fig. 3). Specifically, for each segment (e.g., C^2_{j+1}), we transform the point (e.g., $\mathbf{p} \in C^2_{j+1}$) to $(\cdot)^w$ according to IMU-integrated pose or the uniform motion model. After obtaining the reconstructed sweep P_{i+1}, we transform all points belong to it from $(\cdot)^w$ to $(\cdot)^{l_{i+1}}$ to finish the distortion correction.

3) State Update: When every new reconstructed sweep P_{i+1} completes, we iteratively perform the following steps for state update.

Step1. Point-to-plane residuals computation. During each iteration, we firstly build the point-to-plane residuals. Specifically, for a undistorted point $\mathbf{p}_k \in P_{i+1}$ $(1 \leq \mathrm{k} \leq \mathrm{m})$, we first project $\mathbf{p}_k$ to the world coordinate to obtain ${\mathbf{p}_k}^w$, and then find 20 nearest points around ${\mathbf{p}_k}^w$ from the volume. To

search for the nearest neighbor of ${\mathbf{p}_k}^w$, we only search in the voxel V to which ${\mathbf{p}_k}^w$ belongs, and the 8 voxels adjacent to V. The 20 nearest points are used to fit a plane with a normal $\mathbf{n}$ and a distance d. Accordingly, we can build the point-to-plane residual $r^{\mathbf{p}_k}$ for $\mathbf{p}_k$ as the observation constraint:

$$\begin{aligned} r^{\mathbf{p}_k} &= \omega_{\mathbf{p}} \left(\mathbf{n}^T {\mathbf{p}_k}^w + d \right) \\ {\mathbf{p}_k}^w &= \mathbf{R}_{b_{i+1}}^w \mathbf{p}_k + \mathbf{t}_{b_{i+1}}^w \end{aligned} \tag{14}$$

where $\omega_{\mathbf{p}}$ is a weight parameter utilized in [6], $\mathbf{R}_{b_{i+1}}^w$ is the rotation from $(\cdot)^{b_{i+1}}$ to $(\cdot)^w$ at t_{i+1}. We can express the observation matrix $\mathbf{h}$ as:

$$\mathbf{h} = \left[{r^{\mathbf{p}_1}}^T, {r^{\mathbf{p}_2}}^T, \cdots, {r^{\mathbf{p}_m}}^T \right]^T \tag{15}$$

The corresponding Jacobian matrix of observation constraint $\mathbf{H}$ is calculated as:

$$\begin{aligned} \mathbf{H} &= \left[{\frac{\partial r^{\mathbf{p}_1}}{\partial \boldsymbol{x}_{i+1}}}^T, {\frac{\partial r^{\mathbf{p}_2}}{\partial \boldsymbol{x}_{i+1}}}^T, \cdots, {\frac{\partial r^{\mathbf{p}_m^T}}{\partial \boldsymbol{x}_{i+1}}}^T \right]^T \\ {\frac{\partial r^{\mathbf{p}_k}}{\partial \boldsymbol{x}_{i+1}}}^T &= \begin{bmatrix} \omega_{\mathbf{p}} \mathbf{n}^T & -\omega_{\mathbf{p}} \mathbf{n}^T \mathbf{R}_{b_{i+1}}^w \mathbf{p}_k{}^\wedge & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}^T \end{aligned} \tag{16}$$

The corresponding covariance matrix of observation constraint $\mathbf{V}$ is the diagonal matrix of $(\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_m)$, while $\mathbf{V}_k = 0.001$ in our system.

Step2. Incremental computation. We define the optimal state calculated from state prediction as $\boldsymbol{x}_{b_{i+1}}^w\Big|_0$, and define the optimal state before current iteration as $\boldsymbol{x}_{b_{i+1}}^w\Big|_n$. According to the formula of state update, the incremental $\delta\boldsymbol{x}$ is calculated as:

$$\begin{aligned} \mathbf{K} &= \left(\mathbf{H}^T \mathbf{V}^{-1} \mathbf{H} + \left(\mathbf{J}_n^0 \mathbf{P} {\mathbf{J}_n^0}^T \right)^{-1} \right)^{-1} \mathbf{H}^T \mathbf{V}^{-1} \\ \delta\boldsymbol{x} &= -\mathbf{K}\mathbf{h} - (\mathbf{I} - \mathbf{K}\mathbf{H}) \mathbf{J}_n^0 \left(\boldsymbol{x}_{b_{i+1}}^w\Big|_n \boxminus \boldsymbol{x}_{b_{i+1}}^w\Big|_0 \right) \end{aligned} \tag{17}$$

where $\mathbf{J}_n^0$ is the partial differentiation of $\left(\boldsymbol{x}_{b_{i+1}}^w|_n \boxplus \delta\boldsymbol{x} \right) \boxminus \boldsymbol{x}_{b_{i+1}}^w\Big|_0$ with respect to $\delta\boldsymbol{x}$ evaluated at zero:

$$\mathbf{J}_n^0 = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times9} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times3} & \mathbf{I} - \frac{1}{2}\delta\boldsymbol{\theta}_n^0 & \mathbf{0}_{3\times9} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{9\times3} & \mathbf{0}_{9\times3} & \mathbf{I}_{9\times9} & \mathbf{0}_{9\times2} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times9} & \mathbf{J}_{\mathbf{g}_n}^0 \end{bmatrix} \tag{18}$$

$$\mathbf{J}_{\mathbf{g}_n}^0 = \mathbf{I} + \frac{1}{2}\mathbf{B}\left(\mathbf{g}_{i+1}^w\big|_0 \right)^T \delta\boldsymbol{\theta}_{\mathbf{g}_{i+1}^w}{}^\wedge \mathbf{B}\left(\mathbf{g}_{i+1}^w\big|_0 \right) \tag{19}$$

$\delta\boldsymbol{\theta}_n^0 = \mathrm{Log}\left({\mathbf{R}_{b_{i+1}}^w\Big|_0}^T \mathbf{R}_{b_{i+1}}^w\Big|_n \right)$, $\delta\boldsymbol{\theta}_{\mathbf{g}_{i+1}^w}$ is the Lie algebra of rotation from $\mathbf{g}_{i+1}^w\big|_n$ to $\mathbf{g}_{i+1}^w\big|_0$.

The definition of $\boxminus$: For the variable of type $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, $\mathbf{a} \boxminus \mathbf{b} = \mathbf{a} - \mathbf{b}$. For the variable of type $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$, $\mathbf{R}_1 \boxminus \mathbf{R}_2 = \mathrm{Log}({\mathbf{R}_2}^T \mathbf{R}_1)$. For the variable of type $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{R}^3$, $\delta\mathbf{g} \in S^2$, $\delta\mathbf{g} = \mathbf{g}_1 \boxminus \mathbf{g}_2 = \mathbf{B}(\mathbf{g}_2)^T \delta\boldsymbol{\theta}_{\mathbf{g}}$, where $\boldsymbol{\theta}_{\mathbf{g}}$ is the Lie algebra of rotation from $\mathbf{g}_1$ to $\mathbf{g}_2$.

The definition of $\boxplus$: For the variable of type $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, $\mathbf{a} \boxplus \mathbf{b} = \mathbf{a} + \mathbf{b}$. For the variable of type $\mathbf{R} \in SO(3)$ and $\boldsymbol{\theta} \in so(3)$, $\mathbf{R} \boxplus \boldsymbol{\theta} = \mathbf{R}\mathrm{Exp}(\boldsymbol{\theta})$. For the variable of type $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{R}^3$, $\delta\mathbf{g} \in S^2$, $\mathbf{g}_2 = \mathbf{g}_1 \boxplus \delta\mathbf{g} = \mathrm{Exp}(\mathbf{B}(\mathbf{g}_1)\delta\mathbf{g})\mathbf{g}_1$.

After the incremental $\delta\boldsymbol{x}$ is calculated, we update the optimal state by:

$$\boldsymbol{x}_{b_{i+1}}^w\Big|_{n+1} = \boldsymbol{x}_{b_{i+1}}^w\Big|_n \boxplus \delta\boldsymbol{x} \tag{20}$$

Step1 and **Step2** are performed alternately until one of the following convergence conditions is met: 1) The maximum number of iterations (e.g., 6) was reached. 2) The magnitude of incremental is smaller than a threshold (e.g., 0.1 degree for rotation and 0.01 m for translation). After convergence, the covariance is updated as:

$$\mathbf{P} = \mathbf{J}_{n+1}^0 (\mathbf{I} - \mathbf{K}\mathbf{H}) \mathbf{P} {\mathbf{J}_{n+1}^0}^T \tag{21}$$

E. Map Update

Following CT-ICP [6], the cloud map is stored in a volume. The size of each voxel is $1.0 \times 1.0 \times 1.0$ (unit: m) and each voxel contains a maximum of 20 points. When the state of current down-sampled sweep P_{i+1} has been estimated, we transform P_{i+1} to the world coordinate system $(\cdot)^w$, and add the transformed points into the volume map. If a voxel already has 20 points, the new points cannot be added to it. Meanwhile, we delete the points that are far away from current position.

V. Experiments

We evaluated our SR-LIO on four public datasets $nclt$ [3], $utbm$ [23], $ulhk$ [20] and $kaist$ [10]. $nclt$ is a large-scale, long-term autonomous unmanned ground vehicle dataset collected in the University of Michigans North Campus. The $nclt$ dataset contains a full data stream from a Velodyne HDL-32E LiDAR and 50 Hz data from Microstrain MS25 IMU. Different from the other three datasets (i.e., $utbm$, $ulhk$ and $kaist$), the LiDAR of $nclt$ takes 130∼140 ms to complete a 360 deg sweep (i.e., the frequency of a sweep is about 7.5 Hz). In addition, at the end of some sequences of $nclt$, the Segway vehicle platform enters a long indoor corridor through a door from the outdoor scene, yielding significant scene changes. This large differences in scenes produce great difficulties for ICP point cloud registration, and hence almost all systems would break down here. Therefore, we omit the test for these cases which usually locate at the end of the sequences. In addition, 50 Hz IMU measurements cannot meet the requirements of some systems (e.g., LIO-SAM [17]). Therefore, we increase the frequency of the IMU to 100 Hz by interpolation.

The $utbm$ dataset contains two 10 Hz Velodyne HDL-32E and 100 Hz Xsens MTi-28A53G25 IMU. For point clouds,

TABLE I
DATASETS FOR EVALUATION

	Velodyne LiDAR		IMU	
	Type	Rate	Type	Rate
$nclt$	32	7.5	9-axis	100 Hz
$utbm$	32	10	6-axis	100 Hz
$ulhk$	32	10	9-axis	100 Hz
$kaist$	16	10	9-axis	200 Hz

TABLE II

DATASETS OF ALL SEQUENCES FOR EVALUATION

	Name	Duration (min:sec)	Distance (km)
$nclt\_1$	2012-01-08	92:16	6.4
$nclt\_2$	2012-02-02	98:37	6.5
$nclt\_3$	2012-02-04	77:39	5.5
$nclt\_4$	2012-02-05	93:40	6.5
$nclt\_5$	2012-05-11	83:36	6.0
$nclt\_6$	2012-05-26	97:23	6.3
$nclt\_7$	2012-06-15	55:10	4.1
$nclt\_8$	2012-08-04	79:27	5.5
$nclt\_9$	2012-08-20	88:44	6.0
$nclt\_10$	2012-09-28	76:40	5.6
$nclt\_11$	2012-12-01	75:50	5.0
$utbm\_1$	2018-07-19	15:26	4.98
$utbm\_2$	2019-01-31	16:00	6.40
$utbm\_3$	2019-04-18	11:59	5.11
$utbm\_4$	2018-07-20	16:45	4.99
$utbm\_5$	2018-07-13	16:59	5.03
$ulhk\_1$	2019-01-17	5:18	0.60
$ulhk\_2$	2019-04-26-1	2:30	0.55
$kaist\_1$	urban_07	9:16	2.55
$kaist\_2$	urban_08	5.07	1.56
$kaist\_3$	urban_13	24.14	2.36

TABLE III

RMSE OF ATE COMPARISON OF STATE-OF-THE-ART (UNIT: M)

	LiLi-OM	LIO-SAM	Fast-LIO2	DLIO	IG-LIO	Point-LIO	Ours
$nclt\_1$	50.71	1.85	3.57	3.27	1.85	2.55	**1.34**
$nclt\_2$	91.86	7.18	2.00	1.80	**1.72**	2.45	1.80
$nclt\_3$	92.93	**2.16**	2.77	5.35	2.92	5.31	2.37
$nclt\_4$	215.91	2.70	3.60	18.10	**1.56**	1.73	1.91
$nclt\_5$	185.24	×	2.46	3.14	1.84	11.24	**1.62**
$nclt\_6$	141.83	×	2.60	12.44	2.12	14.89	**2.10**
$nclt\_7$	50.42	2.97	2.37	2.98	**1.82**	4.39	2.13
$nclt\_8$	137.05	**2.26**	2.59	7.84	2.40	16.28	2.70
$nclt\_9$	224.68	10.68	4.01	2.46	**1.68**	10.59	2.11
$nclt\_10$	×	×	2.65	7.72	1.72	16.22	**1.67**
$nclt\_11$	×	×	4.37	3.89	1.89	10.78	**1.61**
$utbm\_1$	67.16	-	15.13	14.25	17.37	22.71	**7.70**
$utbm\_2$	38.17	-	21.21	**13.85**	21.27	23.02	16.28
$utbm\_3$	10.70	-	10.81	55.28	13.75	13.81	**8.42**
$utbm\_4$	70.98	-	15.20	18.05	16.44	21.76	**11.12**
$utbm\_5$	62.57	-	13.24	14.95	×	19.88	**9.14**
$ulhk\_1$	×	1.68	1.20	2.44	1.15	1.07	**1.02**
$ulhk\_2$	3.11	3.13	3.24	×	3.31	**2.82**	3.21
$kaist\_1$	×	16.96	0.88	1.04	61.20	**0.75**	1.10
$kaist\_2$	×	×	16.27	1.91	3.01	1.08	**0.92**
$kaist\_3$	×	×	×	×	×	3.04	**1.36**

Denotations: "×" means the system fails to run entirety on the corresponding sequence, and "-" means the corresponding value is not available.

we only utilize the data from the left LiDAR. The $ulhk$ dataset contains 10 Hz LiDAR sweep from Velodyne HDL-32E LiDAR and 100 Hz IMU data from the 9-axis Xsens MTi-10 IMU. $kaist$ contains two 10 Hz Velodyne VLP-16, 200 Hz Ssens MTi-300 IMU. Two 3D LiDARs are tilted by approximately $45°$. For point clouds, we utilize the data from both two 3D LiDARs. All the sequences of $utbm$, $ulhk$ and $kaist$ are collected in structured urban areas by a human-driving vehicle. The datasets' information, including the sensors' type and data rate, are illustrated in Table I. Details of all the 21 sequences used in this section, including name, duration, and distance, are listed in Table II. For all four datasets, we utilize the absolute translational error (ATE) as the evaluation metrics. A consumer-level computer equipped with an Intel Core i7-12700 and 32 GB RAM is used for all experiments.

A. Comparison of the State-of-the-Arts

We compare our SR-LIO with six state-of-the-art LIO systems, i.e., LiLi-OM [12], LIO-SAM [17], Fast-LIO2 [21], DLIO [4], IG-LIO [5] and Point-LIO [9]. For a fair comparison, we obtain the results of the above systems based on the source code provided by the authors.

Results in Table III demonstrate that our SR-LIO outperforms state-of-the-arts for more than half sequences in terms of smaller ATE. Although IG-LIO achieves comparable results to our system on $nclt$ dataset, it shows poor accuracy and robustness on $utbm$ and $kaist$ dataset. In addition, although our accuracy is not the best on $nclt\_2$, $nclt\_3$, $ulhk\_2$ and $kaist\_1$, we are very close to the best accuracy. "-" means the corresponding value is not available. LIO-SAM needs 9-axis IMU data as input, while the $utbm$ dataset only provides 6-axis IMU data. Therefore, we cannot provide the results of LIO-SAM on the $utbm$ dataset. "×" means the system fails to run entirety on the corresponding sequence.

TABLE IV

OUTPUT POSE FREQUENCY COMPARISON

	LiLi-OM	LIO-SAM	LINs	Fast-LIO2	IG-LIO	Ours
nclt	7.5	7.5	7.5	7.5	7.5	**15**
utbm	10	10	10	10	10	**20**
ulhk	10	10	10	10	10	**20**
kaist	10	10	10	10	10	**20**

Except for our system and Point-LIO, other systems break down on several sequences, which also demonstrate the robustness of our system.

Results in Table IV demonstrate that our sweep reconstruction method can provide higher-frequency sweep than existing state-of-the-arts except Point-LIO. Limited by the frequency of LiDAR scan (e.g., 10 Hz), the frequency of iterated state update can only be performed at a maximum frequency of 10 Hz. Even if the power of computing resources (e.g., CPU) is assumed to be infinite, the iterated state update cannot be performed at higher than 10 Hz. However, with the assistance of our sweep reconstruction, the frequency of LiDAR sweep can be increased to an arbitrary value theoretically. In this work, we increase the frequency of LiDAR sweep from 10 Hz to 20 Hz. Benefit from the very low computational overhead of the framework, our system can still run in real time with 20 Hz reconstructed sweeps.

B. Ablation Study of Sweep Reconstruction

We examine the impact of sweep reconstruction on pose estimation accuracy by comparing the ATE result with vs. without sweep reconstruction. Without using the proposed sweep reconstruction, the system takes 10 Hz raw input to perform state update at 10 Hz. In addition, all other configuration parameters are unchanged.

TABLE V

ABLATION STUDY OF SWEEP RECONSTRUCTION ON RMSE OF ATE (UNIT: M)

	Ours w/o sweep reconstruction	Ours
nclt_1	1.40	**1.34**
nclt_2	1.94	**1.80**
nclt_3	5.25	**2.37**
nclt_4	×	**1.91**
nclt_5	×	**1.62**
nclt_6	2.24	**2.10**
nclt_7	2.23	**2.13**
nclt_8	×	**2.70**
nclt_9	4.00	**2.11**
nclt_10	1.97	**1.67**
nclt_11	1.85	**1.61**
utbm_1	10.74	**7.70**
utbm_2	16.98	**16.28**
utbm_3	9.94	**8.42**
utbm_4	11.35	**11.12**
utbm_5	10.04	**9.14**
ulhk_1	**1.02**	**1.02**
ulhk_2	3.36	**3.21**
kaist_1	1.14	**1.10**
kaist_2	0.95	**0.92**
kaist_3	1.56	**1.36**

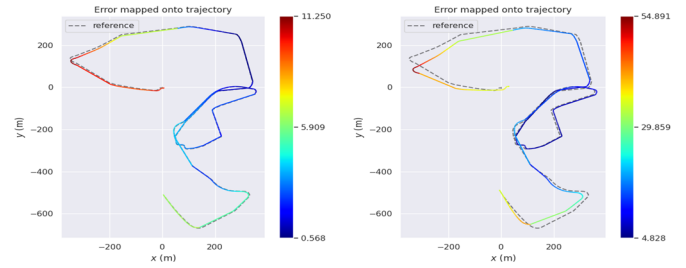
Denotations: "×" means the system fails to run entirety on the corresponding sequence.

TABLE VI

ABLATION STUDY OF DISTORTION CORRECTION ON RMSE OF ATE (UNIT: M)

	distortion correction for each segment	distortion correction for the reconstructed sweep
*nclt_1**	**1.25**	39.37
nclt_2	**1.80**	3.75
nclt_3	**2.37**	10.14
*nclt_4**	**1.79**	4.12
*nclt_5**	**1.62**	2.71
nclt_6	**2.10**	2.21
nclt_7	**2.13**	2.20
*nclt_8**	**2.57**	131.45
nclt_9	**2.11**	2.76
nclt_10	**1.67**	39.31
nclt_11	**1.61**	7.82
*utbm_1**	**5.19**	24.12
*utbm_2**	**6.12**	13.65
utbm_3	**8.42**	63.05
utbm_4	**11.12**	30.13
utbm_5	**9.14**	11.86
ulhk_1	**1.02**	1.62
ulhk_2	3.21	**3.09**
kaist_1	**1.10**	1.19
kaist_2	**0.92**	0.96
kaist_3	**1.36**	2.36

Denotations: "*" means the system fails to run entirety with distortion correction of the reconstructed sweep on the corresponding sequence, and we compare the ATE results for only a portion of the trajectory for these cases.



(a) (b)

Fig. 4. The comparison of partial trajectory of *utbm_1* with (a) distortion correction for each segment and (b) distortion correction for the reconstructed sweep directly.

Results in Table V demonstrate that increasing the frequency of state update by increasing sweep frequency can improve the accuracy and robustness of ESIKF framework.

C. Ablation Study of Distortion Correction

As illustrated in Sec. IV-D.2, the proposed distortion correction for each segment can prevent inaccurate trajectory caused by multiple inconsistent distortion correction to a particular point. Therefore, we evaluate the effectiveness of this distortion correction method in this section. As shown in Fig. 4, (a) is the partial trajectory of exemplar sequence *utbm_1* with distortion correction for each segment, (b) is the partial trajectory of exemplar sequence *utbm_1* with distortion correction for the reconstructed sweep directly. Only partial trajectories are compared because drift occurs halfway through case (b). The comparison result illustrates that the proposed distortion correction method for each segment significantly enhance the undistortion of LiDAR points, leading to more accurate trajectory. Table VI showcases the numerical results across all sequences. Noting that certain sequences were unable to finish complete processing due to the distortion correction of the reconstructed sweep (indicated by "*"). Therefore, we have focused on comparing the ATE results for only a portion of the trajectory for these cases. The results in Table VI demonstrate that the proposed distortion correction for each segment can effectively address the issue of inconsistent distortion arising from sweep reconstruction, thereby ensuring improved accuracy and robustness of the system.

D. Time Consumption

We evaluate the runtime breakdown (unit: ms) of our system for all sequences. In general, the most time-consuming modules are (1) sweep segmentation, (2) sweep reconstruction with distortion correction for each segment, (3) ESIKF based state estimation and (4) map update. Therefore, for each sequence, we test the time cost of above four modules, and the total time for handling a sweep.

Results in Table VII show that our SR-LIO takes 20∼30 ms to handle a reconstructed sweep, while the time interval of two consecutive reconstructed sweep is 65 ms on *nclt* dataset, and 50 ms on *utbm*, *ulhk*, *kaist* dataset. That means our system can not only run in real time, but also save 20∼45 ms per reconstructed sweep.

VI. CONCLUSION

This paper utilizes the previous proposed sweep reconstruction method [25] to increase the frequency of sweeps,

TABLE VII
TIME CONSUMPTION PER RECONSTRUCTED SWEEP (UNIT: MS)

	(1)	(2)	(3)	(4)	Total	Maximum Available Time
$nclt\_1$	1.86	4.35	11.24	10.38	28.32	65
$nclt\_2$	1.81	4.22	12.07	9.72	28.35	65
$nclt\_3$	1.76	3.82	11.76	9.77	27.62	65
$nclt\_4$	1.94	3.96	11.46	10.83	28.78	65
$nclt\_5$	1.87	4.39	13.66	10.28	30.77	65
$nclt\_6$	2.07	4.28	13.00	9.93	29.85	65
$nclt\_7$	2.02	4.37	12.34	7.97	27.37	65
$nclt\_8$	2.09	4.59	13.78	5.62	26.79	65
$nclt\_9$	2.02	4.26	12.45	10.51	29.90	65
$nclt\_10$	2.10	4.15	11.94	10.37	29.10	65
$nclt\_11$	1.85	3.89	11.48	9.29	27.07	65
$utbm\_1$	3.73	3.71	8.44	4.73	21.10	50
$utbm\_2$	3.99	3.86	9.32	4.76	22.45	50
$utbm\_3$	3.87	3.60	9.68	4.96	22.71	50
$utbm\_4$	3.83	3.61	8.24	5.06	21.25	50
$utbm\_5$	3.80	2.85	7.96	5.53	20.64	50
$ulhk\_1$	7.92	3.53	7.28	1.10	20.34	50
$ulhk\_2$	9.48	3.07	6.04	1.85	21.02	50
$kaist\_1$	1.94	3.79	10.29	5.56	22.16	50
$kaist\_2$	1.50	3.84	11.03	4.03	20.97	50
$kaist\_3$	1.13	3.21	11.11	5.98	21.94	50

Denotations: (1) sweep segmentation. (2) sweep reconstruction with distortion correction for each segment. (3) ESIKF based state estimation. (4) map update.

and utilizes the higher frequent reconstructed sweeps to perform state update for ESIKF based LIO framework (i.e., SR-LIO). For LIO systems, this method can not only increase the frequency of LiDAR sweep, but also reduce the accumulative error of predicted state by reducing the time interval of IMU measurements integration, and in turn achieve more accurate and robust state estimation results. In addition, for reconstructed sweeps with common data, we propose to perform distortion correction for each segment but not an entire reconstructed sweep, to better ensure the accuracy of estimated trajectory.

The proposed SR-LIO achieves the state-of-the-art accuracy on four public datasets. Meanwhile, we demonstrate the effectiveness of sweep reconstruction to improve the output frequency and accuracy of ESIKF based LIO systems.

REFERENCES

[1] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.

[2] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental kd tree for robotic applications," *arXiv preprint arXiv:2102.10808*, 2021.

[3] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

[4] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3983–3989.

[5] Z. Chen, Y. Xu, S. Yuan, and L. Xie, "ig-lio: An incremental gicp-based tightly-coupled lidar-inertial odometry," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1883–1890, 2024.

[6] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "Ct-icp: Real-time elastic lidar odometry with loop closure," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5580–5586.

[7] J.-E. Deschaud, "Imls-slam: Scan-to-model matching based on 3d data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2480–2485.

[8] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4666–4672.

[9] D. He, W. Xu, N. Chen, F. Kong, C. Yuan, and F. Zhang, "Point-lio: Robust high-bandwidth light detection and ranging inertial odometry," *Advanced Intelligent Systems*, p. 2200459, 2023.

[10] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.

[11] M. Karimi, M. Oelsch, O. Stengel, E. Babaians, and E. Steinbach, "Lola-slam: Low-latency lidar slam using continuous scan slicing," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2248–2255, 2021.

[12] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.

[13] 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 *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

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

[15] C. Qu, S. S. Shivakumar, W. Liu, and C. J. Taylor, "Llol: Low-latency odometry for spinning lidars," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4149–4155.

[16] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[17] 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 *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[18] H. W. Sorenson, "Kalman filtering techniques," in *Advances in control systems*. Elsevier, 1966, vol. 3, pp. 219–292.

[19] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam: Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4390–4396.

[20] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, "Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 2310–2316.

[21] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[22] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[23] Z. Yan, L. Sun, T. Krajník, and Y. Ruichek, "Eu long-term dataset with multiple sensors for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 697–10 704.

[24] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3144–3150.

[25] Z. Yuan, Q. Wang, K. Cheng, T. Hao, and X. Yang, "Sdv-loam: Semi-direct visual-lidar odometry and mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[26] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[27] ——, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, pp. 401–416, 2017.

[28] T. Zhang, X. Zhang, Z. Liao, X. Xia, and Y. Li, "As-lio: Spatial overlap guided adaptive sliding window lidar-inertial odometry for aggressive fov variation," *arXiv preprint arXiv:2408.11426*, 2024.