GLIM: 3D Range-Inertial Localization and Mapping with GPU-Accelerated Scan Matching Factors

Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno

***Abstract*—This article presents GLIM, a 3D range-inertial localization and mapping framework with GPU-accelerated scan matching factors. The odometry estimation module of GLIM employs a combination of fixed-lag smoothing and keyframe-based point cloud matching that makes it possible to deal with a few seconds of completely degenerated range data while efficiently reducing trajectory estimation drift. It also incorporates multi-camera visual feature constraints in a tightly coupled way to further improve the stability and accuracy. The global trajectory optimization module directly minimizes the registration errors between submaps over the entire map. This approach enables us to accurately constrain the relative pose between submaps with a small overlap. Although both the odometry estimation and global trajectory optimization algorithms require much more computation than existing methods, we show that they can be run in real-time due to the careful design of the registration error evaluation algorithm and the entire system to fully leverage GPU parallel processing.**

***Index Terms*—Sensor fusion, Simultaneous localization and mapping (SLAM).**

I. Introduction

RANGE-based simultaneous localization and mapping (SLAM) has been actively studied due to the emergence of various range sensors, from highly accurate light detection and ranging (LiDAR) sensors to affordable depth cameras. It has become inevitable for wide variety of applications, including surveillance and autonomous driving.

Most modern range-based SLAM systems employ a two-stage estimation approach that first estimates the sensor ego-motion in real-time using local sensor data (i.e., odometry estimation) and then corrects accumulated errors by considering the global consistency of the map and trajectory.

For range-based odometry estimation, a combination of scan-to-model point cloud matching [1] and tight coupling of range and IMU constraints has been used in many studies [2]–[4]. A notable range-IMU odometry estimation algorithm, FAST-LIO2 [5], which has state-of-the-art accuracy with a limited computation cost, fuses the scan-to-model matching constraint and the IMU constraint on an efficient iterated Kalman filter. However, because the scan-to-model matching and state filtering approaches immediately fix the current sensor state estimate as a new observation arrives, they have difficulty dealing with situations where range data become completely degenerate for a moment. Because it is impossible to determine the current sensor state using only past observations in such situations, existing methods based on causal estimation cannot avoid estimation drift or corruption.

Pose graph optimization is the gold standard for global trajectory optimization owing to its efficiency and stability [6]. Based on relative pose observations between frames obtained using scan matching, it minimizes the sum of relative pose errors to correct estimation drift. In pose graph optimization, each relative pose constraint is modeled as a Gaussian distribution (i.e., mean and covariance matrix). In practice, however, it is difficult to accurately estimate the covariance matrix of a scan matching result because of the nonlinearity and non-continuity of the scan matching objective function [7]. Furthermore, the mean point (relative pose) is difficult to be obtained accurately when point clouds have only a small overlap. These difficulties lead to inaccurate modeling of relative pose constraints, resulting in deteriorated estimation accuracy for long trajectories with large loops.

Our aim in this work is to overcome the aforementioned shortcomings in existing range-based SLAM systems by introducing an extremely fast GPU-accelerated registration error evaluation algorithm and replacing the standards in both the odometry estimation and global trajectory optimization algorithms, namely scan-to-model matching and pose graph optimization with relative pose constraints.

We first propose an odometry estimation algorithm that fuses point cloud registration error constraints and IMU constraints for low-drift odometry estimation with a bounded computation cost. In contrast to conventional approaches, the proposed approach continually updates past sensor states in the optimization window (i.e., fixed-lag smoothing) and represents the scan matching target model as a set of selected past frames (i.e., keyframes). It propagates the latest sensor state estimate to past ones to correct estimation drift. This approach makes the odometry estimation extremely robust to a few seconds of completely degenerate range data as well as quick sensor motion. Furthermore, we incorporate multi-camera visual constraints in the odometry estimation factor graph in a tightly coupled way to further improve the stability and accuracy.

We then propose a global trajectory optimization algorithm that directly minimizes registration errors between submaps over the entire map (i.e., global multi-scan registration). Unlike pose graph optimization, this method avoids the Gaussian approximation of relative pose constraints and directly computes point cloud registration errors on the factor graph. It allows the relative pose between point clouds with a very small overlap to be accurately constrained. Furthermore, we incorporate IMU constraints into the global optimization factor graph in a tightly

This work was supported in part by JSPS KAKENHI Grant Number 23K16979 and a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

coupled way to better stabilize the global optimization and reduce trajectory estimation errors in four degrees of freedom (DoFs).

The proposed algorithms both require much more computation than that for conventional algorithms. Running them in real-time was thus considered to be infeasible. In this work, by carefully designing the registration error evaluation algorithm and the entire framework to fully leverage GPU parallel processing, we show that these algorithms can run in real-time on a single consumer-grade GPU.

Contributions: The main contributions of this work are as follows:

- We extend our GPU-accelerated voxelized GICP registration error factor [8] with surface-orientation-based correspondence validation and a multi-resolution voxelmap to improve its stability and accuracy in indoor environments. We also propose an efficient linearization mechanism for GPU-based factors to minimize CPU-GPU synchronization overhead and maximize optimization speed.
- We propose a keyframe-based tightly coupled range-IMU odometry estimation algorithm that is extremely robust to quick sensor motion and momentary degeneration of range data. We present a multi-camera visual-range-IMU extension to further improve the estimation accuracy.
- We propose a global trajectory optimization algorithm based on global registration error minimization with tightly coupled IMU constraints. We introduce the concept of submap *endpoints* to strongly constrain submap poses with a large time interval with IMU constraints.
- The proposed framework is carefully designed to be general to the sensor model. We show that it can handle various range-IMU sensors including not only LiDARs but also depth and stereo cameras.
- We release the majority of the code of the proposed framework, GLIM, as open source[1].

II. Related work

Modern SLAM frameworks comprise two modules: 1) an odometry estimation module that estimates sensor ego-motion in real-time using local sensor data and 2) a global trajectory optimization module that corrects accumulated estimation errors by considering the global consistency. We briefly review recent trends in both odometry estimation and global trajectory optimization for range-based SLAM and explain the concept of the proposed framework in this section.

A. Odometry Estimation

Feature vs. direct point cloud registration: Point cloud registration is a key element for sensor ego-motion estimation. Two types of method for point cloud registration are commonly used for range-based SLAM, namely feature-based approaches, which extract plane and edge points and align point clouds by matching these points, and direct approaches, which avoid feature extraction and use most of the input points to align point clouds.

[1]https://github.com/koide3/glim

Feature-based point cloud registration was first proposed for LOAM [1]. It extracts plane and edge points based on local smoothness, and then aligns point clouds by minimizing point-to-edge and point-to-plane matching distances. Usually, a process called frame-to-model matching follows the frame-by-frame scan matching to reduce the estimation drift. In the frame-to-model matching, aligned point clouds are accumulated in a single target point cloud (i.e., map model) and the sensor pose is estimated by aligning the current scan with the model point cloud. Many works employed a combination of feature-based registration and frame-to-model matching [9]–[11] because of its efficiency. This approach requires careful tuning of feature extraction, and feature extraction algorithms dedicated to specific sensors are often used [3], [9], [10].

Direct point cloud registration avoids feature extraction and uses most points to perform scan matching [5], [12]–[15]. Variants of the ICP algorithm with a local surface model (e.g., point-to-normal ICP [12] and generalized ICP [14], [16]) are often used owing to their accuracy and robustness. Although the computation cost of direct registration methods is higher than that of feature-based approaches, the former are more robust to sensor and environment changes because they do not require a delicate feature extraction process. The scan-to-model matching approach is often used for direct registration methods as well [5], [12], [14], [15].

Loose vs. tight IMU coupling: With progress in visual-inertial SLAM [17]–[19], IMU fusion has become an important technique for range-based SLAM [10]. The use of an IMU enables the prediction of sensor motion at a frequency of 100 to 1000 Hz, facilitating good initial estimates of the sensor pose and distortion correction of point clouds under quick sensor motion. Furthermore, IMU measurements provide information on the direction of gravity, enabling a reduction of trajectory estimation drift in four DoFs [17].

IMU and point cloud measurements can be fused using a loose coupling scheme, which separately considers range-based estimation and IMU-based estimation and fuses the estimation results in the pose space using a method such as the extended Kalman filter [20] or factor graph optimization [10], [21]. Because this approach enables heterogeneous sensor data (e.g., LiDAR, camera, and IMU) to be easily fused, several methods employ an IMU-centric loose coupling approach to robustly estimate the sensor state in extreme environments (e.g., an underground mine) [22]–[24]. Although a loose coupling scheme is computationally efficient and can easily handle heterogeneous sensor data, it is difficult to accurately propagate and fuse the uncertainty for each sensor with a loose coupling scheme, and tight coupling schemes are more accurate and robust than loose coupling schemes [2], [17].

A tight coupling scheme estimates the sensor states by directly minimizing a unified objective function that combines range- and IMU-based constraints. This approach enables the robust estimation of sensor states in environments where sufficient geometric information is not available through range data because the IMU constraints help to constrain the sensor state based on inertial information. Owing to their accuracy and robustness, tightly coupled LiDAR-IMU methods have been widely studied in recent years [2], [3], [25], [26].

Filtering vs. optimization: Tight coupling schemes can be categorized into filtering- and optimization-based approaches. The filtering approach keeps only the latest sensor state and immediately marginalizes old states when a new observation arrives. The optimization approach keeps past frames active and optimizes them via nonlinear optimization. To limit the computation cost, only a subset of past frames (e.g., a number of keyframes or frames in a sliding window) is usually considered for optimization. Although the filtering approach is efficient since it holds and optimizes only the latest sensor state, it quickly accumulates linearization errors. In the context of visual SLAM, it has been shown that the optimization-based approach outperforms the filtering-based approach in terms of accuracy because it re-linearizes past measurements [27], [28].

Despite the theoretical advantage of the optimization-based approach, in the context of range-based SLAM, many studies have used the filtering approach for IMU fusion [3], [5]. This was likely performed for two reasons. First, point cloud matching is more computationally expensive than visual feature matching and jointly considering multiple frames in a few seconds of the optimization window in real-time is considered to be infeasible. Second, the widely used frame-to-model matching approach needs to immediately fix the sensor pose estimate to accumulate points into a target model point cloud. Updating the sensor state estimates of past frames does not make a significant difference because the model point cloud is already frozen.

Continuous-time pose representation: LiDARs typically use a laser sweeping mechanism for wide field-of-view measurements that can cause point cloud distortion when the sensor is in motion. To compensate for this distortion, some works have used continuous-time sensor pose representation based on pose interpolation, which enables a sensor pose at any time step to be obtained [11], [29]–[31]. Compared to the discrete-time pose representation, the continuous-time representation allows for obtaining a sensor pose at any time step by nature and thus facilitates compensation of motion distortion on a per-point basis. It also enables incorporating IMU constraints without pre-integration by considering derivatives of the interpolated pose.

We, however, avoid the continuous-time representation and employ the conventional discrete-time representation in this work for two reasons. Firstly, emerging range sensors without a laser sweeping mechanism provide distortion-free point clouds with points acquired simultaneously, rendering continuous-time deskewing unnecessary (e.g., Microsoft Azure Kinect, Intel Realsense, and solid state LiDARs). Secondly, in the context of visual SLAM, it has been reported that the continuous-time representation does not improve estimation accuracy compared to the discrete counterpart, as long as the camera is well-synchronized with the IMU [32]. We thus consider the continuous-time representation unnecessary when point cloud distortion is not significant and can sufficiently be cancelled by IMU-based motion prediction.

Proposal: We propose a tightly coupled range-IMU odometry estimation algorithm based on keyframe-based point cloud matching and fixed-lag smoothing. In contrast to the conventional filtering-based approaches, which only optimize the latest sensor state, the proposed method keeps each frame active for a few seconds and enables dealing with momentary degeneration of range data by propagating the sensor states of successive frames to past frames. To make the framework generic (i.e., suitable for any range sensor), we use a direct point cloud registration approach based on a distribution-to-distribution comparison. Although these design choices lead to an enormous amount of computation, we show that it is feasible to run the algorithm in real-time by fully leveraging a modern GPU.

B. Global Trajectory Optimization

Pose graph optimization: Pose graph optimization constructs a factor graph with (SE3) relative pose constraints and optimizes the sensor poses by minimizing the errors in the pose space. This approach has been widely used for global trajectory optimization in many SLAM systems owing to its simplicity and efficiency [9], [10], [17]. Pose graph optimization models each relative pose constraint as a Gaussian distribution on a SE3 manifold. In practice, however, it is difficult to accurately estimate the covariance matrix of a relative pose given by scan matching. Although a closed-form method is commonly used to estimate the covariance matrix from the Hessian matrix of an ICP scan matching result [33], this approach tends to be optimistic because it considers only the last linearized objective function and cannot take into account point correspondence changes [7]. Although Monte-Carlo-based covariance estimation methods [34] are more accurate than the Hessian-based method, they require scan matching to be performed multiple times, resulting in a large computation cost. Landry et al. proposed a learning-based method that takes point cloud descriptors and estimates the covariance matrix based on an offline trained model. While this approach aims to balance estimation accuracy and processing cost, its estimation accuracy can deteriorate in unseen environments. Most existing range-based SLAM frameworks use only a constant covariance matrix [12], a simple weighting scheme [9], [10], or Hessian-based closed-form covariance estimation [35]. Such inaccurate constraint modeling deteriorates the estimation accuracy of long trajectories with large loops.

For point clouds with only a small overlap, it is difficult to explicitly determine the relative pose between the point clouds via scan matching. Thus, existing frameworks, which rely on pose graph optimization, close loops only when there is sufficient overlap between point clouds, and discard information on point cloud pairs with a small overlap. If we forcibly close loops between such frames, the corrupted scan matching results turn into inaccurate relative constraints and result in deteriorated trajectory estimation [36].

Bundle adjustment: Bundle adjustment (BA) is an approach for simultaneously estimating environment parameters (e.g., 3D landmark positions) and sensor states over frames. It is an essential technique for visual SLAM and is widely used for both odometry estimation [17] and global optimization [19]. For range-based SLAM, the BA problem is often formulated as the joint optimization of line and plane environment parameters and sensor poses [37], [38]. Liu and Zhang pointed

out that line and plane parameters can be eliminated from the optimization variables because they can be determined from point coordinates and sensor poses, and formulated line and plane BA as the eigenvalue minimization of accumulated points [39]. Huang et al. further re-formulated the plane BA problem in least-squares form for efficient optimization [40]. Although BA-based methods promise consistent mapping results, they need a good initial guess of the sensor poses for correspondence estimation. In particular, the least squares BA formulation [40] is sensitive to the initial guess because its surface normal estimation is separated from sensor pose optimization. BA-based methods thus have difficulty in closing loops under a large estimation drift and need to be combined with hierarchical optimization with pose graph [41].

Global matching cost minimization: Lu and Milios formulated the 2D range-based mapping problem as the minimization of scan registration errors on a factor graph [42]. Several works extended this approach to three dimensions by explicitly handling loop closing events to reduce optimization executions [43], [44]. Because this approach does not explicitly require the relative pose between frames (it just evaluates the registration error between them), it can naturally propagate the point matching uncertainty to the frame uncertainty and accurately constrain the relative pose between frames with only a small overlap. However, this approach is computationally expensive because registration errors need to be re-evaluated over the entire map in each optimization iteration and thus real-time global registration error minimization is considered to be infeasible.

Reijgwart et al. proposed a volumetric mapping framework based on signed distance function submaps [45]. Their framework creates local submaps in the form of the Euclidean signed distance field (ESDF) and optimizes the submap poses such that the registration errors between them are minimized. Because the distance between a point and a submap surface can be directly obtained with ESDF, efficient registration error computation can be conducted without a costly nearest neighbor search. It is, however, still computationally demanding to compute the global registration error and the optimization is carried out with only a random subset of registration errors with the support of SE3 relative pose constraints.

IMU constraints in global optimization: Considering that IMU data provide information on the direction of gravity and suppress odometry estimation drift in yaw and pitch rotation, one can perform global trajectory optimization with four DoF pose graph optimization to keep the global optimization result gravity aligned [17]. Odometry estimation results, however, may not be perfectly aligned to the gravity direction due to bias estimation errors. In such cases, the four DoF global optimization would result in an inconsistent mapping result.

We can also consider directly inserting IMU constraints into the global optimization factor graph [46], [47]. However, keyframes (or submaps) for global optimization are usually created with a larger time interval. If we simply create an IMU factor between keyframes, we cannot strongly constrain their poses because a longer integration time results in a larger uncertainty of the IMU constraint [46]. Thus, this approach requires the frequent creation of keyframes, which results in redundant computation and a limitation of the factor graph design.

Usenko et al. proposed a nonlinear factor recovery technique for transferring information accumulated during visual inertial odometry to global optimization [48]. In this technique, a set of nonlinear factors are created to approximate a linearized subgraph (Markov bracket) of the odometry estimation factor graph to represent the original distribution with a sparse graph topology. However, because this approach recovers a linearized subgraph, linearization errors are unavoidable.

Proposal: Compared to the conventional pose graph optimization approach, the proposed global registration error minimization approach allows the relative pose between submaps with a very small overlap to be accurately constrained, resulting in highly consistent mapping results. As in the proposed odometry estimation algorithm, we leverage GPU parallel processing to perform global matching cost minimization in real-time. Furthermore, we introduce the concept of submap *endpoints* to keep the integration time for each IMU factor small and strongly constrain the submap poses with inertial information, which is often unused in existing works.

III. Proposed System

We give an overview of the proposed framework in Sec. III-A. We then describe the notation and building blocks for the estimation modules in Sec. III-B to Sec. III-D. Lastly, we present the proposed odometry estimation and global optimization algorithms in Sec. IV and Sec. V.

A. System Overview

Fig. 1 shows an overview of the proposed framework. The mapping system comprises a preprocessing module and three estimation modules, namely odometry estimation, local mapping, and global mapping. All of the estimation modules are based on tight coupling of the range and IMU constraints. Optionally, multi-camera visual information is incorporated in the odometry estimation module to improve the stability and accuracy. We designed the framework to be monolithic for efficiency; each module runs in an individual thread and the entire system is executed as a single process.

B. Notation

We define the sensor state $\boldsymbol{x}_t$ at time t that will be estimated in the estimation modules as

$$\boldsymbol{x}_t = [\boldsymbol{T}_t, \boldsymbol{v}_t, \boldsymbol{b}_t], \tag{1}$$

where $\boldsymbol{T}_t = [\boldsymbol{R}_t|\boldsymbol{t}_t] \in \mathrm{SE}(3)$ is the sensor pose, $\boldsymbol{v}_t \in \mathbb{R}^3$ is the velocity, and $\boldsymbol{b}_t = [\boldsymbol{b}_t^a, \boldsymbol{b}_t^\omega] \in \mathbb{R}^6$ is the IMU linear acceleration and angular velocity bias. We estimate the time series of sensor states from point clouds $\mathcal{P}_t$, IMU measurements (linear acceleration $\boldsymbol{a}_t$ and angular velocity $\boldsymbol{\omega}_t$), and optional camera images $\mathcal{I}_t$. Note that we assume that the transformations between the range sensor, IMU, and camera are known. We transform point clouds into the IMU frame to treat them as if they are in a unified sensor coordinate frame for efficiency and simplicity.

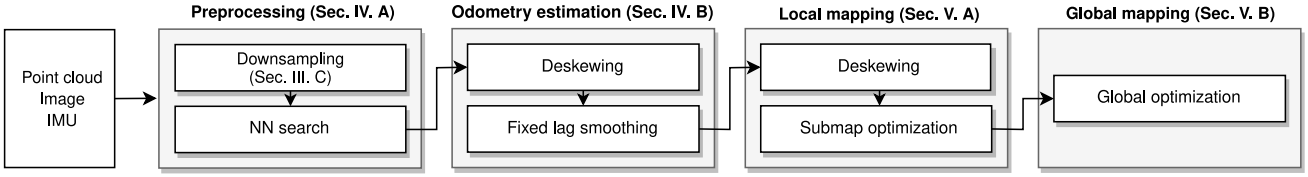


Fig. 1. The proposed framework comprises preprocessing module and three estimation modules (odometry estimation, local mapping, and global mapping) that are all based on tight coupling of range and IMU constraints. The input point clouds are fed to the preprocessing module for downsampling and nearest neighbor search. Then, the odometry estimation module performs point cloud deskewing to remove motion distortion, and the sensor motion is estimated through fixed-lag smoothing. The local mapping module performs deskewing again with the odometry estimation result and concatenates several frames into one submap after local map optimization. Finally, the global mapping module takes as input the submaps and optimizes their poses to improve global consistency. Each module runs in an individual thread and the entire system is executed as a single process.

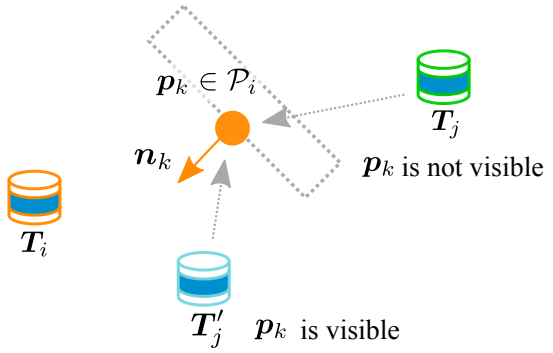


Fig. 2. Correspondence validation based on surface orientation. $\boldsymbol{p}_k \in \mathcal{P}_i$ is not visible from $\boldsymbol{T}_j$, which is on the opposite side of the surface, but is visible from $\boldsymbol{T}'_j$, which is on the same side of the surface.

C. Matching Cost Factor

The matching cost factor constrains two sensor poses ($\boldsymbol{T}_i$ and $\boldsymbol{T}_j$) such that the registration error between the point clouds ($\mathcal{P}_i$ and $\mathcal{P}_j$) is minimized. As the matching cost, we choose voxelized GICP (VGICP) [49], which is a variant of GICP [16] with a voxel-based data association. To improve the estimation accuracy in indoor environments, we extend our previous VGICP implementation [49] with surface-orientation-based correspondence validation and a multi-resolution voxelmap.

Preprocessing: VGICP models each input point $\boldsymbol{p}_k \in \mathcal{P}_i$ as a Gaussian distribution $\boldsymbol{p}_k = (\boldsymbol{\mu}_k, \boldsymbol{C}_k)$, which represents the local shape of the surface around $\boldsymbol{p}_k$. The covariance matrix $\boldsymbol{C}_k$ is calculated from neighboring points of $\boldsymbol{p}_k$ given by a k-nearest-neighbor search. We then create a sparse voxelmap with spatial voxel hashing [50] and take the average of the points and their covariances in each voxel. To enlarge the basin of convergence, we create voxelmaps with L resolution levels (e.g., three levels) scaled by a power of two; the voxel size of the l-th voxelmap is given by $r^l = r_0 2^{(l-1)}$, where r_0 is the base voxel size. An input point is associated with corresponding voxels in each of the resolution levels to compute the point-to-voxel distance.

Note that comparing a point and the averaged distribution of a voxel is equivalent to comparing the point and all of the points in the voxel in a distribution-to-distribution comparison, as shown in [49]. Because we can obtain a valid distribution even on a distant voxel with only a few points, this approach results in better registration accuracy compared to that for normal distributions transform (NDT) [51], which needs many points for each voxel to obtain a proper voxel distribution.

Correspondence search: In indoor environments, both sides of a thin wall are often observed as the sensor moves. In such cases, a proximity-based correspondence search can wrongly associate points on one side of the wall with points on the other side. To avoid this, we check whether each input point $\boldsymbol{p}_k \in \mathcal{P}_i$ is visible from the viewpoint of the other point cloud $\mathcal{P}_j$ based on the surface normal $\boldsymbol{n}_k$, as shown in Fig. 2. If $(\boldsymbol{p}_k - \boldsymbol{T}_i^{-1}\boldsymbol{t}_j) \cdot \boldsymbol{n}_k > 0$, $\boldsymbol{T}_j$ is on the other side of the surface and the point is considered to be not visible from $\boldsymbol{T}_j$ and discarded. Otherwise, we consider that $\boldsymbol{p}_k$ is visible from $\boldsymbol{T}_j$ and calculate the corresponding voxel coordinates $\boldsymbol{q}_k = \text{floor}(\boldsymbol{p}_k / r) \in \mathbb{Z}^3$, where r is the voxel resolution. The corresponding voxels at each of the voxel resolution levels are looked up from the voxelmaps of $\mathcal{P}_j$. This simple correspondence validation works effectively with the odometry estimation based on frame-by-frame comparison proposed in Sec. IV.

Linearization: As the objective function to be minimized, we calculate the sum of distribution-to-distribution distances between points $\boldsymbol{p}_k = (\boldsymbol{\mu}_k, \boldsymbol{C}_k)$ and their corresponding voxels $\tilde{\boldsymbol{p}}_k^l = (\tilde{\boldsymbol{\mu}}_k^l, \tilde{\boldsymbol{C}}_k^l)$ as follows:

$$e^{PC}(\boldsymbol{T}_i, \boldsymbol{T}_j) = \sum_{l \in [1 \cdots L]} \sum_{\boldsymbol{p}_k \in \mathcal{P}_i} e^{D2D}\left(\boldsymbol{p}_k, \tilde{\boldsymbol{p}}_k^l, \boldsymbol{T}_i^{-1}\boldsymbol{T}_j\right), \quad (2)$$

$$e^{D2D}(\boldsymbol{p}_k, \tilde{\boldsymbol{p}}_k^l, \boldsymbol{T}_{ij}) = \boldsymbol{d}_k^T \left(\tilde{\boldsymbol{C}}_k^l + \boldsymbol{T}_{ij}\boldsymbol{C}_k\boldsymbol{T}_{ij}^T\right)^{-1} \boldsymbol{d}_k, \quad (3)$$

where $\boldsymbol{T}_{ij} = \boldsymbol{T}_i^{-1}\boldsymbol{T}_j$ is the relative pose between $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ and $\boldsymbol{d}_k = \tilde{\boldsymbol{\mu}}_k^l - \boldsymbol{T}_{ij}\boldsymbol{\mu}_k$ is the residual between $\boldsymbol{\mu}_k$ and $\tilde{\boldsymbol{\mu}}_k^l$. As in the original GICP, we assume that the displacements of $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ are small in each optimization iteration and use $\boldsymbol{\Omega}_k = \left(\tilde{\boldsymbol{C}}_k^l + \boldsymbol{T}_{ij}\boldsymbol{C}_k\boldsymbol{T}_{ij}^T\right)^{-1}$ fixed at the linearization point to turn Eq. 3 into a weighted least squares form.

Because e^{PC} is in a weighted least squares form, we can derive a linear quadratic factor, which is composed of information matrices $\boldsymbol{H}_{ii}, \boldsymbol{H}_{ij}$, and $\boldsymbol{H}_{jj}$ and information vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, from the first-order derivatives:

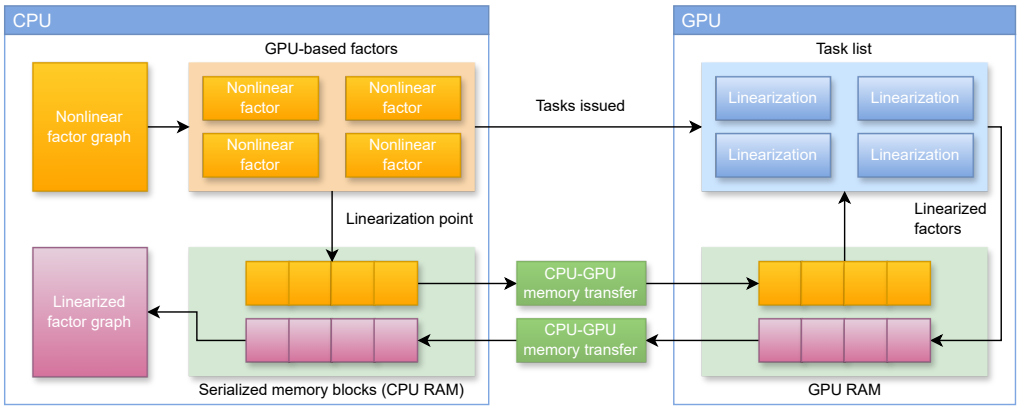


Fig. 3. Efficient factor linearization on GPU. The input and output data for linearization are serialized in a memory block and uploaded to and downloaded from the GPU with a single memory transfer. Linearization tasks are issued to the GPU task list and processed in parallel. This mechanism invokes only two CPU-GPU data transfer calls per factor graph linearization task regardless of the number of factors in the factor graph.

$$\boldsymbol{A}_k = \frac{\partial \boldsymbol{d}_k}{\partial \boldsymbol{T}_i} = [\boldsymbol{R}_{ij}(\boldsymbol{\mu}_k)_\times, -\boldsymbol{R}_{ij}], \tag{4}$$

$$\boldsymbol{B}_k = \frac{\partial \boldsymbol{d}_k}{\partial \boldsymbol{T}_j} = [-(\boldsymbol{T}_{ij}\boldsymbol{p}_k)_\times, \mathbb{I}_{3\times 3}], \tag{5}$$

$$\boldsymbol{H}_{ii} = \sum_k^N \boldsymbol{A}_k \boldsymbol{\Omega}_k \boldsymbol{A}_k^T, \quad \boldsymbol{H}_{ij} = \sum_k^N \boldsymbol{A}_k \boldsymbol{\Omega}_k \boldsymbol{B}_k^T, \tag{6}$$

$$\boldsymbol{H}_{jj} = \sum_k^N \boldsymbol{B}_k \boldsymbol{\Omega}_k \boldsymbol{B}_k^T, \tag{7}$$

$$\boldsymbol{b}_i = \sum_k^N \boldsymbol{A}_k \boldsymbol{\Omega}_k \boldsymbol{d}_k, \quad \boldsymbol{b}_j = \sum_k^N \boldsymbol{B}_k \boldsymbol{\Omega}_k \boldsymbol{d}_k, \tag{8}$$

where $()_\times$ transforms a vector into a skew symmetric matrix (i.e., the hat map).

Implementation: The correspondence search and the linearization can be efficiently computed on a GPU because they are based on simple memory lookup and linear matrix operations and do not involve conditional branches that cause thread divergence. However, simply linearizing factors one-by-one (in a *for* loop) would require CPU-GPU synchronization for each factor, which would lead to a large overhead and would significantly affect processing speed.

To avoid the synchronization overhead and fully leverage the power of a GPU, we developed an efficient linearization mechanism, which is shown in Fig. 3. We first list all GPU-based nonlinear factors (matching cost factors in our case) and serialize all data required for linearization (e.g., linearization points) in a single memory block. The serialized input memory block is uploaded to the GPU with a single memory transfer call. Then, the linearization tasks are issued to the GPU task list and processed via several processing streams in parallel. The linearization results are serialized into a memory block on the GPU and sent back to the CPU after CPU-GPU synchronization. This implementation minimizes GPU overhead and maximizes processing speed because it invokes only two CPU-GPU data transfer calls per factor graph linearization task regardless of the number of factors to be linearized.

D. IMU Preintegration Factor

We use the IMU preintegration technique [28] to efficiently incorporate IMU constraints into the factor graph. Given an IMU measurement ($\boldsymbol{a}_t$ and $\boldsymbol{\omega}_t$), the sensor state evolves over time as follows:

$$\boldsymbol{R}_{t+\Delta t} = \boldsymbol{R}_t \exp\left((\boldsymbol{\omega}_t - \boldsymbol{b}_t^\omega - \boldsymbol{\eta}_k^\omega)\,\Delta t\right), \tag{9}$$

$$\boldsymbol{v}_{t+\Delta t} = \boldsymbol{v}_t + \boldsymbol{g}\Delta t + \boldsymbol{R}_t\left(\boldsymbol{a}_t - \boldsymbol{b}_t^a - \boldsymbol{\eta}_t^a\right)\Delta t, \tag{10}$$

$$\boldsymbol{t}_{t+\Delta t} = \boldsymbol{t}_t + \boldsymbol{v}_t\Delta t + \frac{1}{2}\boldsymbol{g}\Delta t^2 + \frac{1}{2}\boldsymbol{R}_t\left(\boldsymbol{a}_t - \boldsymbol{b}_t^a - \boldsymbol{\eta}_t^a\right)\Delta t^2, \tag{11}$$

where $\boldsymbol{g}$ is the gravity vector and $\boldsymbol{\eta}_t^a$ and $\boldsymbol{\eta}_t^\omega$ represent white noise in the IMU measurement.

The IMU preintegration factor integrates the system evolution between two time steps i and j to obtain the relative body motion $\Delta\boldsymbol{R}_{ij}$, $\Delta\boldsymbol{t}_{ij}$, and $\Delta\boldsymbol{v}_{ij}$ (see [28] for a detailed derivation) to constrain the sensor states:

$$\begin{aligned} e^{IMU}(\boldsymbol{x}_i, \boldsymbol{x}_j) = &\| \log\left(\Delta\boldsymbol{R}_{ij}^T \boldsymbol{R}_i^T \boldsymbol{R}_j\right) \|^2 \\ &+ \| \Delta\boldsymbol{t}_{ij} - \boldsymbol{R}_i^T\left(\boldsymbol{t}_j - \boldsymbol{t}_i - \boldsymbol{v}\Delta t_{ij} - \frac{1}{2}\boldsymbol{g}\Delta t_{ij}^2\right) \|^2 \\ &+ \| \Delta\boldsymbol{v}_{ij} - \boldsymbol{R}_i^T\left(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij}\right) \|^2. \end{aligned} \tag{12}$$

The IMU preintegration factor enables us to keep the factor graph well constrained in environments where geometric features are insufficient and matching cost factors can be degenerated. Furthermore, it provides information on the direction of gravity and reduces the estimation drift in four DoFs [17].

IV. Odometry Estimation

In this section, we present the odometry estimation algorithm based on range and IMU data fusion. The key difference compared to existing methods is that the proposed algorithm employs a combination of fixed-lag smoothing and keyframe-based point cloud matching rather than frame-to-model matching and filtering-based estimation. Fixed-lag smoothing contin-

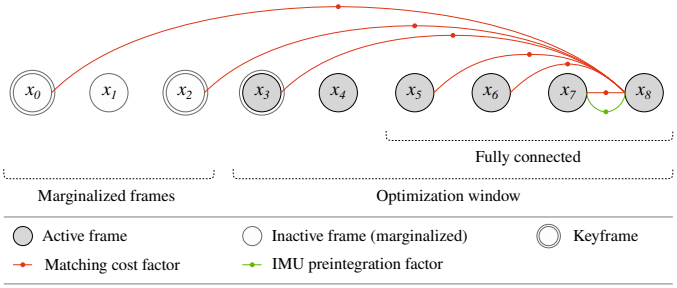


Fig. 4. Odometry estimation factor graph. Only factors related to the latest frame ($\boldsymbol{x}_8$) are visualized. The latest frame is connected with keyframes via matching cost factors to reduce the estimation drift. It is also connected with the last few frames to make the estimation robust to quick sensor motion. IMU factors are created between consecutive frames to keep the factor graph well constrained.

ually updates the sensor states when they are in the optimization window. This can be considered to be frame-to-model matching with an active target model. This approach makes the estimation robust to momentary degeneration of range data because it puts sensor state estimates on hold for a few seconds and propagates the latest sensor state estimate to the past ones to correct estimation drift once sufficient geometric constraints on range data become available. Furthermore, unlike filtering-based methods, variables and constraints do not need to be inserted in a strict time order; we can insert constraints for past sensor states as long as the states are still in the optimization window. This enables us to asynchronously run additional data processing (e.g., visual feature tracking) and insert constraints into the factor graph with a slight delay.

A. Preprocessing

We first downsample the input point cloud and find k neighboring points for each point that are required for the subsequent point covariance estimation performed after motion distortion compensation. We assume that the neighborhood relationship of points does not change significantly during motion distortion compensation and use the precomputed nearest neighbors for covariance estimation to reduce the processing time. Note that the costly exact nearest neighbor search is only performed in the preprocessing step; all following estimation steps use a voxel-based approximate nearest neighbor search.

B. Tightly Coupled Range-IMU Odometry Estimation

We first correct the distortion of the point cloud caused by sensor motion by transforming points into the IMU frame with motion prediction based on IMU dynamics. We then compute the covariance of each point using the precomputed neighboring points.

Given the point clouds and IMU measurements, we construct the factor graph shown in Fig. 4. To limit computation cost and ensure that the odometry estimation algorithm is real-time capable, we use a fixed-lag smoothing approach and marginalize the old frames that move out of the optimization window. We use iSAM2 [52] and its efficient Bayes-tree-based variable elimination for fixed-lag smoothing implemented in GTSAM [53].

Inspired by direct sparse odometry [54], we introduce a keyframe mechanism for efficient and low-drift trajectory estimation. Keyframes are a set of frames that are selected such that they are spatially well distributed while having sufficient overlap with the latest frame. We create a matching cost factor between the latest frame and each keyframe to efficiently reduce estimation drift. If a keyframe is already marginalized by the fixed-lag smoother, we consider the keyframe pose as fixed and create a matching cost factor that constrains the latest sensor pose with respect to the fixed keyframe.

To manage keyframes, we define an overlap rate between two point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$ as the fraction of points in $\mathcal{P}_i$ that fall within a voxel of $\mathcal{P}_j$ [36]. Whenever a new point cloud frame arrives, we evaluate the overlap rate between that frame and the union of all keyframes, and if the overlap is smaller than a threshold (e.g., 90%), we insert that frame into the keyframe list. Similar to the keyframe marginalization strategy in [54], we remove redundant keyframes using the following strategy:

1) We remove keyframes that overlap the latest keyframe by less than a certain threshold (e.g., 5%).
2) If more than N^{odom} (e.g., 20) frames exist in the keyframe list, we remove the keyframe that minimizes the following score:

$$s(i) = o(i, N^{odom}) \sum_{j \in [1, N^{odom}-1] \setminus \{i\}} (1 - o(i, j)), \quad (13)$$

where $o(i, j)$ is the overlap rate between the i-th and j-th keyframes. The score function is heuristically designed to keep keyframes spatially well distributed while leaving more keyframes close to the latest one.

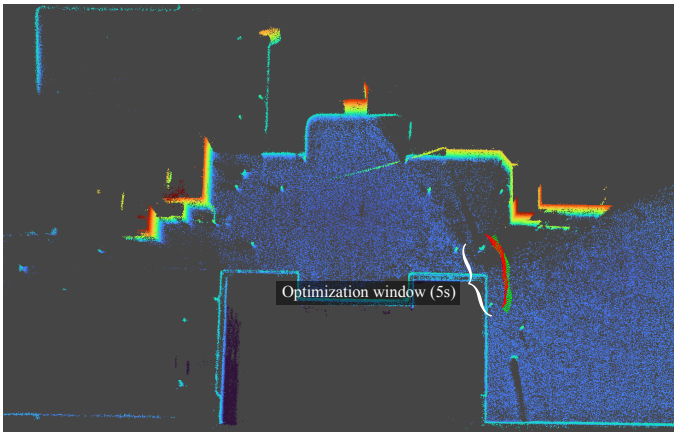
In addition to the keyframes, we create matching cost factors between the latest frame and the last N^{pre} frames (e.g., last three frames) to make the odometry estimation robust to quick sensor motion. We also create an IMU preintegration factor between consecutive frames for robustness in featureless environments.

In summary, the objective function of the odometry estimation algorithm is given by:

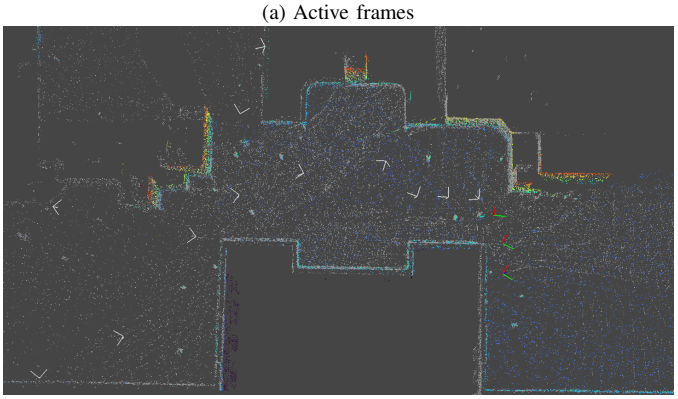
$$\begin{aligned} g^{LIO}(\mathcal{X}^a) = & \sum_{\boldsymbol{x}_i \in \mathcal{X}^a} \sum_{\boldsymbol{x}_j \in \mathcal{X}_i^p \cup \mathcal{X}^k} e^{PC}\left(\mathcal{P}_i, \mathcal{P}_j, \boldsymbol{T}_i, \boldsymbol{T}_j\right) \\ & + \sum_{\boldsymbol{x}_i \in \mathcal{X}^a} e^{IMU}\left(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i\right) + e^{MG}\left(\mathcal{X}^a\right), \end{aligned} \quad (14)$$

where $\mathcal{X}^a$ is the set of active frames in the optimization window, $\mathcal{X}_i^p = [\boldsymbol{x}_{i-N^{pre}}, \cdots, \boldsymbol{x}_{i-1}]$ is the preceding frames of $\boldsymbol{x}_i$, $\mathcal{X}^k$ is the keyframes, and e^{MG} is the error term to compensate for marginalized variables and factors.

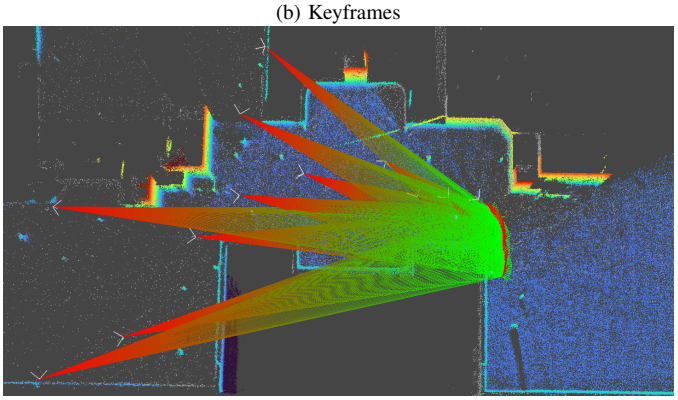
Fig. 5 shows examples of active frames, keyframes, and matching cost factors. Coordinate frames and points shown in color represent active frames in the optimization window of the fixed-lag smoother (5 s) and point clouds. Ones shown in gray are the fixed (marginalized) keyframes and point clouds. Fig. 5 (a) shows active frames subject to optimization and Fig. 5 (b) shows keyframes. We can see that the keyframes are selected such that they are spatially well distributed to efficiently reduce estimation drift. The first few keyframes are still in



(a) Active frames



(b) Keyframes



(c) Matching cost factors

Fig. 5. Example of created keyframes and matching cost factors. Coordinate frames and points shown in color represent active frames in the optimization window (5 s) and those shown in gray are fixed. The matching cost factors are created between the active frames and the keyframes that are selected to be spatially well distributed to efficiently reduce estimation drift.

the optimization window and are considered to be active. Fig. 5 (c) shows matching cost factors densely created between the active frames and the keyframes. Although it is difficult to visually confirm, there are also matching cost factors between active frames, as in Fig. 4, that make the estimation robust to quick sensor motion. It is worth emphasizing that we re-evaluate each matching cost factor (each line in Fig. 5 (c)) in each optimization iteration and minimize the sum of registration errors over the densely connected factor graph in real-time.

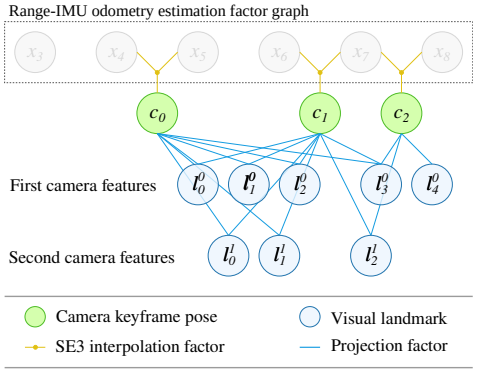


Fig. 6. Multi-camera extension for odometry estimation factor graph. Camera keyframe poses are introduced as interpolation of IMU poses and constrained by visual feature projection factors.

C. Tightly Coupled Multi-Camera Visual Constraints

We designed the proposed framework to be modular so that it can accept additional constraints to further improve the estimation stability and accuracy. Here, we present a multi-camera extension for tightly coupled visual-range-IMU odometry estimation.

Fig. 6 shows the odometry estimation factor graph extended with multi-camera visual constraints. We assume that the cameras are hardware-synchronized and take images at the same moment. For each camera, we detect and track 2D visual feature points using curvature extreme tracking [55]. Whenever the range sensor measures a certain amount of displacement (e.g., 0.25 m or 15°) or the time since the last visual keyframe creation exceeds a threshold (e.g., 3 s), we create a visual keyframe $\boldsymbol{C}_t \in \mathrm{SE}(3)$ that represents the IMU pose at the moment when an image set was taken. To constrain $\boldsymbol{C}_t$, we interpolate the closest two IMU poses $\boldsymbol{T}_L = [\boldsymbol{R}_L|\boldsymbol{t}_L]$ and $\boldsymbol{T}_R = [\boldsymbol{R}_R|\boldsymbol{t}_R]$ that cover the time point of $\boldsymbol{C}_t$ using spherical linear interpolation (SLERP):

$$\boldsymbol{t}_t^c = (1-\alpha)\boldsymbol{t}_L + \alpha \boldsymbol{t}_R, \tag{15}$$

$$\boldsymbol{R}_t^c = \mathrm{SLERP}(\boldsymbol{R}_L, \boldsymbol{R}_R, \alpha), \tag{16}$$

$$\alpha = \frac{t^C - t^L}{t^R - t^L}, \tag{17}$$

$$e^{VK}(\boldsymbol{C}_t) = \| \log\left(\boldsymbol{C}_t^{-1}[\boldsymbol{R}_t^c|\boldsymbol{t}_t^c]\right) \|^2, \tag{18}$$

where $t^L < t^C < t^R$ are the time points of $\boldsymbol{T}_L$, $\boldsymbol{C}_t$, and $\boldsymbol{T}_R$, respectively.

Everytime a visual keyframe is created, we extract visual features that appear in both the previous and current visual keyframes. We then filter out outlier visual features via a two-stage feature validation. It first removes outliers by using RANSAC-based essential matrix estimation and then triangulates 3D feature positions based on the predicted sensor poses given by LiDAR-IMU fusion. Visual features with reprojection errors larger than a threshold are then eliminated. For each visual feature that passes the validation, we create a projection factor to constrain the corresponding landmark position and visual keyframe pose.

Given a 2D visual feature point ${}^i\boldsymbol{f}_t^j \in \mathbb{R}^2$ tracked by the i-th camera at time t with a feature ID j, we create a projection

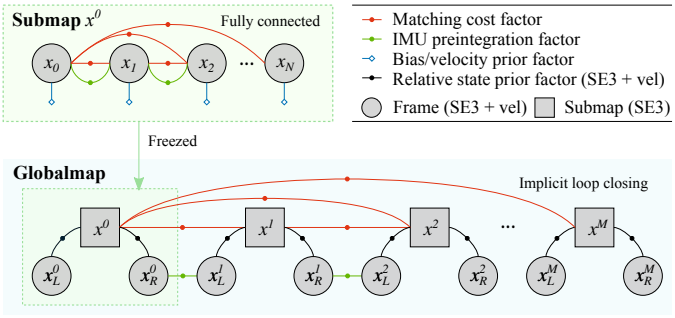


Fig. 7. Factor graph for global optimization. The local mapping module refines estimation results and merges several local frames into one submap using an all-to-all registration strategy. The global mapping module optimizes the submap poses such that the global registration error is minimized over the entire map. Both modules take advantage of IMU factors to stabilize the estimation in severe featureless environments and reduce estimation drift.

factor with Huber's robust kernel defined as:

$$e^{VF}(\boldsymbol{C}_t, \boldsymbol{l}_i^j) = \| {}^i\boldsymbol{f}_t^j - \rho\left(\pi\left(\boldsymbol{C}_t \boldsymbol{T}_{C_i}^I \boldsymbol{l}_i^j\right)\right) \|^2, \tag{19}$$

where $\boldsymbol{l}_i^j \in \mathbb{R}^3$ is the j-th visual landmark position associated with the i-th camera, $\boldsymbol{T}_{C_i}^I$ is the constant transformation between the IMU frame and the optical frame of the i-th camera, π is the projection function, and ρ is the robust kernel.

The objective function for odometry estimation extended with visual constraints is defined as:

$$\begin{aligned} g^{LVIO}(\mathcal{X}^a, \mathcal{C}^a, \mathcal{L}^a) = g^{LIO}(\mathcal{X}^a) + \sum_{\boldsymbol{C}_t \in \mathcal{C}^a} e^{VK}(\boldsymbol{C}_t) \\ + \sum_{i=1}^{N^{CAM}} \sum_{\boldsymbol{C}_t \in \mathcal{C}^a} \sum_{\boldsymbol{l}_i^j \in \mathcal{L}_t^i} e^{VF}(\boldsymbol{C}_t, \boldsymbol{l}_i^j), \end{aligned} \tag{20}$$

where $\mathcal{C}^a$ and $\mathcal{L}^a$ are respectively the sets of visual keyframe poses and visual landmarks in the optimization window, and $\mathcal{L}_t^i$ is a set of visual landmarks that have an observation on i-th camera image at time t. As shown in Fig. 6, all of the variables are jointly optimized on a single factor graph by considering multi-sensor constraints (i.e., tightly coupled visual-range-IMU odometry estimation).

V. Global Trajectory Optimization

A. Local Mapping

Once a frame is marginalized from the odometry estimation graph, it is fed to the local mapping module as an initial estimate of the sensor state. The local mapping module merges several local frames into one submap to reduce the number of optimized variables in the global mapping module.

We first perform deskewing and covariance estimation again with the marginalized state, which is expected to be improved from the initial prediction made at the beginning of the odometry estimation. We then evaluate the overlap rate between that frame and the latest frame in the submap, and if the overlap rate is smaller than a threshold (e.g., 90%), we insert that frame into the submap factor graph.

As shown in Fig. 7, we create a matching cost factor for each combination of frames in the submap (i.e., all-to-all registration). We also add an IMU preintegration factor between consecutive frames and add a prior factor for the velocity and bias of each frame, based on the marginalized state, to better stabilize submap optimization.

The objective function for the local mapping is thus defined as:

$$\begin{aligned} g^{LM}(\mathcal{X}^s) = \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in {}_2C_{\mathcal{X}^s}} e^{PC}(\boldsymbol{T}_i, \boldsymbol{T}_j) \\ + \sum_{\boldsymbol{x}_i \in \mathcal{X}^s} e^{IMU}(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i) + \sum_{\boldsymbol{x}_i \in \mathcal{X}^s} e^{BV}(\boldsymbol{x}_i), \end{aligned} \tag{21}$$

where $\mathcal{X}^s$ is the set of frames in the submap, ${}_2C_{\mathcal{X}^s}$ is all combinations of the frames, $e^{BV}(\boldsymbol{x}_i) = \|\boldsymbol{b}_i - \boldsymbol{b}_i'\|^2 + \|\boldsymbol{v}_i - \boldsymbol{v}_i'\|^2$ is a prior factor to keep the velocity and IMU bias estimates close to those of the marginalized state ($\boldsymbol{b}_i'$ and $\boldsymbol{v}_i'$).

Once the number of frames in the submap becomes equal to N^{sub} (e.g., 15), or the overlap between the first and last frames becomes smaller than a threshold (e.g., 5 %), we perform factor graph optimization using the Levenberg-Marquardt optimizer [56] and merge the frames into one submap based on the optimization result. In contrast to the odometry estimation, we perform batch optimization for local mapping. This enables us to correct estimation drift caused by a longer period of range data degeneration that is difficult to handle through the online optimization with a bounded optimization window used in the odometry estimation.

B. Global Mapping

The global mapping module optimizes the submap poses such that the registration errors between them are minimized over the entire map. It creates a matching cost factor between each submap pair with an overlap rate that exceeds a small threshold (e.g., 5%). This results in an extremely dense factor graph. Because each submap is aligned with not only adjacent submaps on the graph but also each revisited submap, trajectory loops are implicitly closed.

Submaps are created with a larger time interval. If we simply create an IMU factor between submap states, its uncertainty becomes very large and it cannot strongly constrain the relative pose between submaps [18], [46]. Furthermore, we also lose information on the velocity and IMU bias estimated by the odometry estimation module. To address these problems, we introduce two states called *endpoints* ($\boldsymbol{x}_L^i$ and $\boldsymbol{x}_R^i$) for each submap $\boldsymbol{x}^i$; they hold the states of the first and last frames in the submap with respect to the submap pose.

Given an estimate of sensor states $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{N^{sub}}]$ in a submap $\boldsymbol{x}^i$, we define the submap origin $\boldsymbol{T}^i = [\boldsymbol{R}^i | \boldsymbol{t}^i]$ as the pose of the sensor at the center $\boldsymbol{T}_{N^{sub}/2}$. Then, the sensor state $\boldsymbol{x}_t$ relative to the submap origin is given as:

$$\boldsymbol{T}_t' = \left(\boldsymbol{T}^i\right)^{-1} \boldsymbol{T}_t, \tag{22}$$

$$\boldsymbol{v}_t' = \left(\boldsymbol{R}^i\right)^{-1} \boldsymbol{v}_t, \tag{23}$$

$$\boldsymbol{b}_t' = \boldsymbol{b}_t. \tag{24}$$

We consider $\boldsymbol{T}_t'$, $\boldsymbol{v}_t'$, and $\boldsymbol{b}_t'$ to be fixed and create relative state factors between the submap state $\boldsymbol{x}^i$ and endpoints $\boldsymbol{x}_L^i$ and $\boldsymbol{x}_R^i$ respectively from the first and last frames in the

submap ($\boldsymbol{x}_1$ and $\boldsymbol{x}_{N^{sub}}$) such that they satisfy the relative state relationships:

$$\begin{aligned} e_L^{EP}(\boldsymbol{T}^i, \boldsymbol{x}_L^i) &= \| \log \left(\boldsymbol{T}^i \boldsymbol{T}_1' \left(\boldsymbol{T}_L^i \right)^{-1} \right) \|^2 \\ &+ \| \boldsymbol{v}_1' - (\boldsymbol{R}^i)^T \boldsymbol{v}_L^i \|^2 + \| \boldsymbol{b}_1' - \boldsymbol{b}_L^i \|^2, \end{aligned} \tag{25}$$

$$\begin{aligned} e_R^{EP}(\boldsymbol{T}^i, \boldsymbol{x}_R^i) &= \| \log \left(\boldsymbol{T}^i \boldsymbol{T}_{N^{sub}}' \left(\boldsymbol{T}_R^i \right)^{-1} \right) \|^2 \\ &+ \| \boldsymbol{v}_{N^{sub}}' - (\boldsymbol{R}^i)^T \boldsymbol{v}_R^i \|^2 + \| \boldsymbol{b}_{N^{sub}}' - \boldsymbol{b}_R^i \|^2. \end{aligned} \tag{26}$$

We then create an IMU factor between $\boldsymbol{x}_R^i$ and $\boldsymbol{x}_L^{i+1}$. In this way, an IMU factor covers only a small time interval (range data scan interval) and can strongly constrain the submap poses while avoiding the loss of the velocity and bias information estimated by the local mapping module.

Finally, the objective function for the global trajectory optimization is defined as:

$$\begin{aligned} g^{GM}(\mathcal{X}^g) = &\sum_{(\boldsymbol{T}^i, \boldsymbol{T}^j) \in \mathcal{X}^o} e^{PC}(\boldsymbol{T}^i, \boldsymbol{T}^j) \\ &+ \sum_{\boldsymbol{T}^i \in \mathcal{X}^g} e^{IMU}(\boldsymbol{x}_R^{i-1}, \boldsymbol{x}_L^i) \\ &+ \sum_{\boldsymbol{T}^i \in \mathcal{X}^g} \left(e_L^{EP}(\boldsymbol{T}^i, \boldsymbol{x}_L^i) + e_R^{EP}(\boldsymbol{T}^i, \boldsymbol{x}_R^i) \right), \end{aligned} \tag{27}$$

where $\mathcal{X}^g$ is the set of all submap poses, and $\mathcal{X}^o$ is the pairs of submaps with an overlap.

Whenever a new submap is inserted, the factor graph is incrementally optimized using the iSAM2 optimizer [52].

VI. Experiments

In this section, we first show the robustness of the proposed odometry estimation algorithm to momentary degeneration of range data in simulated and real environments (Sec. VI-A). We then demonstrate that the proposed framework can be applied to various range-IMU sensors (Sec. VI-B). Finally, we compare the accuracy of the proposed framework and those of state-of-the-art methods and present ablation studies on the Multi-Camera Newer College Dataset [57] (Sec. VI-C) and the NTU VIRAL Dataset [58] (Sec. VI-D)[2].

A. Robustness to Degenerated Range Data

Evaluation in simulation: To show that the proposed method is robust to momentary range data degeneration, we conducted an evaluation in the simulation environment shown in Fig. 8. We generated point clouds with a LiDAR model (Velodyne VLP-16) while moving it along the corridor, and synthesized IMU data using OpenVINS [59]. We generated five LiDAR-IMU sequences while changing the IMU linear acceleration and angular velocity noise level ([m/s^2] and [°/s]) from 1.0×10^{-3} to 1.0×10^{-1}. Although the environment was 40 m in width, the maximum observation range of the LiDAR was limited to 15 m so that range data became completely degenerated in the middle of the corridor for a few seconds.

We ran the proposed odometry estimation algorithm and two state-of-the-art LiDAR-IMU odometry methods, namely

[2]The datasets and the estimation results are available online: https://staff.aist.go.jp/k.koide/projects/glimsupp

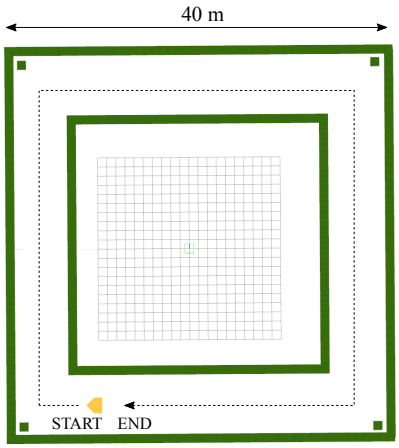
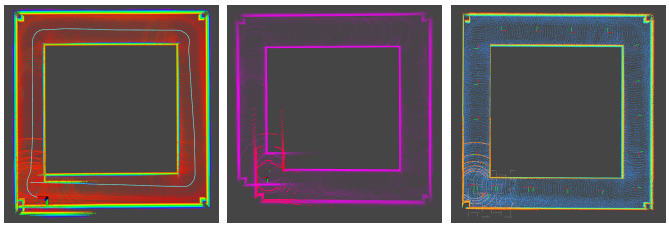


Fig. 8. Simulation environment. The environment is 40 m in width. The LiDAR observation range is limited to 15 m. LiDAR data become completely degenerated in the middle of the corridor.



(a) LIO-SAM [10] (b) FAST-LIO2 [5] (c) GLIM

Fig. 9. Estimation results for lowest IMU noise level (1.0×10^{-3} [m/s^2] and [°/ s]). Existing methods based on frame-to-model matching are sensitive to degenerated LiDAR data whereas the proposed method produced a good trajectory estimation result.

LIO-SAM [10], a loosely coupled LiDAR-IMU odometry method, and FAST-LIO2 [5], a tightly coupled LiDAR-IMU odometry method with an iterated Kalman filter, to evaluate their robustness to range data degeneration. The estimated trajectories were evaluated with the absolute trajectory error (ATE [60]) metric using the *evo* toolkit[3]. Note that loop closure was disabled for all methods.

Fig. 9 shows the estimation results under the lowest IMU noise level (1.0×10^{-3}). LIO-SAM and FAST-LIO, which are based on frame-to-model matching, are sensitive to the momentary degeneration of LiDAR data and thus their estimation results have large errors. Because frame-to-model matching needs to immediately determine and fix the sensor pose to merge the latest point cloud into the target map model, these methods cannot avoid corruption of the map model when point clouds are degenerated.

The proposed method had a small trajectory estimation error under degeneration of LiDAR data. Fig. 10 shows how the velocity estimation error for the proposed method was corrected while the sensor was moving in a corridor. The color of the trajectory encodes the magnitude of velocity errors and the red points show a 2D slice of the latest point cloud. While the velocity estimation error accumulated in the middle of the corridor due to the degenerated point clouds (Fig. 10 (a)), once the matching cost factors became well constrained

[3]https://michaelgrupp.github.io/evo/

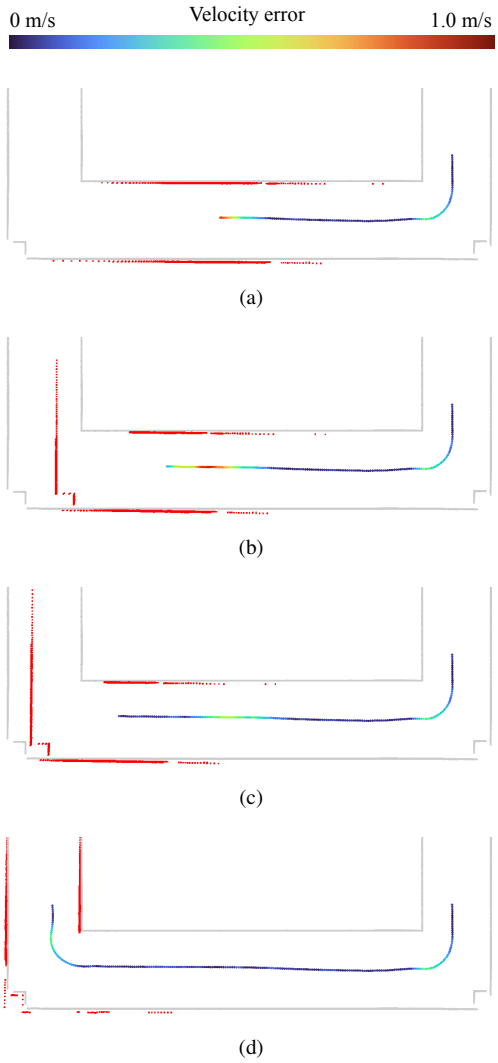


Fig. 10. Correction of velocity estimation error by proposed method for moving sensor. The trajectory color encodes the magnitude of the velocity estimation errors and the red points indicate the latest LiDAR scan. Map points shown in gray are shown to aid visualization. The estimation error accumulated in the middle of the corridor due to the degenerated LiDAR data. Once the opposite wall became observable, the estimation error of the latest frame was corrected. This correction was then propagated to past frames and a correct estimation was recovered at the end.

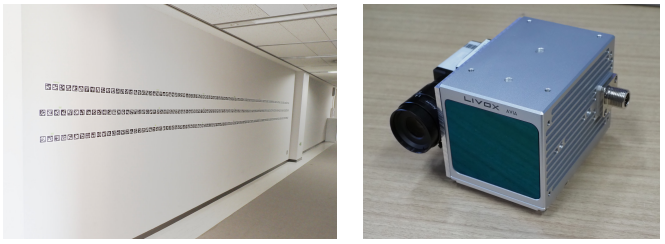
by observing the opposite wall, the estimation of the latest frame was corrected (Fig. 10 (b)). It was then propagated to past frames (Fig. 10 (c)). The velocity estimates of all past frames were corrected and a correct trajectory estimation was recovered at the end (Fig. 10 (d)).

Table I summarizes the ATEs for each method under several IMU noise levels. Even under the lowest IMU noise level (1.0×10^{-3}), LIO-SAM and FAST-LIO2 were sensitive to degenerated LiDAR data and showed large ATEs (1.474 and 1.294 m, respectively). As the IMU noise became larger, the ATEs for these methods also became larger. The output of LIO-SAM became corrupted when the noise level was set to 5.0×10^{-2}. Although FAST-LIO2, which employs iterated-Kalman-filter-based LiDAR-IMU tight coupling, was more robust to IMU noise compared to LIO-SAM, its output became corrupted under the highest IMU noise level (1.0×10^{-1}).

TABLE I
RANGE DATA DEGENERATION TEST RESULTS IN SIMULATION

IMU Noise	Absolute Trajectory Error [m]		
[m/s^2] [°/s]	LIO-SAM [10]	FAST-LIO2 [5]	GLIM
1.0×10^{-3}	1.473 ± 0.696	1.294 ± 0.691	**0.099 ± 0.012**
5.0×10^{-3}	4.670 ± 1.863	1.449 ± 0.843	**0.064 ± 0.029**
1.0×10^{-2}	12.888 ± 5.674	2.495 ± 1.111	**0.133 ± 0.083**
5.0×10^{-2}	✗	11.735 ± 4.224	**0.384 ± 0.163**
1.0×10^{-1}	✗	✗	**1.566 ± 0.839**

✗ indicates that the estimation was corrupted.



(a) Environment (b) Used sensor

Fig. 11. Experimental environment and used sensors. The LiDAR system was moved between two pillars while facing the flat wall so the LiDAR data became degenerated in the middle of the path. The fiducial tags and camera data were used for acquiring ground truth trajectories.

The proposed method showed the best ATEs among the compared methods under all IMU noise level settings (0.099 to 1.566 m) thanks to its combination of fixed-lag smoothing and keyframe-based point cloud matching. Although its estimation error gradually increased as the IMU noise became larger, the proposed method had a reasonable ATE (1.566 m) even under the highest IMU noise level (1.0×10^{-1}).

Evaluation in real environment: To show that the proposed method can deal with range data degeneration in real situations, we recorded eight LiDAR-IMU data sequences in the environment shown in Fig. 11 with a non-repetitive scan LiDAR (Livox Avia). The LiDAR was moved between two pillars while facing the flat wall, and thus the LiDAR data were degenerated in the middle of the path for each sequence. To obtain the ground truth sensor trajectories, we placed fiducial tags (AprilTag [62]) on the wall and estimated their poses using full bundle adjustment in advance. For each sequence, we recorded images using a camera (OMRON SENTECH STC-MBS202POE) along with LiDAR-IMU data. We estimated the camera trajectory via batch optimization of the tag projection and IMU constraints as a ground truth trajectory. The LiDAR and camera were synchronized with the host PC via the IEEE 1588 protocol (PTP) and the transformation between them was calibrated using a normalized-information-distance-based direct LiDAR-camera alignment method [63].

In addition to LIO-SAM [10] and FAST-LIO2 [5], we ran VoxelMap [61], a tightly coupled LiDAR-IMU odometry estimation with adaptive resolutional voxel mapping.

Table II summarizes the ATEs for the proposed method and FAST-LIO2 and VoxelMap. Note that we could not obtain decent estimation results with LIO-SAM [10]. This was likely due to LIO-SAM employing a loosely coupled LiDAR-

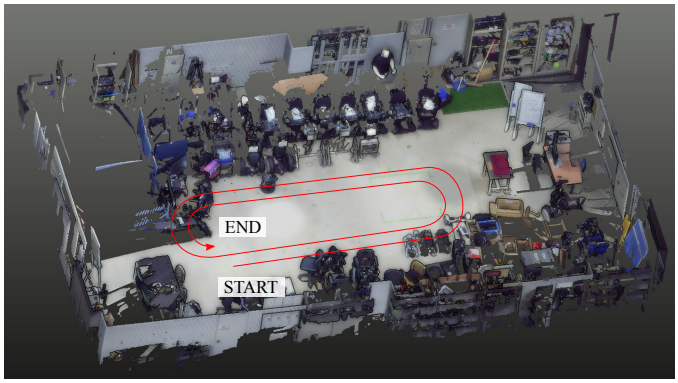
TABLE II
RANGE DATA DEGENERATION TEST RESULTS IN REAL ENVIRONMENT

Seq.	Path length [m]	Duration [s]	Velocity (Avg / Max)		Absolute Trajectory Error [m]		
			Linear [m/s]	Angular [°/s]	FAST-LIO2 [5]	VoxelMap [61]	GLIM
01	4.106	18.5	0.22 / 0.55	11.6 / 39.9	0.815 ± 0.500	0.577 ± 0.157	**0.118 ± 0.047**
02	4.192	13.6	0.32 / 0.62	13.4 / 33.6	0.822 ± 0.255	**0.146 ± 0.057**	0.299 ± 0.116
03	4.766	14.4	0.33 / 0.65	18.0 / 43.7	0.873 ± 0.364	0.950 ± 0.263	**0.040 ± 0.015**
04	4.213	13.1	0.33 / 0.62	14.1 / 51.3	1.137 ± 0.412	0.586 ± 0.299	**0.389 ± 0.145**
05	4.121	14.0	0.30 / 0.69	17.1 / 50.0	1.048 ± 0.582	0.786 ± 0.400	**0.228 ± 0.083**
06	3.725	8.8	0.43 / 0.79	16.0 / 42.7	15.551 ± 8.780	0.807 ± 0.309	**0.056 ± 0.023**
07	3.086	8.9	0.35 / 0.64	17.6 / 53.1	0.635 ± 0.218	0.366 ± 0.111	**0.017 ± 0.007**
08	2.229	12.1	0.19 / 0.49	14.2 / 31.7	0.297 ± 0.110	0.279 ± 0.082	**0.146 ± 0.066**

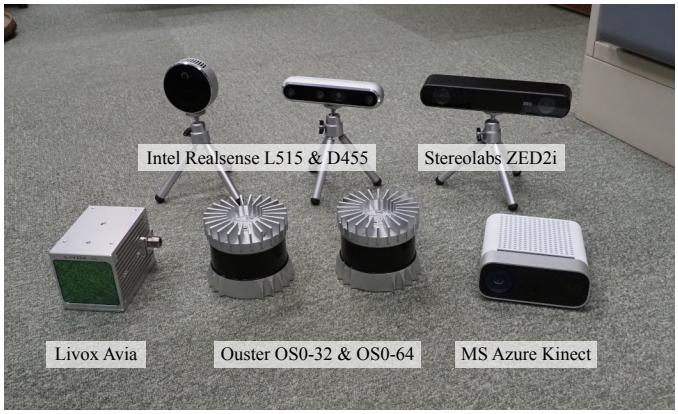
TABLE III
CROSS SENSOR EVALUATION

Sensor	ATE [m]	RTE [m]
Ouster OS0-32	0.037 ± 0.018	0.037 ± 0.017
Ouster OS0-64	0.022 ± 0.009	0.025 ± 0.012
Livox Avia	0.041 ± 0.014	0.043 ± 0.019
Microsoft Azure Kinect	0.007 ± 0.003	0.007 ± 0.003
Intel Realsense L515	0.042 ± 0.018	0.045 ± 0.020
Intel Realsense D455	0.206 ± 0.086	0.331 ± 0.161
Stereolabs ZED2i	0.139 ± 0.062	0.194 ± 0.102

ATE: absolute trajectory error; RTE: relative trajectory error



(a) Experimental environment



(b) Range-IMU sensors

Fig. 12. Experimental environment and used range-IMU sensors. The proposed method was validated with various range-IMU sensors with different sensing mechanisms.

IMU fusion approach, in which point cloud scan matching is separately performed without IMU constraints. LIO-SAM thus has difficulty handling severely degenerated LiDAR data. From Table II, we can see that FAST-LIO2 produced worse estimation results (0.297 to 15.551 m). Although VoxelMap showed better estimation results compared to those of FAST-LIO2 (0.146 to 0.950 m) thanks to its adaptive environment representation and data association, it also largely suffered from degenerated LiDAR data.

This reveals the weakness of state-of-the-art methods based on frame-to-model matching. As discussed in [5], FAST-LIO2 (and other methods based on frame-to-model matching) cannot deal with situations where point clouds are completely degenerated (e.g., when the LiDAR is facing a flat wall, the sky, or the ground or is close to objects).

The proposed method had better ATEs than those for FAST-LIO2 and VoxelMap for most of the sequences (0.017 to 0.389 m). We refer the reader to the supplementary page, which clearly demonstrates how the proposed fixed-lag smoothing approach behaves and corrects estimation drift by actively updating past sensor poses.

B. Mapping with Various Range-IMU Sensors

To demonstrate the versatility of the proposed framework, we conducted mapping experiments with various range-IMU sensors with different sensing mechanisms including spinning LiDAR (Ouster OS0-32 and OS0-64), non-repetitive scan LiDAR (Livox Avia), time-of-flight (ToF) depth camera (Microsoft Azure Kinect), solid-state LiDAR (Intel Realsense L515), active stereo camera (Intel Realsense D455), and passive stereo camera (Stereolabs ZED2i). Fig. 12 shows the experimental environment and the used range-IMU sensors. Similar to the ground truth acquisition protocol used in the Newer College Dataset [64], we estimated sensor trajectories by aligning point clouds with an environment point cloud recorded with a survey-grade LiDAR (FARO Focus).

We applied the proposed method to the recorded range-IMU sequences and evaluated the trajectory estimation errors using the ATE and relative trajectory error (RTE) metrics [60]. We set the sub-trajectory length of the RTE evaluation to 2 m. To show the generality of the proposed algorithm, we used the same mapping parameter set for all sensors. Note that we also tested FAST-LIO2 [5]; however, we could not obtain reasonable results for sensors other than the Ouster and Livox

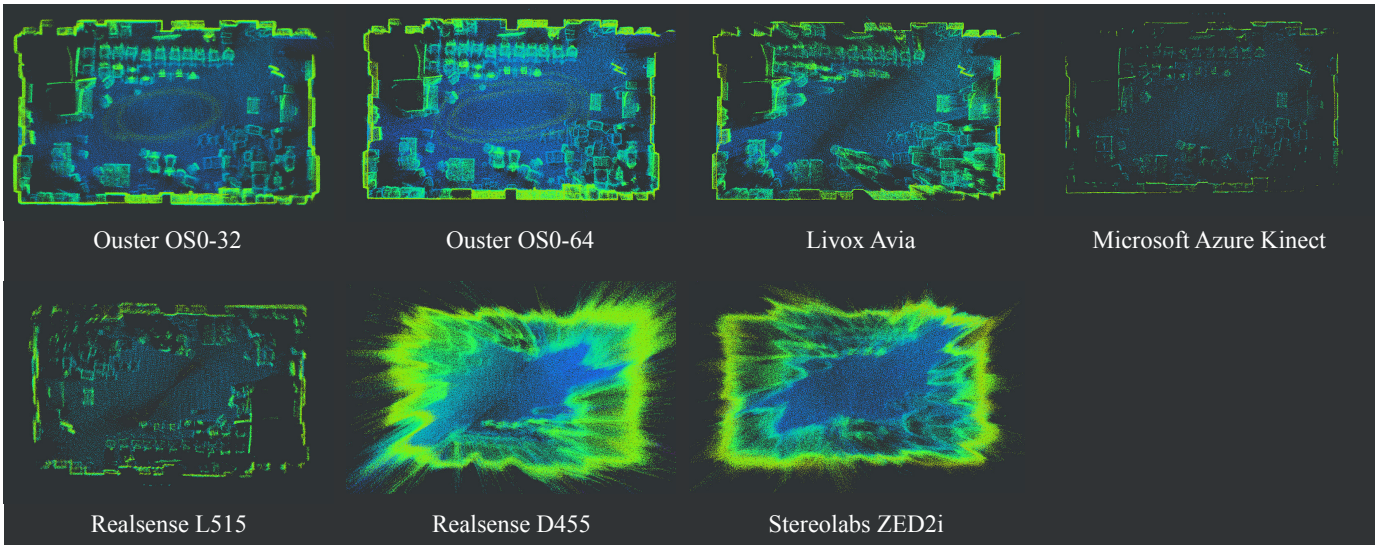


Fig. 13. Mapping results obtained with various range-IMU sensors. The proposed method achieved consistent mapping results for all sensors. The same mapping parameter set was used for all sensors.

LiDARs.

Table III summarizes the trajectory estimation errors. The proposed method showed accurate estimation results for the LiDARs and the ToF depth camera (OS0-32, OS0-64, Livox Avia, Azure Kinect, Realsense L515), which provided accurate point cloud data (ATE: 0.007 to 0.042 m, RTE: 0.007 to 0.045 m). Although the estimation results slightly deteriorated for the stereo-based cameras (Realsense D455 and Stereolabs ZED2i) due to strongly distorted point clouds, the estimation errors were still at a reasonable level (ATE: 0.139 and 0.206 m, RTE: 0.194 and 0.331 m).

Fig. 13 shows the mapping results. For all sensors, the proposed method achieved consistent mapping results. Although the mapping results for stereo-based sensors were distorted because of the large distortion of the input point clouds, we obtained consistent maps without doubled walls and floors.

These results show the robustness of the proposed odometry estimation algorithm to noise on point clouds. Conventional odometry estimation methods based on feature-based point cloud matching are sensitive to such large noise in input point clouds, which makes the extraction of edge and plane feature points difficult. Because the proposed odometry estimation algorithm is based on direct distribution-to-distribution point cloud matching, which avoids noise-sensitive feature extraction and employs probabilistic surface modeling, it can robustly deal with noisy point clouds.

Furthermore, these results demonstrate the flexibility of the proposed global optimization algorithm based on global matching cost minimization. Because we cannot expect reasonable frame-by-frame scan matching results for the noisy point clouds of stereo-based sensors, existing methods based on pose graph optimization have difficulty accurately closing loops. In contrast, the global matching cost minimization approach does not explicitly require the relative pose between frames for each factor. Because it directly minimizes multi-scan registration errors on the factor graph, it is robust to such frame-by-frame matching failures.

C. Quantitative Estimation Accuracy Evaluation on the Newer College Dataset

We quantitatively evaluated the estimation accuracy of the proposed framework on the Multi-Camera Newer College Dataset [57], [64]. This dataset provides sequences of LiDAR-camera-IMU data recorded with an Ouster OS0-128 (10-Hz point clouds and 100-Hz IMU) and a Sevensense Alphasense Core (30-Hz images $\times$ 4 hardware-synchronized cameras) in several indoor and outdoor environments. The LiDAR and cameras were synchronized via the IEEE 1588 protocol (PTP). Table IV summarizes the data duration and path length of each sequence in the dataset.

Comparison with state-of-the-art: We compared the trajectory estimation accuracy of the proposed method with those of state-of-the-art LiDAR-IMU SLAM methods, including a loosely coupled LiDAR-IMU method (LIO-SAM [10]), tightly coupled methods based on an iterated Kalman filter (LINS [3], FAST-LIO2 [5]), a tightly coupled continuous SLAM method (CLINS [11]), and a loosely coupled method based on GICP matching (DLO [14]). Because LINS, FAST-LIO2, and DLO do not have a loop closure mechanism, only odometry estimation errors were evaluated for these methods. We evaluated the estimation errors with the translational ATE and RTE (delta = 10 m) metrics.

Tables V and VI respectively summarize the ATEs and RTEs for the evaluated methods. Among the existing methods, FAST-LIO2 showed the best ATEs and RTEs for most of the sequences owing to the robust state estimation based on tightly coupled range and IMU data fusion on an iterated Kalman filter. However, the ATEs and RTEs for FAST-LIO2 and the other methods based on point-to-plane (and point-to-edge) point cloud matching (LINS, LIO-SAM, and CLINS) deteriorated significantly for the *stairs* sequence in a small indoor environment.

TABLE IV
DATA DURATIONS AND PATH LENGTHS FOR MULTI-CAMERA NEWER COLLEGE DATASET

Sequence	quad easy	quad medium	quad hard	stairs	park	cloister	math easy	math medium	math hard
Duration [s]	198.7	190.6	187.8	118.9	1572.0	278.6	215.9	176.9	243.7
Path length [m]	246.7	260.4	234.8	57.0	2396.2	428.8	263.6	304.3	320.6

TABLE V
ABSOLUTE TRAJECTORY ERROR [M] RESULTS FOR VARIOUS METHODS ON MULTI-CAMERA NEWER COLLEGE DATASET

Method	Loop closure	quad easy	quad medium	quad hard	stairs	park	cloister	math easy	math medium	math hard	Average
LINS [3]		0.160	0.212	16.824	3.405	0.612	1.170	0.216	0.259	5.710	3.174
LIO-SAM [10]		0.086	**0.069**	0.105	3.438	1.381	0.085	**0.088**	**0.114**	0.089	0.606
	✓	0.083	0.075	0.099	2.942	**0.519**	0.082	**0.083**	**0.111**	0.092	0.454
FAST-LIO2 [5]		**0.068**	**0.059**	**0.050**	1.320	**0.319**	**0.078**	0.091	0.134	**0.058**	0.242
CLINS [11]		0.197	0.674	0.151	2.336	18.265	0.293	0.225	0.235	0.494	2.541
	✓	0.252	0.451	0.140	2.321	2.757	0.262	0.241	0.242	0.484	0.795
DLO [14]		**0.080**	0.100	0.168	**0.129**	0.686	0.163	0.161	0.211	0.192	**0.210**
GLIM		**0.070**	**0.061**	**0.044**	**0.106**	0.459	**0.063**	0.096	**0.119**	**0.064**	**0.120**
	✓	**0.070**	**0.059**	**0.043**	**0.046**	**0.269**	**0.056**	**0.082**	**0.119**	**0.048**	**0.088**

Red, **blue**, and **black** bold values are respectively the **best**, **second best**, and **third best** results.

TABLE VI
RELATIVE TRAJECTORY ERROR [M] RESULTS FOR VARIOUS METHODS ON MULTI-CAMERA NEWER COLLEGE DATASET

Method	Loop closure	quad easy	quad medium	quad hard	stairs	park	cloister	math easy	math medium	math hard	Average
LINS [3]		0.328	0.463	0.996	4.592	0.435	0.923	0.306	0.523	0.887	1.050
LIO-SAM [10]		**0.143**	0.172	0.201	2.662	0.174	0.109	0.129	**0.223**	0.246	0.451
	✓	**0.151**	0.195	0.209	2.630	0.205	0.117	**0.118**	**0.216**	0.211	0.450
FAST-LIO2 [5]		**0.150**	**0.161**	**0.152**	1.137	**0.090**	**0.103**	0.146	0.293	**0.102**	0.259
CLINS [11]		0.177	0.271	0.196	3.070	0.291	0.149	0.145	0.276	**0.120**	0.522
	✓	0.202	0.486	**0.194**	2.933	0.157	0.155	0.145	0.282	0.163	0.524
DLO [14]		0.222	0.328	0.365	**0.192**	0.233	0.240	**0.053**	**0.108**	0.379	**0.236**
GLIM		**0.151**	**0.168**	**0.164**	**0.169**	**0.096**	**0.100**	0.125	0.250	0.137	**0.151**
	✓	0.154	**0.170**	**0.164**	**0.096**	**0.095**	**0.107**	**0.123**	0.243	**0.135**	**0.143**

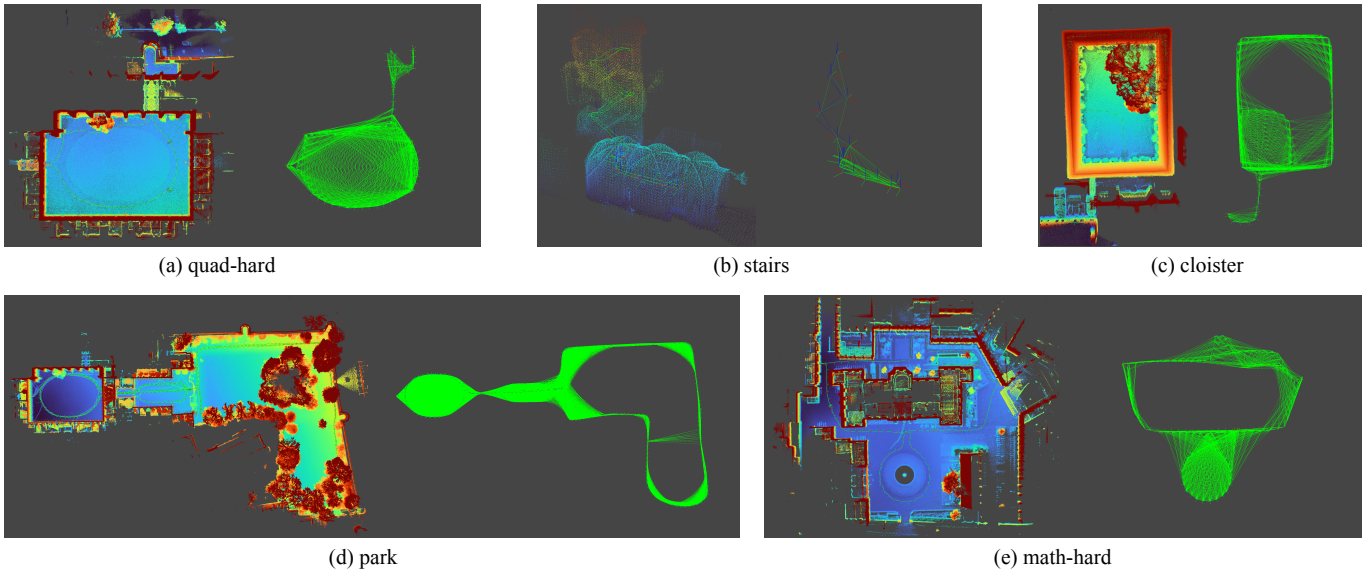
Red, **blue**, and **black** bold values are respectively the **best**, **second best**, and **third best** results.

TABLE VII
PROCESSING TIME FOR *park* SEQUENCE

Module	Process	Time [ms]
Preprocess	Downsampling	8.9 ± 1.5
	kNN search	5.7 ± 0.4
	Total (per frame)	15.1 ± 1.6
Odometry estimation	Factor creation	3.3 ± 0.3
	Optimization	25.7 ± 30.5
	Total (per frame)	29.3 ± 30.5
Local mapping	Factor creation	12.3 ± 2.0
	Total (per frame)	12.4 ± 2.0
	Optimization	295.0 ± 111.0
	Total (per submap)	295.0 ± 111.0
Global mapping	Factor creation	20.1 ± 10.3
	Optimization	35.6 ± 51.4
	Total (per submap)	55.8 ± 55.6

We consider this to be due to point-to-plane-based matching requiring precise point association and needing careful tuning of parameters to deal with environment changes. It is worth mentioning that although we fine-tuned the parameters of these methods to improve the results on the *stairs* sequence, we could not obtain an improvement without sacrificing accuracy for the other sequences in outdoor environments. DLO showed better ATE and RTE for the *stairs* sequence because it employs a direct scan matching algorithm based on GICP, which does not require feature extraction and is robust to environment changes. However, the overall accuracy of DLO was slightly worse than that of FAST-LIO2.

The proposed odometry estimation algorithm showed comparable ATEs and RTEs to those for FAST-LIO2 for most of the sequences. For the *stairs* sequence, it showed the best estimation result among all evaluated methods. This suggests that the proposed method can deal with a large variety of environments while suppressing estimation drift. With loop closure, the ATEs and RTEs for the proposed method were further improved, and we achieved the best ATEs for most of the sequences. Fig. 14 shows the map point clouds and factor graphs created with the proposed framework. We can



(a) quad-hard (b) stairs (c) cloister

(d) park (e) math-hard

Fig. 14. Mapping results and factor graphs for Multi-Camera Newer College Dataset. Highly consistent environmental maps were created via global matching cost minimization on extremely dense factor graphs.

TABLE VIII
ABSOLUTE TRAJECTORY ERROR [M] RESULTS FOR ODOMETRY ESTIMATION ABLATION STUDY

Method	quad easy	quad medium	quad hard	stairs	park	cloister	math easy	math medium	math hard	Average
GLIM odometry (baseline)	0.070	0.061	0.044	0.106	0.459	0.063	0.096	0.119	0.064	0.120
w/o Surface validation	0.070 (+0.000)	0.061 (+0.000)	**0.043 (-0.001)**	**0.114 (+0.008)**	**0.533 (+0.074)**	**0.061 (-0.002)**	**0.140 (+0.044)**	**0.147 (+0.027)**	**0.097 (+0.033)**	**0.140 (+0.020)**
w/o Multi resolution	**0.078 (+0.008)**	**0.092 (+0.031)**	**0.071 (+0.027)**	**0.073 (-0.033)**	**1.096 (+0.637)**	**0.355 (+0.292)**	**0.127 (+0.031)**	**0.140 (+0.021)**	**0.074 (+0.010)**	**0.234 (+0.114)**
w/ Multi-cam constraints	**0.069 (-0.001)**	**0.059 (-0.002)**	**0.040 (-0.004)**	**0.045 (-0.061)**	**0.418 (-0.041)**	**0.053 (-0.010)**	**0.095 (-0.001)**	**0.122 (+0.003)**	**0.055 (-0.009)**	**0.106 (-0.014)**

Red and **blue** values are respectively **improved** and **deteriorated** results.
Values in parentheses indicate absolute trajectory error differences from those of the baseline.

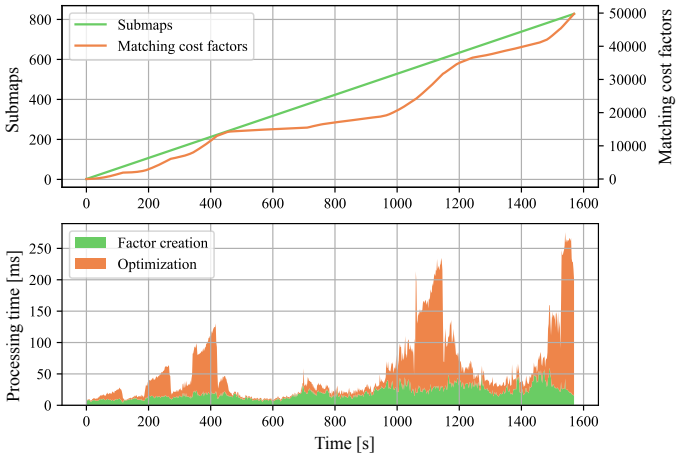


Fig. 15. Number of submaps and matching cost factors and processing time of global optimization module for *park* sequence. Owing to the efficient and incremental optimization scheme, the proposed method achieved a real-time processing.

see that the proposed method created an extremely dense factor graph for global matching cost minimization, which resulted in consistent mapping results.

Processing time: Table VII summarizes the processing times, measured with an Intel Core i7 8700K and an NVIDIA RTX 1660 Ti, for each module in the proposed framework running through the *park* sequence, which is the longest sequence in the dataset.

The preprocessing and odometry estimation modules respectively took 15.1 and 29.3 ms per frame on average and are thus sufficiently faster than the real-time requirement (100 ms per frame). The submap optimization, which was performed approximately every 2 s, took 295.0 ms on average. The global map optimization took 55.8 ms on average for each submap creation. Fig. 15 shows how the global optimization time grew as the number of submaps and matching cost factors increased. Although a massive number of matching cost factors were created (over 50,000 factors between 800 submaps), the global optimization converged in less than 50 ms for most of the frames thanks to the GPU-accelerated matching cost factor and iSAM2, which efficiently updates only sub Bayes trees that can be influenced by the insertion of a new submap. Although the global optimization took a longer time when closing a large loop, the maximum optimization time was only about

TABLE IX
ABSOLUTE TRAJECTORY ERROR [M] RESULTS FOR GLOBAL OPTIMIZATION ABLATION STUDY

Method	quad easy	quad medium	quad hard	stairs	park	cloister	math easy	math medium	math hard	Average
GLIM (Baseline)	0.070	0.059	0.043	0.046	0.269	0.056	0.082	0.119	0.048	0.088
w/o IMU factors	0.070 (+0.000)	0.059 (+0.000)	**0.045 (+0.002)**	**0.054 (+0.008)**	**0.268 (-0.001)**	**0.217 (+0.161)**	**0.081 (-0.001)**	0.119 (+0.000)	0.048 (+0.000)	**0.107 (+0.019)**
w/ IMU factors between submaps	0.070 (+0.000)	0.059 (+0.000)	0.043 (+0.000)	**0.054 (+0.008)**	**0.268 (-0.001)**	**0.341 (+0.285)**	**0.082 (+0.000)**	0.119 (+0.000)	0.048 (+0.000)	**0.120 (+0.032)**

Red and **blue** values are respectively **improved** and **deteriorated** results.
Values in parentheses indicate absolute trajectory error differences from those of the baseline.

250 ms, which is much smaller than the real-time requirement (2 s). Note that the linearization of the matching cost factors accounted for most of the optimization time; the linear solver performed on the CPU accounted for only about 5% of the total optimization time.

Odometry estimation ablation study: To examine the effects of surface-orientation-based correspondence validation and a multi-resolution voxelmap, we ran the proposed framework using the same settings used in the previous experiment but without these functionalities. We also ran the proposed odometry estimation algorithm with the multi-camera constraints to show that we can improve the mapping accuracy in real situations by introducing additional constraints via the global callback slot mechanism. Table VIII summarizes the evaluation results.

Without the surface-orientation-based correspondence validation, the ATEs deteriorated for about half of the sequences (*stairs, park, math-easy, math-medium*, and *math-hard*) and the average ATE increased from 0.120 to 0.140 m. We consider this to be due to the points on a surface being possibly wrongly associated with points on the other side of the surface in another frame when correspondence validation is not performed.

Without the multi-resolution voxelmap, we observed an accuracy improvement for the *stairs* sequence (ATE decreased from 0.106 to 0.073 m). However, for all other sequences, the ATEs largely deteriorated, resulting in a deteriorated average ATE (0.234 m). The ATEs tend to be large for large outdoor environments (e.g., *park* and *cloister*). The results indicate that the multi-resolution voxelmap enables a robust association of points and voxels in a wide variety of environments at the cost of a slight accuracy decrease in small indoor environments.

With the multi-camera visual constraints, the ATEs improved for all sequences except for *math-medium* and the average ATEs decreased from 0.120 to 0.106 m. The ATE improvements were marginal in sequences where rich geometric features were available and scan matching was well constrained. The ATEs for the *stairs*, *park*, *cloister*, and *math-hard* sequences, where point cloud matching might be unstable due to environment scale changes and aggressive sensor motion, were largely improved with the tight coupling of visual constraints.

Global trajectory optimization ablation study: We evaluated the effect of the proposed *endpoint* mechanism for stabilizing submap pose optimization with IMU constraints. We ran the proposed framework with two settings: 1) without IMU constraints and 2) with IMU constraints created directly between submap states. Table IX summarizes the evaluation results. We can see that for both settings, the ATEs deteriorated significantly for the *cloister* sequence. When the framework was running on the *cloister* sequence, the optimizer had indeterminant linear system errors, which indicate that the factor graph was under-constrained. Even when we inserted IMU factors directly between submap states, the optimizer still suffered from an under-constrained system. This is because submaps were created with a large time interval and the IMU factors created between them had a large integration time, resulting in large uncertainty. This shows the advantage of the proposed *endpoint* mechanism, which can strongly constrain the submap poses because each IMU factor covers only a minimum scan interval by being created between the *endpoints* of consecutive submaps.

D. Quantitative Estimation Accuracy Evaluation on the NTU VIRAL Dataset

We further compared the accuracy of the proposed method with those of state-of-the-art methods in a more challenging environment on the NTU VIRAL dataset [58]. This dataset provides time-synchronized sequences of two LiDARs (Ouster OS1-16), two cameras (uEye 1221 LE), an IMU (VectorNav VN100), and a UWB (Humatic P440) on a UAV. Because it was recorded with dynamic 3D motion of a flying UAV, we consider evaluation on this dataset would reveal the robustness of state-of-the-art methods in challenging situations.

We ran the proposed method with three configurations (LiDAR-IMU, visual-LiDAR-IMU, LiDAR-IMU with loop closure) and compared its accuracy with those of state-of-the-art LiDAR-IMU odometry estimation methods (LIO-SAM [10], MLOAM [65], FAST-LIO2 [5], VoxelMap [61], and BALM [39]), visual-LiDAR-IMU odometry estimation methods (VIRAL-SLAM [66] and FAST-LIVO [67]), and a LiDAR-IMU mapping method with loop closure (SLICT [68]). We evaluated the translational ATEs of those methods by using the evaluation code provided by the dataset[4].

Table X summarizes the ATEs for the evaluated methods. The ATEs for LIO-SAM, MLOAM, VIRAL-SLAM, and SLICT were taken from [66] and [68].

[4]https://github.com/ntu-aris/viral_eval

TABLE X
RELATIVE TRAJECTORY ERROR [M] RESULTS FOR VARIOUS METHODS ON NTU VIRAL DATASET

Method	Camera	Loop closure	eee 01	eee 02	eee 03	nya 01	nya 02	nya 03	sbs 01	sbs 02	sbs 03	Average
LiDAR-based methods												
LIO-SAM [10]			0.075	0.069	0.101	0.076	0.090	0.137	0.089	0.083	0.140	0.096
MLOAM [65] †			0.249	0.166	0.232	0.123	0.191	0.226	0.173	0.147	0.153	0.184
FAST-LIO2 [25]			0.056	0.031	0.047	0.041	0.047	0.036	**0.036**	0.033	**0.034**	0.040
VoxelMap [61]			0.075	0.033	0.070	0.048	0.040	0.050	0.041	0.072	0.058	0.054
BALM [39]			0.060	**0.021**	**0.028**	**0.031**	0.037	0.028	0.041	0.054	0.055	0.039
GLIM			**0.034**	0.030	0.032	**0.031**	**0.028**	**0.027**	0.037	**0.025**	**0.034**	**0.031**
Visual-LiDAR-based methods												
VIRAL-SLAM [66] †	✓		0.060	0.058	0.037	0.051	0.043	0.032	0.048	0.062	0.054	0.049
FAST-LIVO [67]	✓		0.077	**0.020**	0.032	0.034	0.044	**0.025**	0.039	0.044	0.039	0.039
GLIM	✓		**0.032**	0.026	**0.030**	**0.031**	**0.028**	0.026	**0.037**	**0.025**	**0.033**	**0.030**
LiDAR-based methods with loop closure												
SLICT [68]		✓	0.032	0.025	0.028	**0.023**	**0.023**	0.026	**0.030**	0.029	0.034	0.028
GLIM		✓	**0.025**	**0.018**	**0.023**	0.025	0.027	**0.025**	0.031	**0.024**	**0.027**	**0.025**

Black bold values are the best results for each configuration.
† Both the horizontal and vertical LiDARs are used.

Among the existing LiDAR-IMU-based methods, FAST-LIO2 showed accurate estimation results (Average ATE: 0.040 m) owing to its efficient fusion of LiDAR and IMU constraints. We can also see that BALM also showed comparable results (0.039 m) owing to its BA-based optimization ensuring consistent mapping results. The proposed method showed the best ATEs for six out of the nine sequences that resulted in the best average ATE (0.031 m) among LiDAR-IMU-based methods. We consider the fixed-lag-smoothing-based optimization enabled dealing with the UAV's dynamic motion robustly and achieved smooth and accurate trajectory estimation results.

With visual constraints, the proposed method improved the average ATE (0.030 m) that was the best results among the evaluated visual-LiDAR-IMU-based methods. This result confirms that the proposed tight fusion of visual constraints enable enhancing the estimation accuracy without sacrificing the accuracy of the LiDAR-IMU-based estimation.

Finally, with loop closure, the ATE of the proposed method got further improved to 0.025 m that was better than the ATE of SLICT (0.028 m). We observed large accuracy gain in large outdoor sequences that demonstrate the effectiveness of the proposed global trajectory optimization to compensate for odometry estimation drift in challenging situations.

VII. DISCUSSION AND FUTURE WORK

Although we showed the robustness of the proposed framework to momentary degeneration of range data, it is still challenging for the proposed range-IMU odometry estimation to deal with long-term range data degeneration because we need to bound the duration of the optimization window of the odometry estimation for real-time performance. There are two possible approaches for making the system robust to long-term degeneration. The first approach is to incorporate an additional data source (e.g., a camera, radar, or wheel odometry) to suppress the estimation drift during the degeneration of one sensor. This approach has been successfully applied to visual-range-IMU methods [67]. The second approach is to incorporate learning-based motion estimation into the odometry estimation. Recent pedestrian dead reckoning methods [69], [70] show that we can efficiently reduce trajectory estimation drift using only IMU data with a learning-based motion estimation model. These models enable the injection of prior knowledge of the motion pattern to reduce the uncertainty of the IMU-based velocity estimation during range data degeneration. We believe the extensibility of GLIM makes it easy to implement and evaluate both approaches for dealing with long-term range data degeneration.

Regarding global trajectory optimization, the experimental results show that the proposed GPU-accelerated global optimization algorithm can sufficiently handle practical situations. However, for more large-scale mapping problems (e.g., city- or nation-scale mapping), a more scalable optimization algorithm is needed. Distributed and GPU-suitable optimization algorithms such as Gaussian belief propagation [71] are promising approaches for handling extremely large problems. The proposed GPU-based registration error factor would nicely fit with such GPU-based optimization algorithms.

VIII. CONCLUSION

This article presented GLIM, a 3D range-IMU SLAM framework with GPU acceleration. By fully leveraging the computation power of a modern GPU, we proposed odometry estimation and global trajectory optimization algorithms that enable robust and accurate localization and mapping in challenging situations. The experimental results show that state-of-the-art methods based on frame-to-model matching have difficulty dealing with completely degenerated range data whereas the proposed method can robustly estimate the sensor trajectory in such situations.

REFERENCES

[1] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, Feb. 2016.

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

[3] 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 *IEEE International Conference on Robotics and Automation*. IEEE, May 2020, pp. 8899–8906.

[4] 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, Apr. 2022.

[5] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Transactions on Robotics*, pp. 1–21, Jan. 2022.

[6] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, Dec. 2010.

[7] D. Landry, F. Pomerleau, and P. Giguere, "CELLO-3D: Estimating the covariance of ICP in the real world," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2019, pp. 8190–8196.

[8] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent and tightly coupled 3d LiDAR inertial mapping," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2022, pp. 5622–5628.

[9] 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*. IEEE, Oct. 2018, pp. 4758–4765.

[10] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2020, pp. 5135–5142.

[11] J. Lv, K. Hu, J. Xu, Y. Liu, X. Ma, and X. Zuo, "CLINS: Continuous-time trajectory estimation for lidar-inertial system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Dec. 2021, pp. 6657–6663.

[12] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.

[13] J. Wang, M. Xu, F. Foroughi, D. Dai, and Z. Chen, "FasterGICP: Acceptance-rejection sampling based 3d lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 255–262, Jan. 2022.

[14] K. Chen, B. T. Lopez, A. akbar Agha-mohammadi, and A. Mehta, "Direct LiDAR odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, Apr. 2022.

[15] Z. Wang, L. Zhang, Y. Shen, and Y. Zhou, "D-LIOM: Tightly-coupled direct LiDAR-inertial odometry and mapping," *IEEE Transactions on Multimedia*, pp. 1–1, Apr. 2022.

[16] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems V*. Robotics: Science and Systems Foundation, June 2009, pp. 435–443.

[17] 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, Aug. 2018.

[18] L. V. Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2018, pp. 2510–2517.

[19] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, pp. 1–17, May 2021.

[20] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 4531–4537.

[21] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 721–738, Aug. 2013.

[22] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2021, pp. 8729–8736.

[23] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, and A. akbar Aghamohammadi, "LOCUS: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 421–428, Apr. 2021.

[24] A. Reinke, M. Palieri, B. Morrell, Y. Chang, K. Ebadi, L. Carlone, and A.-A. Agha-Mohammadi, "LOCUS 2.0: Robust and computationally efficient lidar odometry for real-time 3d mapping," *IEEE Robotics and Automation Letters*, pp. 1–8, June 2022.

[25] 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, Apr. 2021.

[26] 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, July 2021.

[27] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, Feb. 2012.

[28] 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, Feb. 2017.

[29] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2022.

[30] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3d lidar-based online mapping," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2018.

[31] C. L. Gentil, T. Vidal-Calleja, and S. Huang, "IN2LAAMA: Inertial lidar localization autocalibration and mapping," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 275–290, Feb. 2021.

[32] G. Cioffi, T. Cieslewski, and D. Scaramuzza, "Continuous-time vs. discrete-time vision-based SLAM: A comparative study," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2399–2406, Apr. 2022.

[33] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *IEEE International Conference on Robotics and Automation*. IEEE, Apr. 2007, pp. 3167–3172.

[34] T. M. Iversen, A. G. Buch, and D. Kraft, "Prediction of ICP pose uncertainties using monte carlo simulation with synthetic depth images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2017, pp. 4640–4647.

[35] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2016, pp. 1271–1278.

[36] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent 3D LiDAR mapping with GPU-accelerated GICP matching cost factors," *IEEE Robotics and Automation Letters*, pp. 1–8, Sept. 2021.

[37] L. Zhou, S. Wang, and M. Kaess, "π-LSAM: LiDAR smoothing and mapping with planes," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2021, pp. 5751–5757.

[38] D. Wisth, M. Camurri, and M. Fallon, "VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 309–326, Feb. 2023.

[39] Z. Liu and F. Zhang, "BALM: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, Apr. 2021.

[40] 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, Oct. 2021.

[41] 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, Mar. 2023.

[42] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, Oct. 1997.

[43] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, Feb. 2008.

[44] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "A heuristic loop closing technique for large-scale 6D SLAM," *Automatika*, vol. 52, no. 3, pp. 199–222, Jan. 2011.

[45] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, Jan. 2020.

[46] R. Mur-Artal and J. D. Tardos, "Visual-Inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, Apr. 2017.

[47] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "Maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, July 2018.

[48] V. Usenko, N. Demmel, D. Schubert, J. Stuckler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, Apr. 2020.

[49] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2021, pp. 11 054–11 059.

[50] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–11, Nov. 2013.

[51] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2003, pp. 2743–2748.

[52] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, Dec. 2011.

[53] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam)

[54] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[55] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno, "VITAMIN-e: VIsual tracking and MappINg with extremely dense feature points," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2019, pp. 9633–9642.

[56] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, July 1944.

[57] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera lidar inertial extension to the newer college dataset," arXiv:2112.08854, 2021.

[58] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "NTU VIRAL: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 270–280, Nov. 2021.

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

[60] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2018, pp. 7244–7251.

[61] 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, July 2022.

[62] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2016, pp. 4193–4198.

[63] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023.

[64] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld LiDAR, inertial and vision with ground truth," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2020, pp. 4353–4360.

[65] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust odometry and mapping for multi-LiDAR systems with online extrinsic calibration," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 351–371, Feb. 2022.

[66] T.-M. Nguyen, S. Yuan, M. Cao, T. H. Nguyen, and L. Xie, "Viral slam: Tightly coupled camera-imu-uwb-lidar slam," May 2021.

[67] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "FAST-LIVO: Fast and tightly-coupled sparse-direct LiDAR-inertial-visual odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2022.

[68] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping," Nov.

[69] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2020.

[70] Q. Wang, H. Luo, J. Wang, L. Sun, Z. Ma, C. Zhang, M. Fu, and F. Zhao, "Recent advances in pedestrian navigation activity recognition: A review," *IEEE Sensors Journal*, vol. 22, no. 8, pp. 7499–7518, Apr. 2022.

[71] A. J. Davison and J. Ortiz, "FutureMapping 2: Gaussian belief propagation for spatial AI," 2019.