

Voxel-SLAM: A Complete, Accurate, and Versatile LiDAR-Inertial SLAM System

Zheng Liu, Haotian Li, Chongjian Yuan, Xiyuan Liu, Jiarong Lin, Rundong Li, Chunran Zheng, Bingyang Zhou, Wenyi Liu, and Fu Zhang

Abstract—In this work, we present Voxel-SLAM: a complete, accurate, and versatile LiDAR-inertial SLAM system that fully utilizes *short-term*, *mid-term*, *long-term*, and *multi-map* data associations to achieve real-time estimation and high precision mapping. The system consists of five modules: *initialization*, *odometry*, *local mapping*, *loop closure*, and *global mapping*, all employing the same map representation, an adaptive voxel map. The initialization provides an accurate initial state estimation and a consistent local map for subsequent modules, enabling the system to start with a highly dynamic initial state. The odometry, exploiting the short-term data association, rapidly estimates current states and detects potential system divergence. The local mapping, exploiting the mid-term data association, employs a local LiDAR-inertial bundle adjustment (BA) to refine the states (and the local map) within a sliding window of recent LiDAR scans. The loop closure detects previously visited places in the current and all previous sessions. The global mapping refines the global map with an efficient hierarchical global BA. The loop closure and global mapping both exploit long-term and multi-map data associations. We conducted a comprehensive benchmark comparison with other state-of-the-art methods across 30 sequences from three representative scenes, including narrow indoor environments using hand-held equipment, large-scale wilderness environments with aerial robots, and urban environments on vehicle platforms. Other experiments demonstrate the robustness and efficiency of the initialization, the capacity to work in multiple sessions, and relocalization in degenerated environments. To benefit the community, we make our code publicly available¹.

I. INTRODUCTION

3D Light Detection and Ranging (LiDAR) has become a popular sensing technology in the field of autonomous mobile robots, such as autonomous driving vehicles [1] and unmanned drones [2, 3], due to their direct, dense, active, and accurate (DDAA) depth measurements. LiDAR simultaneous localization and mapping (SLAM) and LiDAR odometry (LO) utilize the measurements of LiDAR to provide essential state feedback and a 3D dense point cloud of surroundings for the use of subsequent processes (e.g., planning, control) on the robot. Moreover, with the development of LiDAR technology, solid-state LiDARs have been extensively investigated [4]–[6] and applied in some LO [7, 8] and LiDAR SLAM systems [9, 10] due to the benefits of small size, lightweight, and affordability.

Zheng Liu, Haotian Li, Chongjian Yuan, Xiyuan Liu, Jiarong Lin, Rundong Li, Chunran Zheng, Bingyang Zhou, Wenyi Liu, and Fu Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (e-mail: {u3007335, haotianl, ycj1, xliua, zivlin, rdli10010, zhengcr, byzhou, liuwenyi}@connect.hku.hk; fuzhang@hku.hk)

¹<https://github.com/hku-mars/Voxel-SLAM>

The difference between LiDAR SLAM and LiDAR odometry (LO) is that the target of LiDAR SLAM is to build a consistent map of the environment and estimate current poses within that map in real-time on a mobile robot, whereas LO systems focus on real-time localization and accumulate the map without considering refining it to mitigate drift. Following the concepts of ORB-SLAM3 [11, 12], the superiority of LiDAR SLAM over LO is the ability to match and refine previous measurements by making full use of four types of data association:

- 1) Short-term data association: associating the current scan to a map and estimating the current ego-motion as efficiently as possible for the use of subsequent processes (e.g., planning, control). Most LO/LIO systems have the short-term data association [8, 13]–[15] and accumulate the current LiDAR scan into their maps without considering any further refinement of the state or map, causing cumulative drifts.
- 2) Mid-term data association: associating multiple latest scans to map and refining the map accordingly. This is generally achieved via a bundle adjustment (BA) technique to simultaneously refine the states and local map within a period of recent time, alleviating the cumulative drift and enhancing the robustness.
- 3) Long-term data association: associating all previous LiDAR scans to achieve global map consistency. Long-term data association allows to detect previously visited places to reset the drift by pose-graph optimization (PGO). Global BA could also be used to further improve the accuracy.
- 4) Multi-map data association: associating multiple map sessions. It uses similar methods in long-term data association to merge different map sessions, which may be caused by different collection times or re-initialization of new sessions due to LiDAR degeneration, into one.

In this work, we fully utilize these four data associations and propose the Voxel-SLAM, comprising five key modules: initialization, odometry, local mapping, loop closure, and global mapping. Voxel-SLAM builds upon the efficient LiDAR bundle adjustment of BALM2 [16], global mapping method of HBA [17], the place recognition work of BTC [18], and the map representation of VoxelMap [19]. Besides the open-source system itself, the main contributions of Voxel-SLAM are as follows:

- 1) Employment of the same map format in all tasks: initialization, odometry, local mapping, loop closure, and

global mapping. Voxel-SLAM employs an efficient and versatile adaptive voxel map, which provides sufficient features for different tasks and are well adapted to various scenarios.

- 2) Robust and fast initialization. Voxel-SLAM only requires short-time data (1s in our realization) to initialize (or re-initialize in the case of system divergence) and can initialize in both stationary and dynamic initial states. The initialization provides accurate states and a consistent map for subsequent modules.
- 3) Efficient local mapping. Voxel-SLAM uses an efficient LiDAR-inertial BA to refine the local voxel map and states within a sliding window of recent LiDAR scans, enhancing the accuracy and robustness of the system. The local mapping (window size 10) can run at the same rate as its odometry (10 Hz) in real-time on a robot onboard computer with limited computation resources.
- 4) Loop closure on multiple sessions. Voxel-SLAM has the ability to detect loops in both the current and previous sessions and optimize all the involved scan poses globally.
- 5) Efficient and accurate global mapping for single or multiple sessions. Voxel-SLAM employs a hierarchical BA to achieve efficient global optimization of scan poses and map consistency.
- 6) Full exploitation of four types of data association. Voxel SLAM exploits the short-term and mid-term data associations in its odometry and local mapping, respectively, and the long-term and multi-map data associations in both loop closure and global mapping, to achieve both real-time operation and global map consistency.

II. RELATED WORKS

A. LiDAR-Inertial Odometry and SLAM

As an early LiDAR odometry and mapping framework, LOAM [20] has greatly influenced subsequent works. It includes three modules: feature extraction, scan-to-scan odometry, and scan-to-map mapping. Through the local smoothness of each scanning line, feature points, including planar and edge feature points, are extracted to reduce the computational burden of pose estimation. Then the odometry module uses the feature points to match the current scan to the previous scan to obtain a rough relative pose in real-time. For improved accuracy, the mapping module accumulates historical feature points to build a k -d tree and takes the distances of point-to-plane (edge) as the cost to refine the poses from odometry. However, due to the requirement of rebuilding the k -d tree, the module of mapping can only run at 1 Hz, compared with the 10 Hz odometry. Following the framework of LOAM, LeGO-LOAM [21] takes a ground segmentation method in feature extraction and introduces loop closure to mitigate long-term drift. LOAM-Livox [7] adapts the framework into a solid-state LiDAR. The small field of view (FoV) and non-repetitive scanning pattern of the solid-state LiDAR make very few correspondences between two consecutive scans. Therefore, LOAM-Livox only uses the scan-to-map registration for odometry and mapping, which improves the odometry accuracy but

increases the computational loading for building the k -d tree at each scan.

Incorporating the measurements from an inertial measurement unit (IMU) can provide better initial poses, compensate for motion distortion, and enhance the robustness of estimation for LiDAR SLAM. LIOM [22], inspired by VINS [23], is one of the first open-source tightly coupled LiDAR-inertial odometry (LIO). However, due to the lack of an efficient BA method, the batch optimization in LIOM is too time-consuming to run in real-time. LiLi-OM [24], similar to LIOM using a local map and sliding window optimization for tightly coupled LIO, adapts to solid-state LiDARs and uses an ICP-based loop closure. To ensure real-time performance, LiLi-OM adopts a small optimization window. LIO-SAM [25], discarding the local map with poor efficiency in LIOM, directly feeds the LiDAR odometry results into a factor graph to optimize with IMU preintegration [26]. Thus, LIO-SAM can maintain a larger sliding window and easily fuse with other sensors, such as the GNSS (Global Navigation Satellite System). LINS [27] introduces a filter-based method into a LIO system to avoid optimizing multiple states. To reduce the dimension of computing Kalman gain, FAST-LIO [8] uses a new Kalman gain. FAST-LIO2 [13] as the state-of-the-art LIO system, avoids the feature extraction, and designs a novel incremental k -d tree. LTA-OM [10] adds the loop closure to FAST-LIO2 with the STD place recognition descriptor [28]. Faster-LIO [14] replaces the incremental k -d tree with parallel sparse incremental voxels in FAST-LIO2, for better efficiency. Point-LIO [15] is a point-by-point LIO framework with a high-frequency odometry output rate and good robustness for aggressive motion and IMU saturation. Recently, there have also been some attempts at combining LIO with LiDAR bundle adjustment (BA) to reduce the LIO drift [29, 30] but without any open-source implementation.

Compared with the mainstream open-source LiDAR-inertial SLAM systems, Voxel-SLAM has the modules of initialization to enable the system to start at a highly dynamic initial state, local mapping using a LiDAR-inertial BA to enhance the accuracy and robustness, loop closure to detect loop constraints in multiple sessions, and global mapping to further refine the global map. Voxel-SLAM takes full advantage of short-term, mid-term, long-term, and multi-map data associations to achieve real-time estimation and high accuracy mapping.

B. LiDAR Scan Registration Scheme

The registration methods for LiDAR point clouds can be divided into two categories: pairwise registration and multiview registration. Pairwise registration only considers estimating a single pose, which is the main registration method in LO and LIO. Due to the sparsity of LiDAR scans, LOAM and its successors employ point-to-plane (edge) registration instead of point-to-point registration, which is used for dense point clouds. They accumulate historical scans into a *points map* to match the latest scan. The drawback of the *points map* is that it takes additional time to form the planes or edges for new scan registration. To avoid this, Suma [31], LIPS [32], and

VoxelMap [19] use the *planes map* rather than *points map* to find the corresponding plane for each point in the current scan, achieving better efficiency and accuracy.

Using pairwise registration leads to a cumulative drift, while multiview registration can solve this by registering multiple scans simultaneously. EigenFactor (EF) [33] takes the distances of each point to the plane as cost and then transfers the cost into the minimum eigenvalue of a covariance matrix. The cost function is then optimized by a gradient-based method, leading to slow convergence due to the second-order nature of the eigenvalue-type cost. Referencing visual BA, plane adjustment (PA) [34] considers both poses and plane parameters as optimization variables, employing Schur complement to reduce the dimension brought by plane parameters. The drawbacks are that it takes more iterations to converge when the number of planes increases and the plane parameter has a singularity. For better convergence, BAREG [35] modifies the cost function by adding an extra heuristic penalty term, which could disturb the map consistency terms, especially with real-world noisy point measurements. BALM [36] introduces a second-order derivative solver and proves the geometry features can be analytically solved, which takes much fewer steps to converge than the methods mentioned earlier. The main shortcoming of BALM is that it requires enumerating all individual points to compute the Jacobian and Hessian matrix, resulting in high computational complexity due to the dense LiDAR points. To address this problem, BALM2 [16] proposes the *point cluster* to encode the raw points into a compact set of parameters and derive the relevant Hessian and Jacobian matrix. BALM2 has fast convergence speed and high accuracy. Building on BALM2, Liu *et al.* proposes a hierarchical bundle adjustment (HBA) framework [17] for large-scale global mapping.

Voxel-SLAM employs both pairwise registration and multiview registration. In the odometry, Voxel-SLAM leverages an efficient plane map (the VoxelMap [19]) to achieve efficient scan-to-map registration. In the initialization and local mapping, Voxel-SLAM adopts the BALM2 for the LiDAR(-inertial) BA optimization, considering its efficiency and accuracy. Moreover, to improve the global mapping accuracy, Voxel-SLAM incorporates the efficient HBA framework [17] after a normal PGO.

C. LiDAR-Based Place Recognition

Inspired by the normal distributions transform (NDT), Magnusson *et al.* [37] subdivides the point cloud into a regular grid of cells, encodes all cells into a histogram matrix, and detects loop closure by matching these histogram matrices. M2DP [38] projects a 3D point cloud into multiple 2D planes and generates a density signature for each plane, utilizing the left and right singular vectors of these signatures as the descriptors to match. SegMatch [39] segments the point cloud into semantic features to detect loop retrievals. Giseop Kim *et al.*, inspired by shape context [40], proposes a novel spatial descriptor named Scan Context [41], which encodes a 3D point cloud into a 2.5D information matrix. Their follow-up work, Scan Context++ [42], addresses the issues of lateral invariance

and inefficiency of brute-force search in Scan Context by proposing a novel generic descriptor and using sub-descriptors to expedite the loop retrieval. Similar to BoW [43] in visual SLAM, BoW3D [44] builds the bag of words for 3D features created by Link3D in LiDAR point cloud. In recent years, the deep neural network (DNN) has been introduced to extract local features and encode global descriptors for LiDAR SLAM (LCDNet [45], LoGG3D-Net [46], LocNet [47]). The main drawback of these learning-based methods is that they are sensitive to the training data and have limited generalization ability. They need to be re-trained when the type of LiDAR or environment is changed. Recently, Yuan *et al.* proposed a binary triangle combined (BTC) descriptor [18, 28], which introduces a robust binary feature extraction and triangle descriptor matching method with good viewpoint invariance. Besides, BTC can be adapted to any type of LiDAR easily and offers a fast loop retrieval strategy with an accurate 6-DoF relative pose estimation.

Considering both accuracy and efficiency, Voxel-SLAM incorporates the BTC [18] for loop closure detection. Similar to the voxel map-based scan to map registration and (hybrid) LiDAR bundle adjustment reviewed previously, BTC also uses a voxel map to detect planes for extracting its binary triangle combined descriptors. This makes the adaptive voxel map the unified map structure for all modules of Voxel-SLAM, including the initialization, odometry, local mapping, and global mapping.

III. SYSTEM OVERVIEW

The system overview is shown in Fig. 1. The green parts in Fig. 1 are the different modules in Voxel-SLAM, and the modules in the same thread are grouped in the same gray dashed frame. The blue parts are the data pyramid, and the red parts are the adaptive voxel map used by relevant modules. We define the system state at the end of the i -th LiDAR scan as:

$$\mathbf{x}_i = [\mathbf{R}_i \quad \mathbf{p}_i \quad \mathbf{v}_i \quad \mathbf{b}_i^g \quad \mathbf{b}_i^a] \quad (1)$$

where $\mathbf{R}_i \in \text{SO}(3)$, $\mathbf{p}_i \in \mathbb{R}^3$, and $\mathbf{v}_i \in \mathbb{R}^3$ are the rotation, position, and the velocity of the IMU in the world frame, respectively. The world frame is the first IMU frame but having its z -axis aligned with the gravity vector, which will be determined in the Initialization module (Section V-C). $\mathbf{b}_i^g \in \mathbb{R}^3$ and $\mathbf{b}_i^a \in \mathbb{R}^3$ are the bias of gyroscope and accelerator of the IMU in the body frame, respectively.

A. Workflow

As depicted in Fig. 1, Voxel-SLAM consists of five modules running in three parallel threads. Firstly, the system conducts an initialization process (if it has not done so after the start or restart) based on a short time period (1s in our current implementation) of LiDAR scan and IMU data. The initialization employs a specialized LiDAR-inertial BA optimization to estimate the states of all scans, the initial local map, and the gravity vector in the world frame. With the initialized state, local map, and gravity vector, the odometry tightly fuses the measurements from LiDAR and IMU to

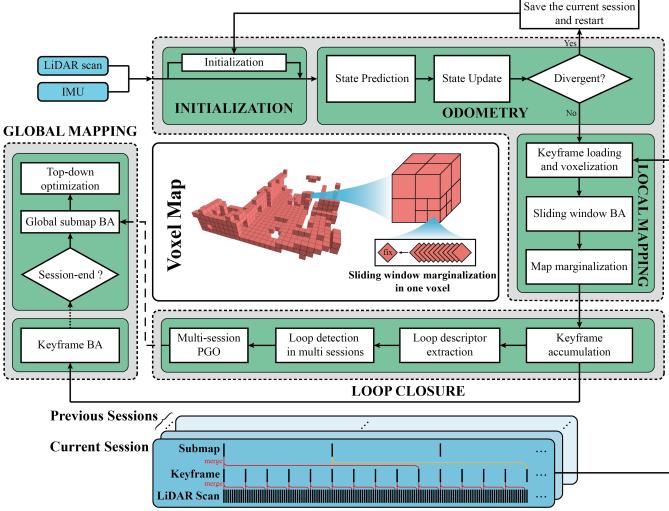


Fig. 1. The overview of Voxel-SLAM. The green parts are the different modules of Voxel-SLAM consisting of initialization, odometry, local mapping, loop closure and global mapping. The modules in the same gray dashed box are run in the same thread. The bottom blue parts are the data pyramid of multiple sessions and the center red part is the adaptive voxel map.

estimate the current state in real-time and detect potential system divergence caused by constant LiDAR degeneration. Subsequently, the local mapping incorporates the current scan into the sliding window and employs a LiDAR-inertial BA to refine all the states within the sliding window and the local map concurrently. The oldest scan within the sliding window will be marginalized to accumulate keyframes, which are then used by the loop closure module to extract loop descriptors and detect loop retrievals across both the current and previous sessions. Upon successful loop detection, the involved poses in the current or previous sessions will be used to construct a pose graph for optimization. Following the pose graph optimization (PGO), the global mapping performs a keyframe BA and merges keyframes into submaps in real time. Upon receiving a session-end signal, the global mapping executes a global BA on all submaps and a top-down optimization to obtain accurate poses of all scans.

B. Data Pyramid

A three-level pyramid of data structures is used to contain the input LiDAR point cloud sequence. The bottom level is the raw LiDAR scans, which are directly collected by LiDAR sensors, typically at 10 Hz. The LiDAR scans are used in odometry and local mapping to achieve real-time state estimation. 10 LiDAR scans are merged into one keyframe, which is used in the loop closure to extract the loop descriptor. Finally, 10 keyframes are further merged into a submap to be used in the global mapping module. The merging of LiDAR scans into keyframes and keyframes into submaps is achieved by LiDAR bundle adjustment, which concurrently optimizes the poses of all LiDAR scans (or keyframes) with respect to the first scan (or keyframe) in the respective merging window. Moreover, in the process of merging keyframes into submaps, two consecutive merging windows share five overlapped keyframes (i.e., a sliding window with a size of 10

and a stride of 5) to increase the portion of co-visible areas among the two consecutive submaps. On the other hand, the merging window from scans into keyframes has no overlap (a sliding window with a size of 10 and no stride), considering that the long LiDAR measuring distances have already ensured large overlaps among scans in two consecutive keyframes.

C. Adaptive Voxel Map

The adaptive voxel map plays a crucial role in all modules to extract plane features and provide feature associations across multiple scans, keyframes, or submaps. The voxel map maintains the plane features of different sizes in the environments. To do so, it splits the space into uniform voxels (i.e., root voxels), each with size L_r . The uniform-sized voxels are organized by a Hash table, where each Hash key maps each LiDAR point to the corresponding voxel by its point location. A voxel contains an octree data structure of multiple layers, where each leaf node represents a plane. Leaf nodes at different layers have different sizes, representing plane features of varying sizes.

An adaptive voxel map is constructed as follows. For each scan, keyframe, or submap, its points are distributed into the corresponding voxels by their positions. If the points in a voxel are on the same plane (the ratio between the minimum and the second minimum eigenvalue of the point covariance matrix is less than a specified value), the voxel is considered a plane voxel and saved with the LiDAR points for subsequent use; otherwise, the voxel is recursively split into smaller subvoxels until the points are on the same plane or the termination condition is met (e.g., reaching the minimum sub-voxel size or number of points).

The adaptive voxel map structure is used in different modules for different purposes. First, a local adaptive voxel map within the distance L_m around the LiDAR is shared by the modules of initialization, odometry, and local mapping to estimate the states of LiDAR scans in real time (see the voxel map in Fig. 1). The initialization first obtains a number of initial scans to initialize the voxel map. The odometry aligns the current scan with the planes in the voxel map to estimate the current state. The local mapping module slides the local map and refines the state of the current and recent scans through an efficient LiDAR-inertial bundle adjustment (BA) optimization. Besides the local voxel map, two other adaptive voxel map structures are also used: one used by the loop closure module, the BTC [18], for descriptor extraction in a keyframe, and the other by the global mapping module, the HBA [17], for plane feature extraction and association across different keyframes and submaps.

IV. LiDAR-INERTIAL BA OPTIMIZATION

The LiDAR-inertial BA optimization is used in the initialization and local mapping modules to estimate multiple scan states simultaneously. Compared with the filter-based estimation of LIO methods [8, 13], the estimation of LiDAR-inertial BA is more accurate and robust due to the use of longer (mid-term) data association. However, a disadvantage of the BA method is its increased computation time, preventing

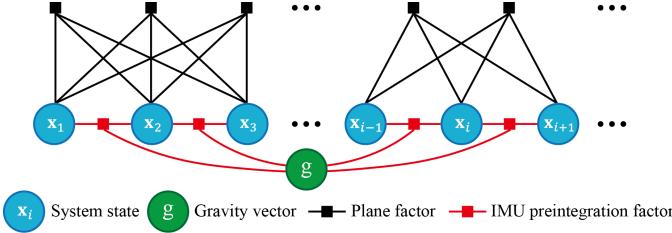


Fig. 2. The factor graph representation of the proposed LiDAR-inertial bundle adjustment.

its use in most existing LiDAR(-inertial) odometry or SLAM systems. To solve this problem, BALM2 [16] introduces an efficient LiDAR BA method by: (1) solving the geometric features (i.e., plane and line) analytically, which eliminates the feature parameters from the BA optimization, hence drastically reducing the optimization dimension; (2) introducing the *point cluster* data structure, which aggregates all points into a compact representation, hence avoiding the enumeration of each raw point in the BA; (3) deriving the analytical second-order derivatives of the BA optimization, which enable the development of an efficient second-order solver. The three techniques make the LiDAR BA optimization highly efficient for real-time implementation.

We employ the method of BALM2 and further combine it with IMU preintegration [26] to formulate a LiDAR-inertial BA optimization to estimate multiple states concurrently as well as the gravity vector if needed (see the factor graph representation in Fig. 2). The cost function is:

$$\arg \min_{\mathcal{X}} \left(\frac{1}{2} \sum_{i=1}^{N-1} \|\mathbf{r}_{i,i+1}(\mathcal{X})\|_{\Sigma_{i,i+1}}^2 + \sum_{j=1}^M \lambda_j^{\min}(\mathcal{X}) \right) \quad (2)$$

$$\mathcal{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N, \mathbf{g}] \quad (3)$$

where \mathcal{X} is the state vector consisting of individual state \mathbf{x}_i and the gravity vector \mathbf{g} , $\mathbf{r}_{i,i+1}(\mathcal{X})$ is the residual of IMU preintegration between i -th and $(i+1)$ -th states with its covariance matrix $\Sigma_{i,i+1}$, λ_j^{\min} is the minimum eigenvalue of the points covariance matrix associated to the j -th plane feature, which accounts for the contribution of LiDAR BA factor (see details in [16]), and N and M are the number of involved states and plane features, respectively. The residual of IMU preintegration $\mathbf{r}_{i,j}$ ($i < j$) directly follows [26]:

$$\begin{aligned} \mathbf{r}_{i,j} &= [\mathbf{r}_{\Delta \mathbf{R}_{ij}}, \mathbf{r}_{\Delta \mathbf{p}_{ij}}, \mathbf{r}_{\Delta \mathbf{v}_{ij}}, \mathbf{r}_{\Delta \mathbf{b}_{ij}^g}, \mathbf{r}_{\Delta \mathbf{b}_{ij}^a}] \\ \mathbf{r}_{\Delta \mathbf{R}_{ij}} &= \text{Log}(\Delta \tilde{\mathbf{R}}_{ij}^T(\mathbf{b}_i^g) \mathbf{R}_i^T \mathbf{R}_j) \\ \mathbf{r}_{\Delta \mathbf{p}_{ij}} &= \mathbf{R}_i^T (\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta t_{ij} - \frac{1}{2} \mathbf{g} \Delta t_{ij}^2) - \Delta \tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) \\ \mathbf{r}_{\Delta \mathbf{v}_{ij}} &= \mathbf{R}_i^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) \\ \mathbf{r}_{\Delta \mathbf{b}_{ij}^g} &= \mathbf{b}_j^g - \mathbf{b}_i^g \quad \mathbf{r}_{\Delta \mathbf{b}_{ij}^a} = \mathbf{b}_j^a - \mathbf{b}_i^a \end{aligned} \quad (4)$$

where \mathbf{R}_i , \mathbf{p}_i , \mathbf{v}_i , \mathbf{b}_i^g , and \mathbf{b}_i^a constitute the state \mathbf{x}_i as defined in (1), and the Jacobian matrices of $\mathbf{r}_{i,j}$ about state \mathbf{x}_i are the same with [26]. $\Delta \tilde{\mathbf{R}}_{ij}^T(\mathbf{b}_i^g)$, $\Delta \tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a)$, and $\Delta \tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a)$ are the corrected preintegrated rotation, position, and velocity using first-order expansion in [26]. Δt_{ij} is the interval between

time i and j . $\text{Log} : \text{SO}(3) \rightarrow \mathbb{R}^3$ maps from the Lie group to the vector space. $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector.

It is noted that the gravity vector is treated as a vector in \mathbb{R}^3 with scale subject to optimization instead of \mathbb{S}^2 with fixed scale as in visual SLAM. This is because the measurement of LiDAR has accurate scale information, making it possible to determine the scale from the data. In contrast, the visual measurements is well-known of lacking the scale information, making a fixed-scale gravity necessary in most visual SLAM systems [11, 23].

The cost function in (2) is optimized iteratively using a second-order solver, the LM solver. To speed up the optimization process, we derived the Jacobian and Hessian of the cost function in analytical form. The details of Jacobian, Hessian, and the solver can be found in Appendix A.

V. INITIALIZATION

Many filter-based LIO systems [8, 13]–[15] assume the sensor is initially static. For non-static initial states (e.g., restart in the middle of a sequence), the static assumption will propagate incorrect states, leading to errors in distortion correction of LiDAR scans and feature association in scan registration. This can significantly impact the accuracy of the earlier time of the state estimation and result in an inconsistent (or even incorrect) initial map that further degrades the subsequent odometry. To address this limitation, we propose a robust and efficient initialization stage, based on the previous LiDAR-inertial BA optimization, to provide robust estimation of the initial states, the gravity vector, and a consistent initial local map even in the presence of a non-static initial state. Besides, the module can confirm if this initialization is successful and re-align the gravity vector to the z -axis of the world frame.

The initialization is conducted on the first N LiDAR scans along with corresponding IMU measurements. To be specific, it first runs the LIO method in [8] in real time to estimate roughly the states of the first N scans and the gravity vector. The estimated states and the gravity vector serve as the initial values for the subsequent coarse-to-fine voxelization and BA optimization (Section V-A). The optimized states and gravity vector are used to determine whether the initialization is successful (Section V-B). If yes, the initialization is considered complete (Section V-C); otherwise, the system will collect the next N LiDAR scans and repeat the above initialization process until success.

A. Coarse-to-Fine Voxelization and BA Optimization

Given a rough initial state (e.g., estimated by LIO), the coarse-to-fine voxelization and BA optimization aim to refine the state and corresponding map based on the LiDAR-initial BA optimization outlined in Section IV. Inaccurate initial states may introduce large discrepancies in the registered point cloud map, leading to false positive or false negative plane associations in the voxelization process (see Section III-C). The wrong plane associations then cause the errors of BA optimization in turn. To overcome this issue, we propose a coarse-to-fine voxelization and BA optimization where the criterion for determining a plane feature in the adaptive

voxelization process (Section III-C) is tightened gradually. At first, the criterion is looser, recalling plane features even under large initial state errors. Based on the recalled plane features, the LiDAR-inertial BA optimization is performed. The refined state is then used to compensate for the motion distortion in each scan and to construct the next round of the adaptive voxel map, which has a tighter criterion for plane association. The voxelization and BA optimizations are conducted many rounds until convergence, where in each round the plane criterion is tightened from the previous round. The coarse-to-fine process is considered to converge when the decrement in the cost value, from the last round to the current one, is less than a certain value.

B. Criteria for Initialization Success

Three criteria are used to determine if the initialization is successful: (1) the coarse-to-fine process should converge within a certain number of rounds; (2) the magnitude of the optimized gravity vector should be close to the value of 9.8 m/s^2 ; (3) the initial voxel map, as constructed from the optimized state, should contain plane constraints in three linearly independent directions. This degeneration assessment is detailed in Appendix B. If the degeneration occurs in the final BA optimization, it means the involved states in the initialization are not constrained well with sufficient plane features and could have large estimation errors.

C. Completion of Initialization

Upon successful initialization, we obtain the accurate gravity vector and states with undistorted point clouds from the first N scans. The gravity vector is then used to align to the z -axis of the world reference frame by a rotation matrix \mathbf{R}_g^z , and the states are also rotated by the matrix. Afterwards, the gravity vector will be fixed in the subsequent modules. The adaptive voxel map built from the optimized states is serving as the initial local map for the subsequent odometry module. The initialization is conducted only once for each start (or restart), meaning that the subsequent LiDAR scans and IMU measurements will be fed to the odometry directly.

VI. ODOMETRY

In Voxel-SLAM, the odometry exploits the short data association through scan-to-map registration. The aim of the odometry is to estimate the current state with minimal time delay, which is necessary for other tasks such as planning and control. The estimated state also provides a relatively accurate initial value for the subsequent local mapping module.

A. State Prediction and Update

To conduct state estimation on the local adaptive voxel map, we use the method in [19] for scan-to-map correspondence and the extended Kalman filter (EKF) in [8] to tightly couple LiDAR and inertial measurements. Specifically, upon synchronizing the LiDAR scan and IMU measurement, the odometry predicts the state through IMU propagation and, meanwhile, compensates for the motion distortion of the LiDAR scan. The

points in the undistorted scan are down-sampled spatially at a prescribed resolution L_d and transformed from the LiDAR body frame to the IMU body frame by the pre-known LiDAR-IMU extrinsic parameter. Each point in the scan computes a covariance matrix by the method in [19] and is projected into the world frame to match the plane in the voxel map with the maximum probability. The point-to-plane distances are utilized as the system measurements, which update the IMU-predicted state within an error-state iterative Kalman filter [8]. The estimated state and LiDAR scan will be published to other modules for further processing.

B. Detection of System Divergence

When a robot encounters a degenerate environment that lacks sufficient geometric features, the state estimation becomes ill-posed. This would cause the SLAM system to diverge and fail, which should be actively monitored and handled. In Voxel-SLAM, we detect the potential divergence of the system after each scan registration. After the state estimation in the odometry, the planes used in the scan-to-map correspondence are collected to assess the degeneration by the method in Appendix B. An occasional degeneration in a scan is not considered a divergence due to the relatively accurate IMU prediction over a short period. Constant degeneration over consecutive scans will be considered a divergence. In such cases, the current data session is saved, and the whole system will restart from a new origin that is far apart from the current session to prevent the points in the next session from overlapping with those in the current session. The restarting will also trigger a new initialization to obtain an initial state and local map for the next session.

VII. LOCAL MAPPING

The local mapping, leveraging the mid-term data association, refines the local voxel map and all recent states within a sliding window. Compared with the odometry, the local mapping is more accurate and robust since its LiDAR-inertial BA takes advantage of a longer period of measurements from LiDAR and IMU for state estimation.

A. Keyframe Loading and Voxelization

When the local mapping receives the IMU measurements and the current LiDAR scan with its estimated state from the odometry, the points in the scan are distributed into corresponding voxels by their positions in the world frame. Meanwhile, the IMU measurements are preintegrated according to [26] to obtain the IMU preintegration factors. Due to only maintaining a local map within the distance L_m around the current LiDAR position, the voxel map cannot ensure long-term data association in the case of revisiting a place existing in the current or previous sessions. To establish long-term association, we dynamically load the keyframes that are temporally distant (e.g., $\geq 1 \text{ min}$ to avoid recent keyframes) but spatially close (e.g., $\leq 10 \text{ m}$) to the current scan, a technique proved to be effective in [10]. The points in these keyframes are distributed into corresponding voxels, serving

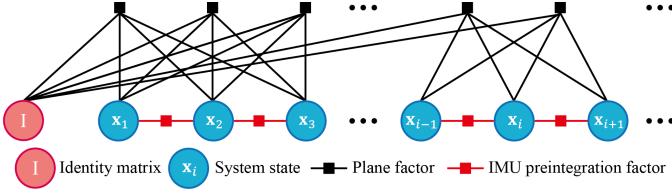


Fig. 3. The factor graph representation of the LiDAR-inertial bundle adjustment used in local mapping. The “identity matrix” represents the map constraints from the “fix” points (represented in the form of point cluster), which are out of the sliding window optimization and are represented in the world frame.

as fixed historical map constraints for the sliding window BA optimization.

When distributing points from the current scan or dynamically-loaded keyframes into voxels, we keep track of the voxels that have newly added points. These voxels are enumerated, where all the contained points, either from the current scan, keyframes, or existing points, go through an adaptive voxelization (Section III-C) until each leaf node has a valid plane. In each leaf node, all points are encoded into a compact structure, *point cluster* [16], and different scans within the sliding window will have different point clusters. The remaining points in the leaf node are termed as “fix” points and serve as the map constraints for the sliding window optimization (these points may be from the dynamically-loaded keyframes or previously-marginalized scans, as in Section VII-C). These “fix” points are encoded into a single point cluster, which, along with the point cluster of each scan in the sliding window, will be used directly for BA optimization without enumerating each individual point.

B. Sliding Window BA Optimization

After obtaining the point cluster associations and IMU preintegrations, the sliding window BA optimization employs the LiDAR-inertial BA optimization in Section IV to refine all the states within the sliding window. One difference between the sliding window BA optimization and that in Section IV is that the sliding window BA optimization incorporates the constraints from the “fix” points in each voxel, serving as map constraints to ensure the accuracy of the estimation in the world frame. Another difference is that the gravity vector is fixed, considering that the initialization has estimated the gravity vector well. Fig. 3 is a factor graph representation of the sliding window LiDAR-inertial BA optimization.

C. Map Marginalization

After the BA optimization, the point clusters and plane parameters (plane centers and normal vectors) in the local voxel map are updated by the refined states. The updated planes will be used in the next scan-to-map correspondence of the odometry module. Moreover, the oldest scan within the sliding window will be marginalized. That is, the oldest scan is removed from the sliding window, and its points in each leaf node are merged into the “fix” point cluster as map constraints. After the marginalization, the oldest scan and its pose will be sent for loop closure for further processing. Moreover, the

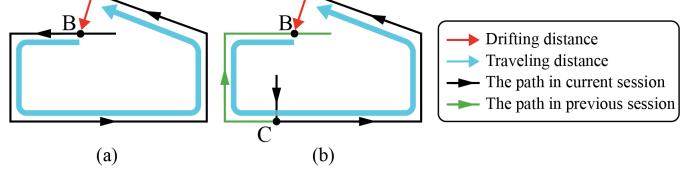


Fig. 4. Drifting and traveling distances. (a) The candidate loop keyframe is in the current session. The traveling distance is the distance accumulated along the current session from the candidate loop keyframe B to the current keyframe A. (b) The candidate loop keyframe is in a previous session. The traveling distance is the distance accumulated along the previous session from the candidate loop keyframe B to the previous loop keyframe C and then along the current session from C to the current keyframe A. For both cases, the drifting distance is the relative distance between the current keyframe A and the candidate loop frame B, which is computed from the loop detection method BTC [18].

voxels whose leaf nodes have no points corresponding to the states within the sliding window or whose distances from the current position are larger than the local map size L_m , will be removed from the local map to reduce memory use.

VIII. LOOP CLOSURE

Loop closure serves two important purposes within our framework. Firstly, by exploiting long-term data association, it effectively mitigates drift by detecting previously visited locations and corrects the cumulative errors by pose graph optimization (PGO). Secondly, loop closure matches the current session with previous sessions in the same world frame to achieve multi-map data associations. This process enables the integration and alignment of data collected from different mapping sessions, facilitating the creation of a unified map of the environment.

A. Keyframe Generation and Loop Detection

A scan, after being marginalized from the sliding window in the local mapping, will be pushed into the pose graph by the loop closure module. N consecutive marginalized scans accumulate into a frame, which is then selected as a keyframe if the moving distance and rotation angle between the current frame and latest keyframe are larger than predetermined thresholds (e.g., 0.5 m and 5 deg). The keyframe is used to extract a BTC descriptor and to match candidate loop keyframes in the current and previous sessions, as detailed in [18]. The keyframe is meanwhile sent to the thread for global mapping for further processing.

Upon identifying a loop closure candidate, it is essential to verify its authenticity to avoid false-positive detection. In addition to the built-in geometric verification of BTC [18], we take two extra criteria to enhance the reliability of loop detection. (1) The point-to-plane alignment of the current keyframe with the detected candidate should contain plane constraints in three linearly independent directions with the criteria in Appendix B. (2) The ratio of drifting distance to traveling distance should be small enough (e.g., $\leq 1\%$). This criterion applies to the cases where the candidate loop keyframe is within the current session (see Fig. 4(a)) or within the previous session for the second (or more) time (see Fig.

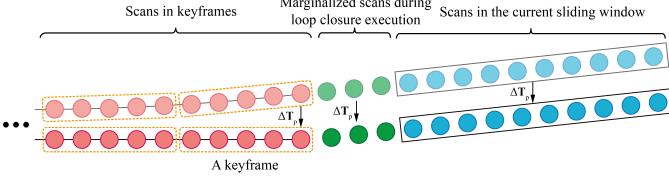


Fig. 5. The scans before and after a PGO. The red nodes are the scans that have been added to the pose graph. The green nodes are the marginalized scans from the local mapping during the loop closure execution, hence having not been added to the pose graph yet. The blue nodes are the scans in the current sliding window of the local mapping. The nodes at the top and bottom are the scans before and after PGO, respectively. ΔT_p is the pose correction of the last scan in the pose graph and used to correct the subsequent scan beyond the pose graph.

4(b)). This criterion is based on the observation that, in a well-performing SLAM system, the drift after a fixed distance of travel is basically within a certain value. If the ratio is too large, this loop retrieval is most likely a mismatch. Moreover, if the candidate loop keyframe is within the previous session for the first time, the second criterion would not apply, and the first criterion as well as the geometric verification within BTC [18] should be more stringent considering the detrimental effect on the subsequent operation of the system if the first loop detection across different sessions is wrong.

B. Pose Graph Optimization and Map Rebuilding

Upon a successful loop detection, if the detected loop keyframe is in the previous session for the first time, the current session is connected to the previous session by aligning all poses in the current session into the world frame of the previous session and concatenating the pose graph of the two sessions. Otherwise, the detected loop keyframe is in the previous session for the second (or more) times or in the current session, either case will cause a loop within the pose graph, which subsequently triggers a pose graph optimization (PGO) via GTSAM [48].

After updating scan poses via PGO, the poses of keyframes and submaps are also synchronized by their corresponding updated scan poses, as shown in the data pyramid of Fig. 1 (see the red nodes in Fig. 5). Moreover, the poses of the scans beyond the pose graph should also be updated to ensure the consistency of the system. These scans consist of (1) marginalized scans during the loop closure execution (see the green nodes in Fig. 5); and (2) all scans in the current sliding window, which has slid forward after the last scan was added to the pose graph (see the blue nodes in Fig. 5). These two types of scans have their poses all corrected by the transformation, ΔT_p , which is the pose correction of the last scan in the pose graph as shown in Fig. 5. The two types of scans with updated poses, along with all scans in the latest five keyframes in the pose graph, are then used to build a new adaptive voxel map. In each leaf node of the new voxel map, points belong to the keyframes, and marginalized scans are the “fix” points serving as the map constraints, so they are encoded into one single point cluster. In contrast, points belonging to scans in the current sliding window are encoded into point clusters separately (see Section VII-A). Finally, the

new voxel map will replace the original voxel map to be used by the odometry and local mapping in the future. To save computational resources, the PGO and map rebuilding are conducted only when the drifting distance in Section VIII-A is larger than a predetermined threshold (e.g., 0.1m).

IX. GLOBAL MAPPING

The point cloud map from the loop closure, particularly in multi-session SLAM, may exhibit inconsistencies, due to the sole consideration of pose constraints in pose graph optimization (PGO), which does not capture direct map consistency constraints. To address this, we introduce the global mapping to further refine the poses and maps from the loop closure, employing a hierarchical global BA method [17] to mitigate excessive optimization dimensions.

Based on the data pyramid in Fig. 1, the global mapping thread encompasses three procedures. Firstly, after receiving keyframes from the loop closure thread, the global mapping thread executes a bundle adjustment (BA) on a window of keyframes in real-time. The window has a size of 10 and slides with a stride of 5. In each sliding window BA, the pose of the first keyframe is fixed at the PGO result, and the rest nine keyframe poses are optimized concurrently in the optimization. After that, the 10 keyframes in each window are merged into a submap. Secondly, once the current session ends (e.g., data collection ends or the system restarts), a global BA on submaps is conducted. Considering that the scan pose returned by PGO may not render sufficient global map consistencies, the coarse-to-fine voxelization and BA optimization method in Section V-A is adopted for the global submap BA, where the IMU pre-integration factors and gravity optimization are removed. The BA optimization from keyframes to submap, and then from submap to global map, is known as the bottom-up BA optimization in [17], which significantly reduces the optimization dimensions compared to the global BA directly on keyframes or scans. Finally, a top-down optimization [17] with PGO is employed to improve the global consistency.

X. EXPERIMENTAL RESULTS

We perform five sets of experiments to validate the accuracy and versatility of our system, as follows:

- 1) Evaluating the robustness, accuracy, and time consumption of the initialization.
- 2) Single-session SLAM benchmark comparison with other odometry and SLAM methods.
- 3) Multi-session SLAM experiments.
- 4) Multi-session SLAM with Online relocalization on a computation-limited computer.
- 5) Computational time of experiments 2)-4) above.

We use three public datasets with significant differences: Hilti [49, 50], MARS-LVIG [51], and UrbanNav [52], which include a total of 30 sequences, and a private dataset with 2 sequences. The details of all the 32 sequences about their duration and distances are listed in Table VIII of the Appendix C. The Hilti is a SLAM dataset collected in indoor and outdoor structured construction environments. We use their handheld sequences, which are collected by a Hesai XT-32 LiDAR and

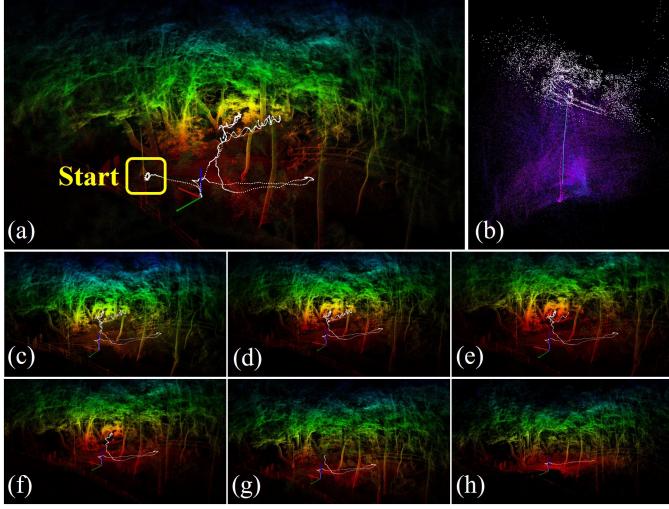


Fig. 6. A sequence with large initial motion. (a) Voxel-SLAM initializes successfully and completes the whole sequence. (b) FAST-LIO2 is divergent due to the violent initial movement. (c)-(h) Voxel-SLAM successfully initializes when starting from the 1/7 to 6/7 of the sequence, which testifies the robustness of initialization.

Bosch BMI085 IMU at 400 Hz, and can export the absolute trajectory error (ATE) results from their website. The MARS-LVIG collects the data on an unmanned aerial vehicle (UAV) from a downward-looking Livox Avia LiDAR (with a built-in IMU BMI088 at 200 Hz) at a height of about 100 meters. The UrbanNav is an urban robot-car dataset with the Xsens-MTI-30 IMU at 400 Hz and a HDL-32E LiDAR. Our private dataset is collected on a light-weight handheld device, as shown in Fig. 13, assembled with a Livox Avia LiDAR and its internal IMU. In all the above sequences, the LiDAR runs at 10 Hz.

The first three experiments mentioned above were run on a laptop computer with an Intel i7-10750H CPU at 3.5 GHz and 32 GB of memory. The fourth one runs on an onboard computer with an Intel i3-N305 CPU at 3.0 GHz and 16 GB of memory. The root voxel size and spatial down-sampling resolution for the scan are $L_r = 1$ m and $L_d = 0.1$ m in all indoor environments, $L_r = 2$ m and $L_d = 0.25$ m in all outdoor scenario, and $L_r = 4$ m and $L_d = 0.5$ m in the high-altitude downward-looking environment, regardless of the datasets or sequences. The local voxel map size is $L_m = 1,000$ m and the maximum layer of the map is $l_{max} = 3$. The minimum number of points on a plane is $N_{min} = 5$, and the plane criteria is $\frac{\lambda_3}{\lambda_2} < \frac{1}{16}$, where λ_l ($l = 1, 2, 3$) is the eigenvalue of the plane covariance matrix with $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The sliding window size is $N = 10$ for the initialization, LiDAR-inertial BA, and keyframe BA. These parameters are kept the same across all datasets and sequences. More details of the experiments can be found in the video².

A. Initialization

1) *Qualitative analysis:* To demonstrate the robustness of our initialization, we tested it on a private sequence called “private1” collected in an unstructured forest environment.

²<https://youtu.be/Cg9W01aIUzE>

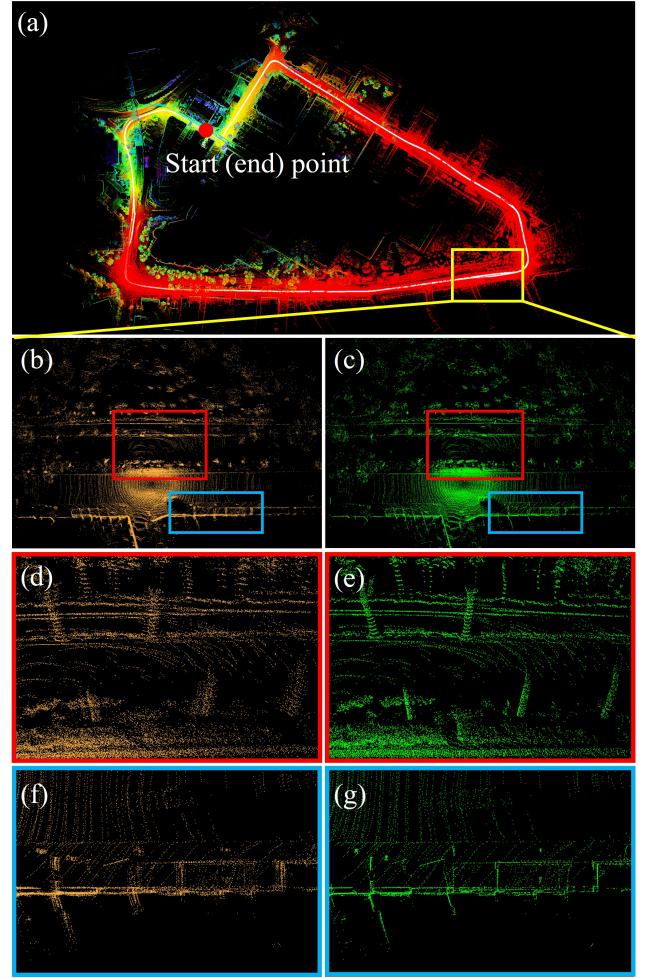


Fig. 7. (a) The point cloud map of the sequence “urban1”. (b)-(g) are the point cloud map obtained from the Initialization beginning at the location of yellow frame in (a), which has an initial velocity of 9.6 m/s. The left column of figures (b, d, f) are the map with inaccurate scan states before the BA in initialization, and the right column figures (c, e, g) are the map after the BA in initialization.

The sequence has fast rotation and quick shaking with an initial angular velocity of 83.1 deg/s and linear acceleration of 6.7 m/s² (excluding the gravity acceleration). Despite the large initial conditions, the Voxel-SLAM can still initialize successfully and generate a trajectory reflecting the real motion with a consistent map, as shown in Fig. 6(a). The odometry of FAST-LIO2 is divergent at the beginning due to the absence of a robust initialization module. FAST-LIO2 accumulates a short period of IMU measurements to compute an average linear acceleration vector. The vector is then scaled to 9.8 m/s² to serve as the estimated gravity vector. Since the initial acceleration is up to 6.7 m/s² (excluding the gravity acceleration), the direction of the gravity vector computed by FAST-LIO2 is far from the truth, causing the divergence as shown in Fig. 6(b). To further prove the effectiveness of the initialization module, we start our system at various beginning time along the sequence, ranging from $\frac{1}{7}$ to $\frac{6}{7}$ of its duration. The corresponding results are presented in Fig. 6(c) to (h), respectively. These results further prove the robustness of the initialization module under various initial conditions and

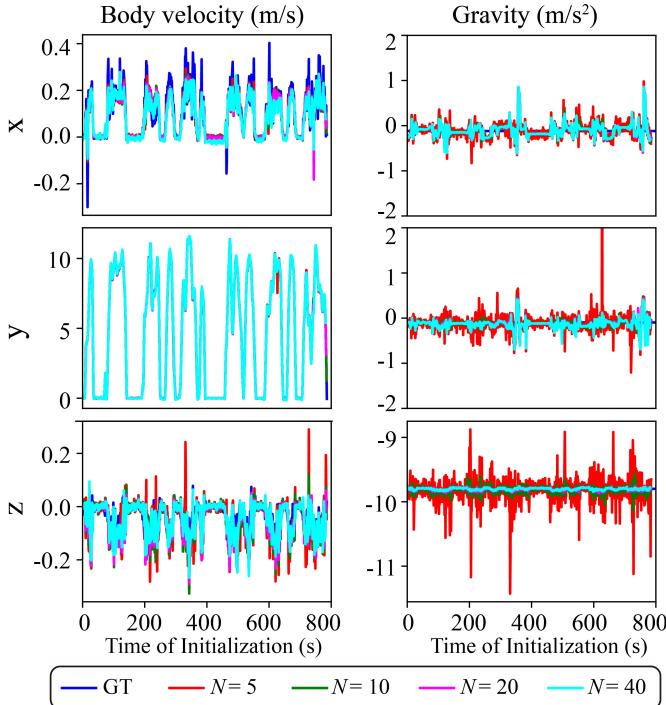


Fig. 8. The estimated initial velocity and gravity vector in the sequence “urban1”. The X axis of each subplot is the time along the sequence when the initialization starts executing. Different colors in all plots represent the estimation with different sliding window size.

environments.

2) *Quantitative analysis:* Next, we conduct a quantitative analysis of the initialization using the sequence “urban1” in UrbanNav [52]. This dataset was collected on a highway and provides the ground-truth poses and velocity at 1 Hz. The initialization module is executed at different starting time along this sequence. In each case, we evaluate the initial velocity and gravity vector, which are two important parameters affecting the initialization. For the ground-truth required in the evaluation, “urban1” provides the ground-truth velocity in the IMU body frame. After each initialization, we transform the velocity of the estimated IMU states into their body frame to compare with the ground-truth directly. For the ground-truth gravity vector, it is not provided directly from the dataset, so we compute the average IMU acceleration in the first one second of the sequence as the ground-truth gravity vector. We confirm the IMU is stationary in this period by the ground-truth velocity and observation of the camera images (the IMU and camera are in rigid connection). The relative scan poses and bias computed from the initialization are not evaluated. The reason is that the ground-truth relative scan poses are unavailable due to the low frequency of the ground-truth (1 Hz), and the ground-truth IMU bias is also inaccessible in the real world. Besides starting from different time in the sequence, we also explore the impact of different sliding windows (5, 10, 20, and 40) on the performance of the initialization module.

The point cloud map from one of the initialization experiments in a mixed structured and unstructured environment is

TABLE I
ESTIMATION ERROR (RMSE) AND AVERAGE COMPUTATION TIME FOR DIFFERENT WINDOW SIZES IN THE INITIALIZATION

Window size (N)	5	10	20	40
Velocity error (m/s)	0.1449	0.1247	0.1239	0.1233
Gravity error (m/s ²)	0.4636	0.3057	0.2801	0.2748
Computation time (s)	0.1423	0.2910	0.6938	2.1138

shown in Fig. 7. The initial velocity in this case is 9.6 m/s. Fig. 7(b, d, f) show the point map obtained by LIO, which serves as the input to the BA optimization in the initialization (Section V). Fig. 7(c, e, g) show the refined point cloud after the coarse-to-fine voxelization and BA optimization of the initialization. We can see the point clouds on the tree trunks and walls have greatly improved consistency, which proves the effectiveness of the initialization.

The comparison between the ground-truth and estimated initial velocity and gravity vector is shown in Fig. 8 and Table I. We can see the velocity is well estimated for all sizes of sliding windows ranging from 5 to 40, regardless of the time the initialization starts executing and even at very high initial speeds (i.e., more than 10 m/s). For the gravity vector, the estimation errors decrease with the sliding window size. This is because a larger sliding window size exploits longer IMU data, thereby increasing the information. These results suggest that our initialization is well able to initialize the system state and initial local map, even at high initial speeds, given a sufficient sliding window size. Considering that the estimation errors of velocity and gravity vector do not decrease noticeably after the sliding window is larger than 10, but a larger sliding window size dramatically increases the computation time, we fix $N = 10$ in Voxel-SLAM.

B. Single-Session SLAM

In this section, we compare the proposed system Voxel-SLAM against other state-of-the-art open-source LiDAR-inertial odometry, including LINS [27], FAST-LIO2 [13], Faster-LIO [14], Point-LIO [15], and full SLAM systems with loop closure, including LeGO-LOAM [21], LiLi-OM [24], LIO-SAM [25], LTA-OM [10] (FAST-LIO2 with loop closure). All the comparative methods use their default parameters unless explicitly stated. When evaluating the accuracy of odometry, we disable the loop closure (LC) for all SLAM methods mentioned above. Specially, we perform ablation studies of our system with odometry (Odom), local mapping (LM), loop closure (LC), and global mapping (GM). Our system with relevant modules is denoted as “Our (Odom)”, “Our (Odom + LM)”, “Our (Odom + LM + LC)”, and “Our (Full)”, respectively.

1) *Hilti handheld dataset:* Table II lists the absolute trajectory error (ATE) for all methods in odometry and SLAM. The main difficulty of the Hilti dataset is crossing the narrow staircases, which is a great challenge for LiDAR odometry due to the limited feature constraints and the potential occurrence of degeneration (e.g., sequences 03, 05, 07, and 13), as shown in the red frame of Fig. 9. The LeGO-LOAM, LiLi-OM, LINS,

TABLE II
ABSOLUTE TRAJECTORY ERROR (RMSE, CENTIMETERS) FOR DIFFERENT ODOMETRY AND SLAM METHODS IN HILTI

Sequences (cm)	hilti01	hilti02	hilti03	hilti04	hilti05	hilti06	hilti07	hilti08	hilti09	hilti10	hilti11	hilti12	hilti13
<i>(odometry without LC)</i>													
LeGO-LOAM	9.1	47	-	25.3	-	67	-	25.3	12.7	14.3	27.1	19.7	-
LiLi-OM	6.2	22.2	-	31	-	28.9	-	20.3	6.9	8.5	19.9	25.3	-
LINS	6.5	18.8	-	20.7	-	23.1	-	17.8	7.5	9.9	28.1	20.0	-
LIO-SAM	7.4	15.2	-	23.4	-	17.4	-	22.4	6.6	6.8	17.6	16.8	74
FAST-LIO2	1.3	2.8	32	6.7	55	2.4	72	1.7	2.4	1.8	4.2	3.5	16
Faster-LIO	1.1	2.1	37	5.0	73	1.4	61	2.4	1.9	2.3	2.7	2.6	11.4
Point-LIO	1.1	3.0	23	3.7	44	0.9	45	2.6	3.2	1.6	3.6	4.0	9.2
Our (Odom)	1.3	2.5	9.3	4.2	23	1.6	15.7	1.8	1.6	2.0	2.8	2.4	4.3
Our (Odom+LM)	0.8	1.8	3.0	3.4	15.9	0.9	9.8	1.2	1.25	1.4	2.4	1.4	1.26
<i>(full SLAM with LC)</i>													
LeGO-LOAM	8.8	39	-	25.3	-	67	-	22.7	12.6	12.9	27.1	16.2	-
LiLi-OM	6.2	14.2	-	31	-	27.1	-	18.6	6.9	8.0	19.9	22.8	-
LIO-SAM	6.1	10.1	-	23.4	-	13.4	-	17.2	6.6	5.5	17.6	12.5	74
LTA-OM	1.27	2.5	33	6.7	40	2.0	65	1.2	2.4	1.78	4.1	2.9	16
Our (Odom+LM+LC)	0.78	1.8	2.8	3.4	14.2	0.9	9.2	0.90	1.25	1.4	2.4	1.4	1.26
Our (Full)	0.62	1.4	2.9	3.3	13.8	0.7	7.8	0.82	1.00	1.14	1.30	1.22	0.80

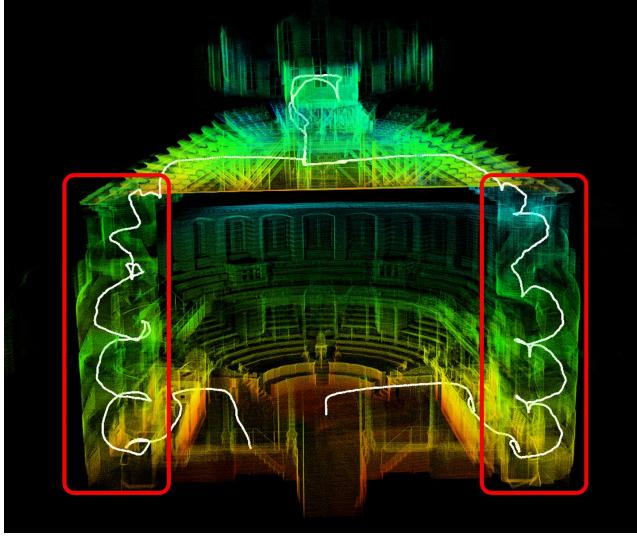


Fig. 9. The point cloud map of sequence “hilti05”, which is built by our full Voxel-SLAM system. The red frame shows the trajectory with point cloud map in the narrow staircase.

and LIO-SAM failed in these challenging sequences. The series of “LIO” methods, including FAST-LIO2, Faster-LIO, and Point-LIO, demonstrated more robustness in handling the staircases but still had unusually large pose errors. Our odometry with a voxel map mitigates this issue by employing adaptive resolution, leading to much smaller pose errors in these scenarios. Additionally, by incorporating local mapping, our odometry improves the accuracy of staircases significantly and achieves the highest accuracy among all sequences. The main reason is the exploitation of a longer data association for estimation compared to the short-term data association in scan-to-map registration used by all previous LIO methods.

The ATE of all SLAM methods with LC are shown in Table II. In general, all SLAM methods achieve improved accuracy than their corresponding odometry due to the consideration of additional loop closure constraints. Comparing among all SLAM methods, our method, even without the

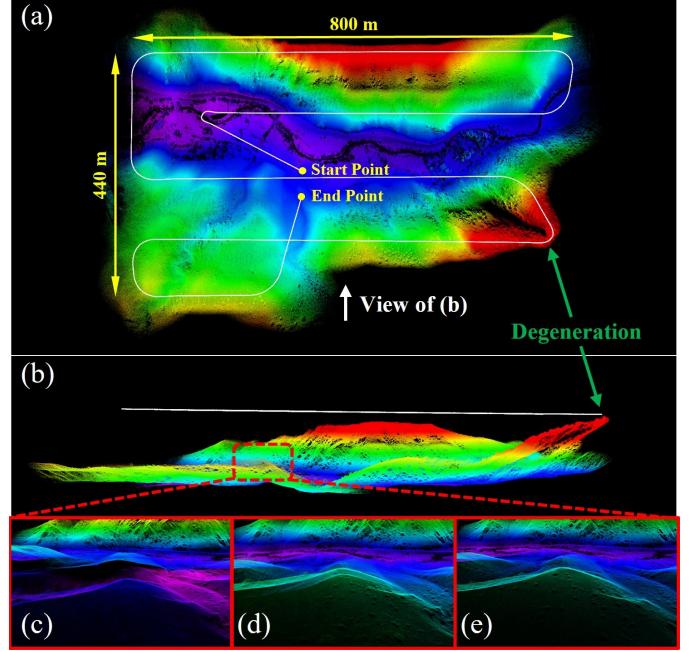


Fig. 10. The overview of “mars4-3”. (a)-(b) The global point cloud map in the top view and front view. The point map is built by our full Voxel-SLAM system. (c)-(e) The local map built by FAST-LIO2, our odometry with local mapping, and our full system with global mapping, respectively, for area in the red dashed frame of (a).

global mapping, has achieved the highest accuracy among all sequences. With global mapping, our method further reduces all trajectory errors. These accuracy improvements confirm that the global mapping yields improved optimization results across all sequences due to the exploitation of long-term data association. Even in cases where no loop retrieval is detected (e.g., sequences 04, 09, 10, 11, 13), the global mapping can still improve the accuracy, whereas other methods lack a means to further refine their estimates.

2) MARS-LVIG aerial mapping dataset: The MARS-LVIG dataset [51] has a unique aerial downward-looking viewpoint

TABLE III
ABSOLUTE TRAJECTORY ERROR (RMSE, METERS) FOR DIFFERENT ODOMETRY AND SLAM METHODS IN MARS-LVIG

Sequences (m)	mars1-1	mars1-2	mars1-3	mars2-1	mars2-2	mars2-3	mars3-1	mars3-2	mars3-3	mars4-1	mars4-2	mars4-3
<i>(odometry without LC)</i>												
LiLi-OM	4.56	4.81	5.35	3.68	3.72	3.70	10.54	11.66	13.08	-	-	-
LIO-SAM	3.77	4.02	4.79	1.22	1.30	1.39	6.89	6.95	8.62	12.09	11.53	14.64
FAST-LIO2	0.66	0.46	0.48	0.26	0.39	0.59	2.17	2.05	2.51	4.46	6.54	8.37
Faster-LIO	0.42	0.56	0.51	0.27	0.36	0.32	1.71	1.49	3.12	5.50	7.43	8.79
Point-LIO	1.53	1.44	1.47	0.35	0.40	0.69	3.63	3.38	4.23	8.92	9.57	12.45
Our (Odom)	0.45	0.42	0.50	0.24	0.42	0.48	2.25	1.39	1.90	4.21	5.33	7.68
Our (Odom+LM)	0.23	0.35	0.32	0.16	0.25	0.33	1.18	1.17	1.29	2.49	2.87	3.28
<i>(full SLAM with LC)</i>												
LiLi-OM	3.83	4.04	4.10	2.87	2.93	2.85	8.47	8.86	11.55	-	-	-
LIO-SAM	2.90	2.91	3.39	0.55	0.67	0.70	6.36	5.73	7.27	12.09	11.53	14.64
LTA-OM	0.49	0.42	0.39	0.22	0.31	0.40	1.09	1.34	1.47	4.46	6.54	8.37
Our (Odom+LM+LC)	0.22	0.25	0.28	0.16	0.24	0.30	0.98	0.99	1.05	2.49	2.87	3.28
Our (Full)	0.18	0.22	0.26	0.15	0.22	0.29	0.84	0.76	0.70	0.51	0.55	0.77

TABLE IV
ABSOLUTE TRAJECTORY ERROR (RMSE, METERS) FOR ODOMETRY AND SLAM OF DIFFERENT METHODS IN URBANNAV

Sequences (m)	urban1	urban2	urban3	urban4	urban5
<i>(odometry without LC)</i>					
LeGO-LOAM	17.93	15.94	11.87	8.53	4.92
LIO-SAM	10.30	11.75	5.90	5.66	1.72
LiLi-OM	11.14	108.22	6.23	7.14	4.12
LINS	12.93	75.93	17.18	9.91	5.21
FAST-LIO2	9.08	7.20	5.22	3.11	1.22
Faster-LIO	9.59	8.14	5.25	3.23	1.62
Point-LIO	8.14	10.53	3.46	3.75	1.18
Our (Odom)	7.83	9.61	4.23	3.30	1.14
Our (Odom+LM)	5.62	7.13	3.45	2.98	1.04
<i>(full SLAM with LC)</i>					
LeGO-LOAM	12.54	10.68	7.23	6.43	2.87
LIO-SAM	8.44	8.87	3.42	3.01	1.17
LiLi-OM	9.43	108.01	4.76	6.52	2.79
LTA-OM	5.28	5.51	2.92	2.73	0.93
Our (Odom+LM+LC)	3.02	4.77	2.15	2.58	0.92
Our (Full)	2.82	4.25	1.74	2.42	0.90

from about 100 m above the ground. LeGO-LOAM and LINS cannot run this dataset since their special feature extraction is not applicable to the Livox AVIA LiDAR. We follow the instructions from LIO-SAM to adapt it to the LiDAR, and the rest methods can work on Livox AVIA LiDAR naturally without modification. The ATE results of all the methods on this dataset are listed in Table III. Sequence names sharing the same first number have the same paths but at different flight speeds. Different from the structured and small environment and slow movement in the Hilti dataset, the MARS-LVIG flies fast with the highest speed of 12 m/s in a large scene with massive natural scenes of woods, hills, and rivers, as shown in Fig. 10(a). Our method is still robust in these unstructured environments with fast motion and achieves the best accuracy in odometry and SLAM methods in most sequences.

Specially, the fourth group of sequences, “mars4-1”, “mars4-2”, and “mars4-3”, presents the most challenging scenarios. A short period of degeneration occurs when the drone flies to the hilltop shown in Fig. 10(a)-(b). Although most odometry methods except LiLi-OM survived, their trajectories become tilted after the hilltop, with ATEs significantly larger

than other sequences in Table III. Due to the degeneration, the point cloud maps of these odometry methods exhibit inconsistencies, as shown in Fig. 10(c)-(d), where points on the hilltop separate into two layers. As can be seen, the separation by our odometry with local mapping (Fig. 10(d)) is notably smaller than that of the representative LIO method, FAST-LIO2 (Fig. 10(c)), due to the use of mid-term data associations. Accordingly, the quantitative ATEs of our odometry with local mapping, as shown in Table III, are much better than other LIO methods. For the full SLAM, all the methods fail to identify valid loop detections in these three sequences, leading to ATEs identical to their corresponding odometry. However, with the global mapping, our method can still improve the trajectory accuracy through the coarse-to-fine voxelization and BA optimization (Section IX). The coarse-to-fine voxelization voxelizes the separated layers of point clouds into the same voxel, hence associating them into one feature that is optimized in the subsequent BA optimization. The resultant map consistency at this location is greatly improved, as shown in Fig. 10(e). Accordingly, the ATEs of our full method with global mapping in these three sequences are also improved noticeably, as shown in Table III.

3) *UrbanNav robotcar urban dataset*: Autonomous driving represents a significant application domain for LiDAR SLAM, and we evaluated our method on the UrbanNav robotcar dataset. The UrbanNav dataset is collected in urban scenarios with a higher speed (maximum speed of 13 m/s), longer collection time (the longest 56 minutes), more dynamic objects, and previously revisited places than the Hilti. The results in Table IV demonstrate the effectiveness of our method in urban scenarios, where it achieves the highest accuracy among all sequences and all evaluated methods in both cases of odometry and full SLAM.

C. Multi-Session SLAM

The sequences from Hilti 09-13 are collected at the same construction site, and their ground truth shares the common world frame. We use these sequences to test the multi-session localization ability of our Voxel-SLAM. Firstly, Voxel-SLAM is fed with the sequences “hilti13”, “hilti12”, “hilti11”,

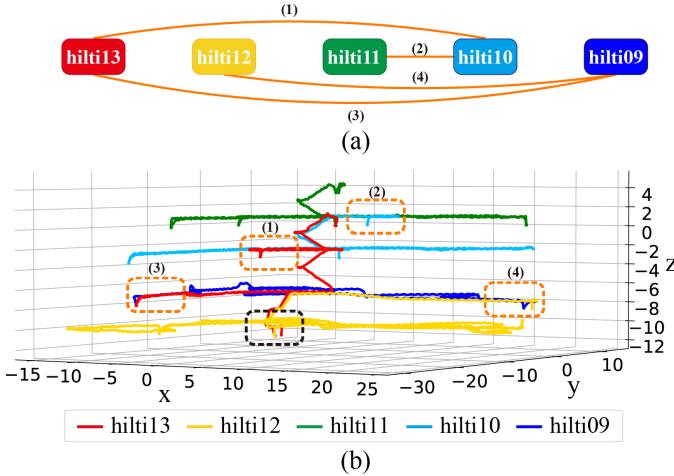


Fig. 11. (a) The connectivity among the 5 sequences of Hilti 09 - 13. The numbers on the connected edges represent the order of the detected loops. (b) The trajectories of the 5 sequences, estimated by the full Voxel-SLAM with global mapping. The orange dashed frames show detected loops in (a). The black dashed frame shows a place overlapped by sequence “hilti13” and “hilti12” but is not detected as a loop due to the missing detection.

TABLE V
ATE (RMSE, CENTIMETERS) FOR EACH INDIVIDUAL SESSION AND FULL MULTI-SESSION, ESTIMATED BY POSE GRAPH OPTIMIZATION (PGO) ONLY OR WITH ADDITIONAL GLOBAL MAPPING

	hilti09	hilti10	hilti11	hilti12	hilti13	Multi-session
PGO only	1.00	1.14	1.30	1.22	0.80	7.6
Global mapping	0.60	0.91	1.13	1.16	0.80	4.9

“hilti10”, and “hilti09”, sequentially. After completing each sequence, Voxel-SLAM saves it in memory as a previous session, enabling the system to retrieve loop detections among these previous sessions in future sequences. Running through all five sequences, the Voxel-SLAM builds a connectivity graph as shown in Fig. 11(a), which successfully detects loops at revisited areas among different sequences, such as the areas labeled as (1-4) in the figure. This leads to a connected pose graph, from which the poses of all five sequences can be optimized jointly by PGO and then by global mapping. The estimated pose trajectory from Voxel-SLAM is shown in Fig. 11(b).

Despite the accuracy and robustness of the BTC descriptors [18] used in our Voxel-SLAM, there could still be missing loop detection (e.g., the black dashed frame in Fig. 11(b)), which is also marked in the white dashed frame of Fig. 12(b)). This missing detection occurred in a narrow corridor, which is very challenging for the LiDAR-based place recognition methods, causing insufficient constraints between the “hilti13” and “hilti12” and hence limiting the accuracy achievable by the PGO. The corresponding point cloud in this area also exhibits clear inconsistency, as shown in Fig. 12(c). This issue is effectively addressed by global mapping, as shown in Fig. 12(d). The entire point cloud maps refined by global mapping are shown in Fig. 12(a, b), colored by different sequences. Table V shows the ATEs of the trajectories estimated by Voxel-SLAM with PGO only or with additional global mapping.

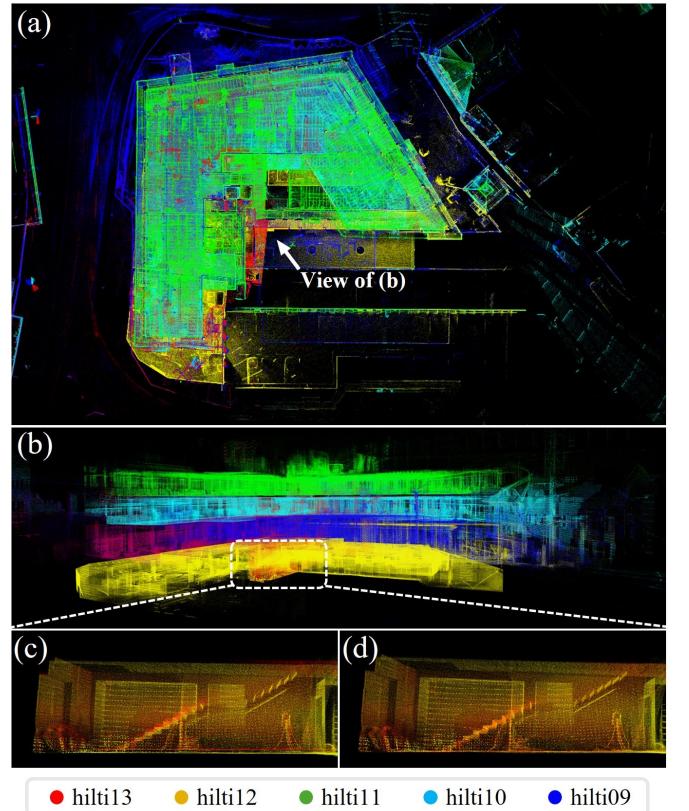


Fig. 12. (a)-(b) The global point cloud map of the five sessions from the top view and side view. (c)-(d) The zoomed-in point cloud of the area in the white dashed frame of (b), which is also area in the black dashed frame of Fig. 11(b), obtained from PGO and global mapping, respectively.

The application of global mapping not only enhances the multi-session accuracy of all five sequences but also improves the accuracy for each individual session, due to the full exploitation of multi-map data association.

The above multi-session experiment further proves the performance enhancements, especially in the map consistencies, brought by global mapping. In the above analysis, we did not compare the results of LTA-OM [10]. Although LTA-OM has multi-session SLAM capability, it only supports finding loop closures in one previous session and cannot handle many previous sessions simultaneously. In addition, it fails to find the loop closure between “hilti11” and “hilti10”.

D. Relocalization

This experiment evaluates the relocalization capability of the system after encountering a degeneration. Moreover, considering the application in mobile robots with limited computation resources, we evaluate the system on the onboard computer equipped with an Intel i3-N305 CPU. The used sequence “private2” was collected using the handheld device. The experiment tests all functions of Voxel-SLAM, encompassing initialization (and re-initialization) in the middle of data collection with non-zero state, odometry, local mapping, loop closure detection across multiple sessions, relocalization, and global mapping. All of these functionalities were executed online on the onboard computer.

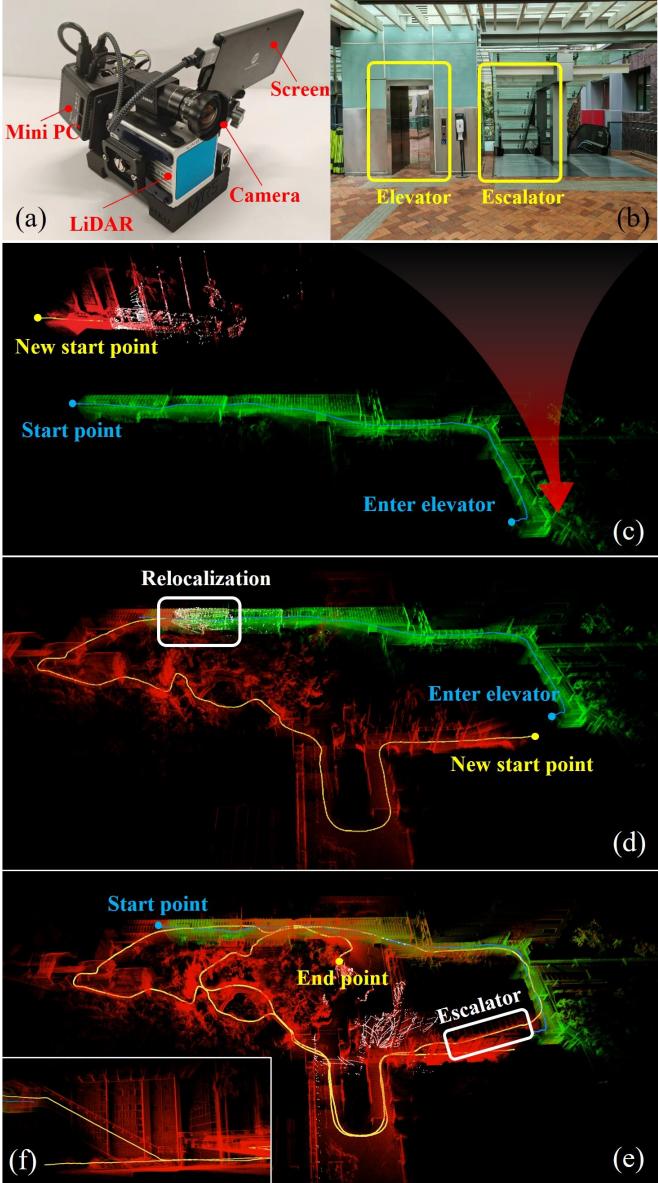


Fig. 13. The sequence “private2”. (a) The handheld equipment. (b) The elevator and escalator travelled by the sequence. The elevator will cause a system divergence due to the confined spaces and inconsistent measurements between LiDAR and IMU. (c) The maps and trajectories after detecting divergence in the first session (the green one) and successful re-initialization of the second session (the red one). (d) The maps and trajectories after relocalization of the second session (red) in the first session (green). (e) The maps and trajectories at the end of data collection. (f) The zoomed-in map of the escalator.

As shown in Fig. 13, the sequence is collected in a campus environment with many moving pedestrians, significant variations in height, confined spaces (e.g., inside elevators) causing degeneration and motion ambiguity, and a mix of structured and unstructured surroundings. The collecting trajectory is shown in Fig. 14 from start to end in order of the waypoints A-G. Specifically, after taking off from the start point, we walk along the blue path, which renders the green point cloud in Fig. 13(c), until entering the elevator (waypoint A), as shown in Fig. 13(b). Due to the confined space in the elevator and the inconsistency between LiDAR and IMU measurements, Voxel-

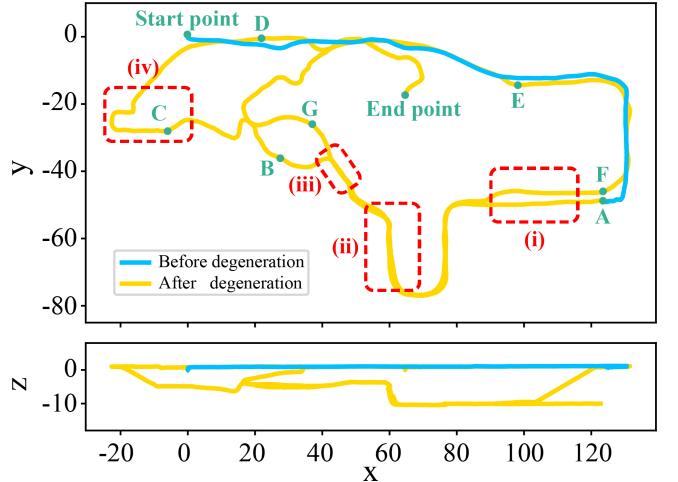


Fig. 14. The complete trajectories of data collection from top view and front view. We mark some waypoints from A to G for readers understanding the trajectories better. The sequence begin at start point and walk in the order of the waypoints until to the end point. The red dashed frames are the places whose point clouds will be displayed in Fig. 15.

SLAM detects the system divergence and starts a new session. The new session initialization keeps running but fails until we exit the elevator, where the successfully initialized new local map is shown as the red point cloud in Fig. 13(c). After exiting the elevator, we pass a small park with woods and vegetation (waypoint B) and a narrow staircase (waypoint C), arriving at waypoint D near the start point of the first session. Revisiting the start point causes Voxel-SLAM to detect loops with the first session and a subsequent relocalization that connects the second session to the first session, as shown in Fig. 13(d). We continue to walk along the previous path (passing through waypoint E), where long-term associations with the first session are consistently exploited by Voxel-SLAM due to its loop closures and keyframe loading in local mapping (Section VII-A). Then, we return to the elevator (waypoint F) that caused the system divergence in the first session, but this time we choose the escalator on the right (Fig. 13(b)). Traveling on the escalator does not cause a degeneration, so the current session proceeds smoothly, as shown in Fig. 13(e). After several loop closures in the current and previous sessions ($F \rightarrow G \rightarrow$ end point), the data collection ends, which triggers the global mapping to optimize the entire point cloud map.

The online relocalization experiment proves the robustness and reliability of Voxel-SLAM. When entering the elevator, the odometry detects the system divergence and restarts the system. The initialization re-initializes constantly in the elevator until we exit the elevator. When returning to previously visited places, Voxel-SLAM can relocalize the current session into the world frame of the previous session. After the relocalization, the long-term and multi-map data associations, including the keyframe loading and loop closure among multiple sessions, are fully exploited to ensure map consistency. The complete trajectories of two sessions are shown in Fig. 14. As can be seen, the two trajectories are consistent on z-axis. The global point cloud map of the whole sequence is shown in Fig. 15

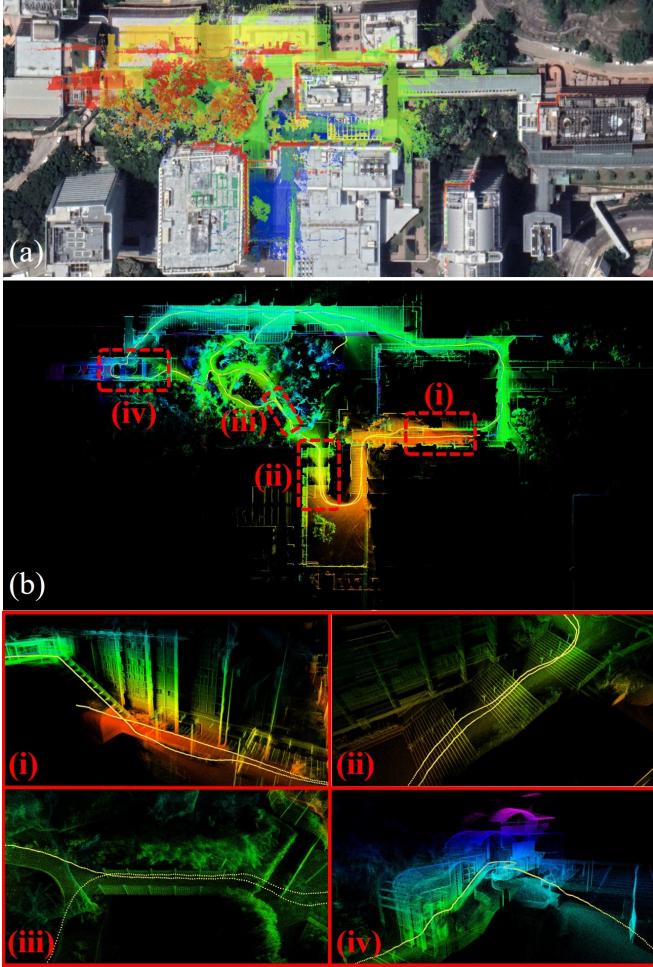


Fig. 15. (a) The map aligned with Google Earth. (b) The point cloud map optimized by our system. (i)-(iv) The zoomed-in maps from the red dashed frames in (b).

(b), and the map is aligned well with Google Earth³ shown in Fig. 15(a). Fig. 15(i-iv) shows the point cloud maps from the multiple revisited places by one or multiple sessions. The points across different sessions are consistent due to the full exploitation of long-term and multi-map data associations in the local mapping, loop closure, and global mapping modules.

E. Computation Time

For all the above datasets and experiments, we evaluate the average time and total memory consumption. Note that the LiDAR scans in all the sequences are all at 10 Hz. To examine the real-time capability of our system, we compute the average computation time of each recurrent modules (odometry, local mapping, loop closure, and keyframe BA) distributed to each scan. For the odometry, and local mapping, they naturally run at the LiDAR scan rate (i.e., 10 Hz), so the time is their actual computation time. For loop closure and keyframe BA, they are triggered by events (e.g., the selection of a key frame or detection of a loop frame), so we accumulate their total computation time over the whole sequence and divide the

accumulated time by the number of LiDAR scans to obtain the per scan average time. Since the odometry & local mapping, loop closure, and keyframe BA are executed in three parallel threads, the average time consumption of each thread should be smaller than the time interval between two consecutive LiDAR scans ($t_s = 0.1\text{s}$) to ensure the real-time operation.

Table VI shows the time and memory consumption of all sequences in the previous experiments. As can be seen, the time consumption of the three working threads, individually or even in total, is way smaller than the scan interval t_s , proving the real-time performance of the Voxel-SLAM even on the onboard computer with constrained computation resources. For the initialization and global mapping modules in Voxel-SLAM, they are generally executed only once prior to or after each session (or sequence), so we evaluate their average time consumption per running. As shown in Table VI, the time consumption of global mapping after session end is much less than (ranging from 0.6% to 4.7%) the sequence duration (i.e., data collection time), proving the efficiency of the designed global mapping module. Specially, in the longest sequence, “urban3”, the global mapping after session end takes 79.7 seconds, versus the data collection duration of 56 minutes and trajectory of 4.86 kilometers.

Memory usage is another important metric for a system exploiting long-term data association, since the system should store all the necessary information in memory to retrieve the long-term data association. As shown in Table VI, the memory usage is way below the physical memory, even on the onboard computer with constrained resources.

XI. CONCLUSION AND FUTURE WORKS

In this paper, we presented the Voxel-SLAM: a complete, accurate, and versatile LiDAR-inertial SLAM system, including the modules of initialization, odometry, local mapping, loop closure, and global mapping, all of which employ the same adaptive voxel map structure. The initialization exhibits remarkable speed and robustness, providing accurate states and a consistent map for the following modules. The odometry swiftly estimates the current state while perceiving potential system divergence. The local mapping utilizes an efficient, tightly-coupled LiDAR-inertial BA to refine the states and map simultaneously, thereby enhancing accuracy and robustness. The loop closure can detect revisited places in multiple sessions. An efficient and accurate global mapping is devised with a data pyramid. Collectively, these modules make full use of four types of data association: short-term, middle-term, long-term, and multi-map.

The system could be extended to fuse image measurements, which can bolster robustness in some degeneration environments, provide color information for point clouds, and enhance place recognition performance. Moreover, although the current system is efficient enough to run in real-time on an onboard computer, GPU parallelization can further accelerate the system efficiency, especially for global mapping. The framework of voxel map is very suitable for parallel operations.

³<https://earth.google.com/>

TABLE VI
THE MEAN AND STANDARD ERROR OF TIME AND MEMORY CONSUMPTION FOR ALL SEQUENCES IN THE EXPERIMENTS

	Hilti	MARS-LVIG	UrbanNav	Private2 (onboard)
<i>(Per scan)[#]</i>				
Odometry (sec)	0.008 / 0.003	0.023 / 0.007	0.013 / 0.002	0.008 / 0.002
Local mapping (sec)	0.013 / 0.004	0.032 / 0.010	0.024 / 0.005	0.019 / 0.008
Loop closure (sec)	0.004 / 0.002	0.031 / 0.016	0.010 / 0.002	0.007 / 0.000*
Keyframe BA (sec)	0.002 / 0.001	0.009 / 0.003	0.004 / 0.001	0.004 / 0.000*
Total (sec)	0.027 / 0.009	0.095 / 0.028	0.051 / 0.006	0.037 / 0.000*
<i>(Per running)[#]</i>				
Initialization (sec)	0.087 / 0.040	0.453 / 0.102	0.207 / 0.023	0.148 / 0.019
Global mapping after session end (sec)	3.157 / 2.895	20.74 / 16.36	34.20 / 26.24	8.819 / 8.124
Sequence duration (sec)	238.6 / 100.9	556.9 / 290.2	1295 / 1118	694.0 / 0.000*
Total memory usage (GB)	1.068 / 0.510	5.203 / 2.553	4.535 / 1.830	2.932 / 0.000*

* The standard error is zero since only one sequence is evaluated.

The average time consumption per scan and per running means distributing the total time consumption to each scan and each running, respectively.

XII. ACKNOWLEDGEMENT

The authors gratefully acknowledge DJI and Livox Technology for fund and equipment support during the development.

APPENDIX

A. Iterative Optimization for LiDAR-Inertial BA

The cost function of the LiDAR-inertial BA is,

$$\begin{aligned} & \arg \min_{\mathcal{X}} c(\mathcal{X}) \\ &= \arg \min_{\mathcal{X}} \left(\frac{1}{2} \sum_{i=1}^{N-1} \|\mathbf{r}_{i,i+1}(\mathcal{X})\|_{\Sigma_{i,i+1}^{-1}}^2 + \sum_{j=1}^M \lambda_j^{\min}(\mathcal{X}) \right) \quad (5) \end{aligned}$$

where the definition of each item follows (2). Given a perturbation $\delta\mathcal{X} = [\delta\mathbf{x}_1, \delta\mathbf{x}_2, \dots, \delta\mathbf{x}_N, \delta\mathbf{g}]$ to state \mathcal{X} ,

$$c(\mathcal{X} \boxplus \delta\mathcal{X}) = \frac{1}{2} \sum_{i=1}^{N-1} \|\mathbf{r}_{i,i+1}(\mathcal{X} \boxplus \delta\mathcal{X})\|_{\Sigma_{i,i+1}^{-1}}^2 + \sum_{j=1}^M \lambda_j^{\min}(\mathcal{X} \boxplus \delta\mathcal{X}) \quad (6)$$

Expand the residual to the highest second-order approximation,

$$\begin{aligned} c(\mathcal{X} \boxplus \delta\mathcal{X}) &\approx \sum_{j=1}^M (\lambda_j^{\min}(\mathcal{X}) + \mathbf{g}_j \delta\mathcal{X} + \frac{1}{2} \delta\mathcal{X}^T \mathbf{H}_j \delta\mathcal{X}) \\ &+ \frac{1}{2} \left(\sum_{i=1}^{N-1} \left(\|\mathbf{r}_{i,i+1}(\mathcal{X})\|_{\Sigma_{i,i+1}^{-1}}^2 + 2\mathbf{r}_{i,i+1}(\mathcal{X})^T \Sigma_{i,i+1}^{-1} \mathbf{J}_i \delta\mathcal{X} \right. \right. \\ &\quad \left. \left. + \delta\mathcal{X}^T \mathbf{J}_i^T \Sigma_{i,i+1}^{-1} \mathbf{J}_i \delta\mathcal{X} \right) \right) \quad (7) \\ &= \sum_{j=1}^M \lambda_j^{\min}(\mathcal{X}) + \frac{1}{2} \sum_{i=1}^{N-1} \|\mathbf{r}_{i,i+1}(\mathcal{X})\|_{\Sigma_{i,i+1}^{-1}}^2 \\ &+ \left(\sum_{j=1}^M \mathbf{g}_j + \sum_{i=1}^{N-1} \mathbf{r}_{i,i+1}(\mathcal{X})^T \Sigma_{i,i+1}^{-1} \mathbf{J}_i \right) \delta\mathcal{X} \\ &+ \frac{1}{2} \delta\mathcal{X}^T \left(\sum_{j=1}^M \mathbf{H}_j + \sum_{i=1}^{N-1} \mathbf{J}_i^T \Sigma_{i,i+1}^{-1} \mathbf{J}_i \right) \delta\mathcal{X} \quad (8) \end{aligned}$$

where \mathbf{g}_j and \mathbf{H}_j are the gradient and Hessian matrix of LiDAR BA factor associated to the j -th plane feature, with exact form drawn from [16]. \mathbf{J}_i is the Jacobian of IMU preintegration residual $\mathbf{r}_{i,i+1}$, which consists of the derivative of $\mathbf{r}_{i,i+1}$ with respect to the state $\delta\mathbf{x}_i$ and that to the gravity vector $\delta\mathbf{g}$. The former is identical to [26] and the latter is:

$$\begin{aligned} \frac{\partial \mathbf{r}_{\Delta \mathbf{R}_{ij}}}{\partial \delta \mathbf{g}} &= \mathbf{0}_{3 \times 3} & \frac{\partial \mathbf{r}_{\Delta \mathbf{P}_{ij}}}{\partial \delta \mathbf{g}} &= -\frac{1}{2} \mathbf{R}_i^T \Delta t_{ij}^2 \\ \frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \delta \mathbf{g}} &= -\mathbf{R}_i^T \Delta t_{ij} & \frac{\partial \mathbf{r}_{\Delta \mathbf{b}_{ij}^g}}{\partial \delta \mathbf{g}} &= \frac{\partial \mathbf{r}_{\Delta \mathbf{b}_{ij}^a}}{\partial \delta \mathbf{g}} = \mathbf{0}_{3 \times 3} \quad (9) \end{aligned}$$

The quadratic function (8) attains its minimum when

$$\begin{aligned} & \left(\sum_{j=1}^M \mathbf{H}_j + \sum_{i=1}^{N-1} \mathbf{J}_i^T \Sigma_{i,i+1}^{-1} \mathbf{J}_i \right) \delta\mathcal{X}^* = \\ & - \left(\sum_{j=1}^M \mathbf{g}_j^T + \sum_{i=1}^{N-1} \mathbf{J}_i^T \Sigma_{i,i+1}^{-1} \mathbf{r}_{i,i+1}(\mathcal{X}) \right) \quad (10) \end{aligned}$$

which leads to the optimum solution $\delta\mathcal{X}^*$. In practice, we optimize the states iteratively within the LM framework until converging.

B. Criterion of Degeneration

The degeneration of a scene is determined by the distribution of independent plane constraints contained in the scene. If there are plane constraints in all three directions, the scene is considered non-degenerated. Otherwise, the scene is degenerating. In practice, the directional distribution of the plane features contained in a scene can be examined from the eigenvalues of the matrix,

$$\mathbf{M} = \sum \mathbf{n}_i \mathbf{n}_i^T \quad (11)$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{n}_i \in \mathbb{S}^2$ is the normal vector of the i -th plane feature in the scene. The three eigenvalues of the matrix \mathbf{M} indicate the sufficiency of the constraints along each eigenvector direction. Consequently, we only need to look at the minimum eigenvalue: if the minimum eigenvalue is larger than the predetermined threshold, it signifies that there are enough plane constraints even in the most insufficient

TABLE VII
DETAILS OF ALL THE SEQUENCES FOR EXPERIMENTS

Sequence	Name	Duration (min:sec)	Distance (km)
hilti01	exp01-construction	3:47	0.16
hilti02	exp02-construction	7:10	0.31
hilti03	exp03-construction	5:09	0.23
hilti04	exp07-long-corridor	2:12	0.11
hilti05	exp09-cupola	7:26	0.19
hilti06	exp11-lower-gallery	2:31	0.08
hilti07	exp15-upper-gallery	4:20	0.13
hilti08	exp21-outside	2:32	0.14
hilti09	site1-handheld-1	3:24	0.17
hilti10	site1-handheld-2	2:47	0.15
hilti11	site1-handheld-3	2:50	0.15
hilti12	site1-handheld-4	4:55	0.28
hilti13	site1-handheld-5	2:39	0.14
mars1-1	HKisland01	9:40	1.85
mars1-2	HKisland02	5:10	1.85
mars1-3	HKisland03	3:46	1.85
mars2-1	HKairport01	10:50	2.04
mars2-2	HKairport02	5:40	2.04
mars2-3	HKairport03	4:00	2.04
mars3-1	AMtown01	20:00	5.11
mars3-2	AMtown02	10:10	5.11
mars3-3	AMtown03	7:10	5.11
mars4-1	AMvalley01	17:00	4.30
mars4-2	AMvalley02	8:45	4.30
mars4-3	AMvalley03	6:00	4.30
urban1	UrbanNav-Medium	13:05	3.64
urban2	UrbanNav-Deep	25:36	4.51
urban3	UrbanNav-Harsh	56:07	4.86
urban4	2019-04-28-20-58-02	8:07	2.01
urban5	2020-03-14-16-45-35	5:00	1.21
private1	Jungle-challenge	2:00	0.06
private2	Campus-elevator	11:34	0.85
Total		281:22	59.28

direction, indicating the scene is not degenerate; otherwise, the scene is considered degenerated.

C. Detail information of all sequences

The detail information about total 32 sequences tested in Section X are listed in Table VII.

REFERENCES

- [1] Y. Li and J. Ibanez-Guzman, “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.
- [2] F. Kong, W. Xu, Y. Cai, and F. Zhang, “Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.
- [3] N. Chen, F. Kong, W. Xu, Y. Cai, H. Li, D. He, Y. Qin, and F. Zhang, “A self-rotating, single-actuated uav with extended sensor field of view for autonomous navigation,” *Science Robotics*, vol. 8, no. 76, p. eade4538, 2023.
- [4] D. Wang, C. Watkins, and H. Xie, “Mems mirrors for lidar: A review,” *Micromachines*, vol. 11, no. 5, p. 456, 2020.
- [5] L. Zhang, D. Chitnis, H. Chun, S. Rajbhandari, G. Faulkner, D. O’Brien, and S. Collins, “A comparison of apd-and spad-based receivers for visible light communications,” *Journal of Lightwave Technology*, vol. 36, no. 12, pp. 2435–2442, 2018.
- [6] Z. Liu, F. Zhang, and X. Hong, “Low-cost retina-like robotic lidars based on incommensurable scanning,” *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 1, pp. 58–68, 2021.
- [7] J. Lin and F. Zhang, “Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.
- [8] 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.
- [9] J. Lin and F. Zhang, “A fast, complete, point cloud based loop closure for lidar odometry and mapping,” *arXiv preprint arXiv:1909.11811*, 2019.
- [10] Z. Zou, C. Yuan, W. Xu, H. Li, S. Zhou, K. Xue, and F. Zhang, “Ltatom: Long-term association lidar-imu odometry and mapping,” *Journal of Field Robotics*, 2024.
- [11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [13] 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.
- [14] 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 Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022.
- [15] 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*, vol. 5, no. 7, p. 2200459, 2023.
- [16] Z. Liu, X. Liu, and F. Zhang, “Efficient and consistent bundle adjustment on lidar point clouds,” *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4366–4386, 2023.
- [17] X. Liu, Z. Liu, F. Kong, and F. Zhang, “Large-scale lidar consistent mapping using hierarchical lidar bundle adjustment,” *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1523–1530, 2023.
- [18] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, “Btc: A binary and triangle combined descriptor for 3d place recognition,” *IEEE Transactions on Robotics*, vol. 40, pp. 1580–1599, 2024.
- [19] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, “Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8518–8525, 2022.
- [20] 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.
- [21] T. Shan and B. Englot, “Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” *arXiv preprint arXiv:2007.00258*, 2020.
- [26] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual-inertial odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.
- [27] 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.
- [28] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, “Std: Stable triangle descriptor for 3d place recognition,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1897–1903.
- [29] R. Li, X. Zhang, S. Zhang, J. Yuan, H. Liu, and S. Wu, “Ba-liom: tightly coupled laser-inertial odometry and mapping with bundle adjustment,” *Robotica*, pp. 1–17, 2024.
- [30] H. Tang, T. Zhang, L. Wang, X. Niu *et al.*, “Ba-lins: A frame-to-frame bundle adjustment for lidar-inertial navigation,” *arXiv preprint arXiv:2401.11491*, 2024.
- [31] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, “Suma++: Efficient lidar-based semantic slam,” in *2019 IEEE/RSJ*

- International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.
- [32] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, “Lips: Lidar-inertial 3d plane slam,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 123–130.
- [33] G. Ferrer, “Eigen-factors: Plane estimation for multi-frame and time-continuous point cloud alignment.” in *IROS*, 2019, pp. 1278–1284.
- [34] L. Zhou, D. Koppel, and M. Kaess, “Lidar slam with plane adjustment for indoor environment,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7073–7080, 2021.
- [35] H. Huang, Y. Sun, J. Wu, J. Jiao, X. Hu, L. Zheng, L. Wang, and M. Liu, “On bundle adjustment for multiview point cloud registration,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8269–8276, 2021.
- [36] Z. Liu and F. Zhang, “Balm: Bundle adjustment for lidar mapping,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.
- [37] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, “Appearance-based loop detection from 3d laser data using the normal distributions transform,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 23–28.
- [38] L. He, X. Wang, and H. Zhang, “M2dp: A novel 3d point cloud descriptor and its application in loop closure detection,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 231–237.
- [39] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, “Segmatch: Segment based place recognition in 3d point clouds,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5266–5272.
- [40] S. Belongie, J. Malik, and J. Puzicha, “Shape context: A new descriptor for shape matching and object recognition,” *Advances in neural information processing systems*, vol. 13, 2000.
- [41] G. Kim and A. Kim, “Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4802–4809.
- [42] G. Kim, S. Choi, and A. Kim, “Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments,” *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1856–1874, 2021.
- [43] D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” *IEEE Transactions on robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [44] Y. Cui, X. Chen, Y. Zhang, J. Dong, Q. Wu, and F. Zhu, “Bow3d: Bag of words for real-time loop closing in 3d lidar slam,” *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2828–2835, 2022.
- [45] D. Cattaneo, M. Vaghi, and A. Valada, “Lcdnet: Deep loop closure detection and point cloud registration for lidar slam,” *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2074–2093, 2022.
- [46] K. Vidanapathirana, M. Ramezani, P. Moghadam, S. Sridharan, and C. Fookes, “Logg3d-net: Locally guided global descriptor learning for 3d place recognition,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2215–2221.
- [47] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, “Locnet: Global localization in 3d point clouds for mobile vehicles,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 728–733.
- [48] F. Dellaert, “Factor graphs and gtsam: A hands-on introduction,” *Georgia Institute of Technology, Tech. Rep*, vol. 2, p. 4, 2012.
- [49] L. Zhang, M. Helmberger, L. F. T. Fu, D. Wisth, M. Camurri, D. Scaramuzza, and M. Fallon, “Hilti-oxford dataset: A millimeter-accurate benchmark for simultaneous localization and mapping,” *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 408–415, 2022.
- [50] A. D. Nair, J. Kindle, P. Levchev, and D. Scaramuzza, “Hilti slam challenge 2023: Benchmarking single+ multi-session slam across sensor constellations in construction,” *arXiv preprint arXiv:2404.09765*, 2024.
- [51] H. Li, Y. Zou, N. Chen, J. Lin, X. Liu, W. Xu, C. Zheng, R. Li, D. He, F. Kong *et al.*, “Mars-lvig dataset: A multi-sensor aerial robots slam dataset for lidar-visual-inertial-gnss fusion,” *The International Journal of Robotics Research*, 2024.
- [52] L.-T. Hsu, F. Huang, H.-F. Ng, G. Zhang, Y. Zhong, X. Bai, and W. Wen, “Hong kong urbannav: An open-source multisensory dataset for benchmarking urban navigation algorithms,” *NAVIGATION: Journal of the Institute of Navigation*, vol. 70, no. 4, 2023.