LTA-OM: Long-Term Association LiDAR-IMU Odometry and Mapping

Zuhao Zou
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
zuhao.zou@connect.hku.hk

Chongjian Yuan
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
ycj1@connect.hku.hk

Wei Xu
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
xuweii@hku.hk

Haotian Li
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
haotianl@connect.hku.hk

Liang Li
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
llihku@connect.hku.hk

Fu Zhang
Department of Mechanical Engineering
University of Hong Kong
Pokfulam, Hong Kong
fuzhang@hku.hk

Abstract

Simultaneous localization and mapping (SLAM) technology is ubiquitously employed in ground robots, unmanned aerial vehicles, and autonomous cars. This paper presents LTA-OM: an efficient, robust, and accurate LiDAR SLAM system. Employing FAST-LIO2 and Stable Triangle Descriptor as LiDAR-IMU odometry and the loop detection method, respectively, LTA-OM is implemented to be functionally complete, including loop detection and correction, false positive loop closure rejection, long-term association mapping, and multi-session localization and mapping. One novelty of this paper is the real-time long-term association (LTA) mapping, which exploits the direct scan-to-map registration of FAST-LIO2 and employs the corrected history map to constrain the mapping process globally. LTA leads to more globally consistent map construction and drift-less odometry at revisit places. We exhaustively benchmark LTA-OM and other state-of-the-art LiDAR systems with 18 data sequences. The results show that LTA-OM steadily outperforms other systems regarding trajectory accuracy, map consistency, and time consumption. The robustness of LTA-OM is validated in a challenging scene - a multi-level building having similar structures at different levels. Besides, a multi-session mode is designed to allow the user to store current session's results, including the corrected map points, optimized odometry, and descriptor database for future sessions. The benefits of this mode are additional accuracy improvement and consistent map stitching, which is helpful for life-long mapping. Furthermore, LTA-OM has valuable features for robot control and path planning, including high-frequency and real-time odometry, drift-less odometry at revisit places, and fast loop closing convergence. Moreover, LTA-OM is versatile as it is applicable to both multi-line spinning and solid-state LiDARs, mobile robots and handheld platforms.

1 Introduction

Simultaneous localization and mapping (SLAM) is a critical technology in ground robots, unmanned aerial vehicles, and autonomous cars. A SLAM system can estimate the trajectory of a moving platform and the map of the surrounding environment (e.g. Fig. 1) in real-time by processing the data from LiDARs (Zhang and Singh, 2014; Xu et al., 2022) or cameras (Mur-Artal et al., 2015; Qin et al., 2018). Trajectory and map are fundamental for robot navigation, control, and path planning. However, a visual SLAM using cameras is sensitive to light condition variation and often provides a sparse map. In contrast, a LiDAR SLAM does not suffer from these problems and holds high potential for safety-critical applications such as autonomous cars.

The past decade has witnessed a significant development in LiDAR SLAM. A LiDAR SLAM system consists of three core modules: LiDAR odometry (LO) (Zhang and Singh, 2014; Shan and Englot, 2018; Ye et al., 2019; Shan et al., 2020; Lin and Zhang, 2020; Li et al., 2021; Pan et al., 2021; Xu et al., 2022), LiDAR loop detection (He et al., 2016; Kim et al., 2021; Dubé et al., 2018; Liu et al., 2019; Chen et al., 2022; Yuan et al., 2022) and pose graph optimization (PGO) (Grisetti et al., 2011; Kaess et al., 2012). Some of the LO methods fuse LiDAR data with inertial measurement unit (IMU) data (Shan et al., 2020; Ye et al., 2019; Li et al., 2021; Xu et al., 2022) and thus are also called LiDAR-IMU odometry (LIO). LIO only fulfills the local map consistency, and the global map consistency is achieved via integrating LIO with loop detection and pose graph optimization. More specifically, LiDAR loop detection can detect revisited places where loop constraints can be exploited to correct the long-term odometry drift in pose graph optimization. A global map is also generated using the pose graph optimization result.

We develop a complete LiDAR SLAM system with all these three modules. For the front-end odometry, we use FAST-LIO2 (Xu et al., 2022) because of its superior accuracy, efficiency, and robustness for different LiDAR types and environments. For the loop detection, we employ our recent work called Stable Triangular Descriptor Loop Closure Detection (STD-LCD) (Yuan et al., 2022), which achieves state-of-the-art precision-recall and provides a loop closure candidate index with reliable 6 degrees of freedom (DoF) transformation. Besides, STD-LCD is well suited to different LiDAR types, from multi-line spinning LiDARs to emerging solid-state LiDARs, and can detect loop closures for revisited places with a small overlap and drastic changes in viewpoint.

When a loop is detected, a natural usage of the loop is adding loop constraints into the pose graph optimization to do loop optimization. Exploiting the relative poses estimated from the LIO and loop detection modules, pose graph optimization does not ensure the live map and the history map at revisited places can always align after the loop closure scan, due to the lack of direct constraints on map consistency. To address this issue, we propose to reload map points from history maps to globally constrain the online localization and the live map construction of LIO, serving as a direct constraint on the map consistency at revisit places. In detail, we first correct points in history maps using the odometry updated by loop optimization and then insert these map points into the live map of LIO module on the fly. Finally, the upcoming LiDAR scans are registered to the live map, consisting of points from recent and corrected history maps. This approach is called long-term association (LTA) mapping, significantly reducing the odometry drift at revisited places and improving map consistency therein. When the robot moves out of the previously mapped areas, the system adds new points to the live map and grows the map seamlessly.

LiDAR SLAM systems employ robust estimators such as Huber loss (Huber, 1973) or Cauchy loss to make the system robust against false-positive loop closures. However, such robust estimators would considerably soften the correction brought by loop constraints, especially when the LIO drift is significant. In detail, the first few loop constraints have large initial residuals, and the optimization with robust estimators would falsely treat them as outliers, assigning small optimization weights for them. Consequently, the significant drift would sustain for a long period until a sufficient number of correct loop constraints are introduced to the pose graph. We call this phenomenon the slow loop closing convergence problem. Besides, robust estimators cannot close the loop correctly when dealing with high false positive ratio cases. We propose actively rejecting false-

positive loop closures by an optimize-and-recover mechanism to address these challenges. More specifically, when the system detects a loop scan, the system optimizes the graph using the corresponding loop closures constraints. Then, the constraint residuals inside the graph loop cycle are examined to assess if the resultant graph is consistent. If not, the system rejects the loop constraints and recovers the pose graph back to its previous status. This optimize-and-recover mechanism decouples the loop optimization from the LIO module so that false-positive loop closures will not cause the system to breakdown (e.g., as in LIO-SAM (Shan et al., 2020) and LILIOM (Li et al., 2021)). The loop optimization result is conveyed to the LIO module only if the graph consistency check is passed.

In summary, the main contributions of this report are:

1. **LTA-OM: an efficient, robust, and accurate SLAM system**. We integrate FAST-LIO2, STD-LCD, and loop optimization into the system. The efficiency and accuracy of LTA-OM are proven to outperform state-of-the-art LiDAR systems on extensive benchmark experiments, including urban, campus, and non-man-made environments with the same or reverse direction, small-overlap revisit, and various time duration ($8 \sim 112$ mins) or travel distances ($1.1 \sim 7.5$ km). We also test our system against a structurally-similar scene - a multi-level building having similar corridors at different levels, and our system can still function properly and build a nice map.

2. **Real-time long-term association**. Our work leverages history map points at revisited places to globally constrain LIO's online localization and mapping. When loop optimization is done, we correct the pose state of FAST-LIO2 with the loop optimization result and dynamically reload map points from history maps to the live map for online LIO. With this, LIO's map construction becomes globally consistent, and LIO's localization becomes driftless due to the usage of the corrected history map. Besides, the mechanism of pose correction and live map update is well-designed to achieve real-time performance and avoid long LIO delays.

3. **Multi-session mode**. The long-term association in our SLAM system extends naturally to a multi-session mode where a user can store the map of LTA-OM in the previous session and load it for relocalization in the current session. More specifically, in this mode, the system first relocates in the pre-stored map using the loop detection module of LTA-OM. Then, the LIO module of LTA-OM dynamically loads the pre-stored map to the live map built in the current session to constrain the online localization and mapping. As a result, the pre-stored and live maps are naturally stitched, and the LIO pose becomes driftless on the pre-stored map. When the robot enters unexplored areas in the pre-stored map, LTA-OM appends new points to the stitched map, which automatically grows the map without further merging.

The rest of this report is organized as follows. A literature review of relevant works will be presented in Sec. 2. The methodology of our system will be illustrated in Sec. 3 and 4. Experiments will be conducted in Sec. 5. Finally, a conclusion is drawn in Sec. 6.

2 Related Works

2.1 LiDAR Inertial Odometry

A pioneering work in LiDAR odometry and mapping is LOAM (Zhang and Singh, 2014). It proposes to utilize the smoothness of the local surface at each point to extract geometrical features such as edges and planes for consecutive scan matching. However, the smoothness computation for a dense point cloud is time-consuming and impacts the performance when computation resource is limited. LeGO-LOAM (Shan and Englot, 2018) is a lightweight alternative to LOAM. It proposes to segment features in range images generated from the 360 degrees LiDAR and utilizes a two-step optimization to make the system operate on an embedded system in real-time. It allows users to optionally import IMUs' onboard estimated poses to provide inter-scan transformation prior for improved robustness. Subsequently, LIOM, (Ye et al., 2019) LIO-SAM (Shan et al., 2020), and LILIOM (Li et al., 2021) propose to maintain a sliding window and register scans to the local map

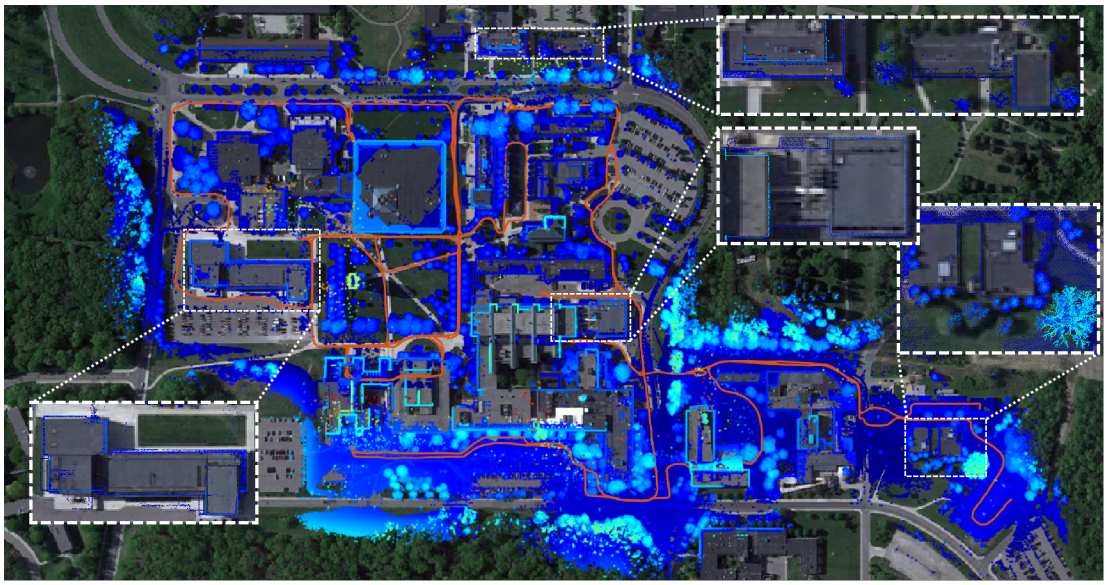

Figure 1: The constructed map (blue points) and estimated trajectory (red curve) by LTA-OM (proposed) on NCLT 2012-02-02 sequence. The constructed map is compared with the corresponding satellite image. The dash boxes are zoom-in regions showing that the map matches the outline of buildings with nice global consistency.

with an optimization approach. LIOM and LIO-SAM use IMU pre-integration method (Forster et al., 2015) to deskew the motion distortion on LiDAR scans. LILIOM uses IMU pre-integration method to introduce inertial constraints between keyframes. FAST-LIO2 (Xu et al., 2022) is a more lightweight LIO system. Compared with the optimization-based methods LIOM, LIO-SAM, and LILIOM, FAST-LIO2 is based on an iterative extended Kalman filter, which tightly couples the high-rate measurements to deskew a LiDAR scan and makes the system robust against rapid motions. FAST-LIO2 further develops an incremental kd-Tree (ikd-Tree) to enable fast and efficient scan-to-map registration. Due to the high computation efficiency and robustness of FAST-LIO2, we choose it for the LIO part of our SLAM system.

2.2 LiDAR Loop Closure Detection

Kim and Kim propose Scan Context (Kim and Kim, 2018), which projects the cloud into the ground plane and encodes the height information as an image. The image is transformed into a ring key vector, enabling fast loop closure candidates retrieval with KD-tree searching. They further upgrade Scan Context to Scan Context++ (Kim et al., 2021) with lower descriptor extraction and retrieval time. Yuan et al. (Yuan et al., 2022) propose STD-LCD, which extracts key points in the cloud using local density and describes the cloud as a set of triangular descriptors using triplets of key points. In the loop closure retrieval phase, triangular descriptors quickly suggest candidates using hash values retrieval and accelerate the subsequent geometrical verification. STD-LCD can detect loop closures quickly and provide a reliable 6D transformation between the history and current maps. Furthermore, STD-LCD is versatile because it is applicable to both multi-line spinning and solid-state LiDARs, and it can handle various environments such as indoor, urban, and wild fields. Therefore, we use STD-LCD as our loop detection method.

2.3 False Positive Rejection

Loop closure detection algorithms cannot always promise 100% precision, necessitating a false positive rejection (FPR) module to reject false-positive loop closures. Robust estimators such as Huber loss (Huber, 1973) and Cauchy loss are traditional methods to keep optimization away from the impact of false-positive

loop closures. However, robust estimators have a slow loop closing convergence problem when LIO drifts severely. Sunderhauf and Protzel (Sünderhauf and Protzel, 2012) propose to add switchable factors into the graph so that inconsistent factors can be disabled automatically by corresponding switchable factors. The extra computation burden in optimization brought by switchable factors is heavy. Latif et al. (Latif et al., 2013) propose the RRR algorithm, which first checks consistency for intra-clusters and inter-clusters constraints. After that, if some inconsistent constraints are rejected, the consistency check is conducted again for the reminder constraints. These three consistency check requires three PGO executions that bring significant latency to SLAM systems. Their method is designed to handle theoretically challenging cases such as a batch of false positive loop closures concurrently appearing at different times or places, and we focus on the case of online SLAM where loop closures come in sequences, not in batch. We adopt the idea of graph consistency check to do active rejection. If a false positive loop constraint is rejected, the pose graph is recovered back to its previous status. Therefore, we do not need to re-estimate graph parameters after the rejection as performed in RRR (see Sec. 4.3.2).

2.4 LiDAR Inertial SLAM

LeGO-LOAM, LIO-SAM, and LILIOM are complete LiDAR SLAM systems, and they integrate LIO with simple region search loop detection and PGO. Kim et al., (Kim et al., 2022) well-integrate FAST-LIO2, LIOSAM, and LeGO-LOAM with a more sophisticated loop detection algorithm - Scan Context, and the resultant systems are called FAST-LIO-SC, LIOSAM-SC, and LeGO-LOAM-SC, respectively. Scan Context and region search loop detection cannot provide a 6D loop closure transformation. As a result, these systems require time-consuming ICP to compute transformation between the current scan (or submap) and the loop scan (or submap). Furthermore, ICP can verify loop closure pairs with a heuristic ICP residual threshold. STD-LCD, providing loop closure pairs with 6D transformation, allows us to avoid using ICP but still requires us to verify the correctness of the provided loop closure pairs and transformations for system robustness. We use the graph consistency check in our FPR module to verify the correctness. Overall, due to the elimination of ICP, our system is efficient and avoids introducing the latency caused by ICP.

Moreover, LIO-SAM and LILIOM couple the local mapping and localization with global loop optimization in PGO. In comparison, we decouple them to avoid the temporally degenerated PGO in FPR module causing frontend mapping and localization to fail.

2.5 Multi-session SLAM

In the multi-session mode, ORB-SLAM3 (Campos et al., 2021) online merges the local active map to the previous session's map in a separate thread. By exploiting the previous map, ORB-SLAM3 becomes robust against fast motion scenarios even without IMU, and the drifting level is significantly alleviated in some sequences. However, our method does not require extra map merging operation as the new scan points are automatically registered to pre-stored maps in the odometry module of our system. LT-mapper (Kim and Kim, 2022), a LiDAR multi-session framework, leverages Scan Context and ICP verification to find map correspondence between the previous and current sessions and stitches maps using map correspondence with PGO. As pointed out by the authors, this framework has inherent map alignment errors (though their LT-removert module is proposed to handle that), while the map stitching in our work is done seamlessly using LTA. LSMS (Large-Scale Multi-Session) (Labbe and Michaud, 2014), an RGBD SLAM framework, also stitches maps in a PGO fashion with inter-session map correspondence. Map correspondence in LSMS is found via visual loop detection and RANSAC transformation verification. LSMS and LT-mapper focus more on large-scale map management compared with our work. LSMS introduces a memory forgetting mechanism to achieve real-time loop detection and PGO, while LT-mapper extracts map change areas to enable storing the new map as a persistent map and the map change areas only. In comparison, our work focuses on the driftless odometry on a pre-stored map, robust relocalization, and efficient and nice map stitching performance.

3 System Description

The pipeline of our system is summarized in Fig. 2. Our system consists of four core modules: LiDAR-IMU odometry (LIO), loop detection, loop optimization, and loop correction.

The pipeline workflow is briefly illustrated here. The LIO module, a variant of FAST-LIO2 incorporating long-term association, provides the registered scan and estimated odometry to the loop detection module. The loop detection module, STD-LCD, extracts the key points and detects the loop closure for the loop optimization backend. Then, the loop optimization employs the provided odometry, key points, and loop closure information to construct and maintain a pose graph and then optimize the pose graph upon a loop closure is detected. Once a successful loop optimization is done, loop correction employs the optimized odometry to correct the pose state and replace the map (i.e., ikd-Tree) of the LIO module.

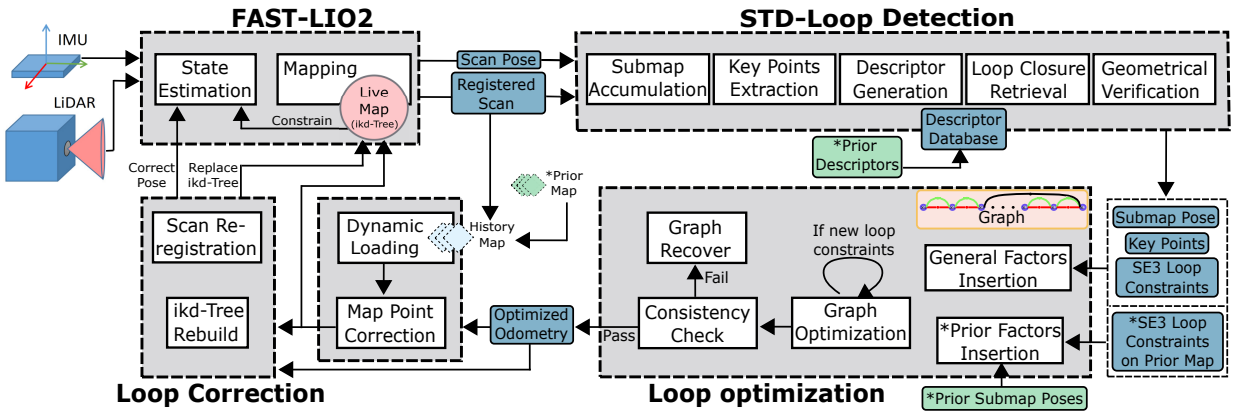

Figure 2: Main system components of LTA-OM. The components with the notation " * " are required only when a pre-stored map is available (so-called multi-session mode).

4 System Modules

4.1 LiDAR-IMU Odometery

We employ FAST-LIO2 as our LIO module. FAST-LIO2 is based on a tightly coupled iterated Kalman filter. It handles the motion distortion problem present in LiDAR measurement and provides online extrinsic estimation for LiDAR and IMU frames. Furthermore, it employs incremental kd-tree (ikd-Tree) to enable fast direct (i.e., without feature extraction) scan-to-map registration and state update.

*1)**Localization***: FAST-LIO2 maintains a state vector containing states of global IMU pose, global IMU velocity, IMU bias, gravity vector, and LiDAR-to-IMU transformation. It uses IMU measurement and IMU noise models to forward-propagate states. Subsequently, using a kinematic model with IMU measurement, it backward-propagates LiDAR points to accurate positions. Backward propagation can compensate for the motion distortion of points. Then, compensated points are used to compute state update residuals and the Kalman gain. Finally, FAST-LIO2 iteratively updates states until convergence.

*2)**Mapping with long-term Association***: FAST-LIO2 introduces ikd-Tree to enable incremental point insertion and deletion. When a new point is inserted into a conventional kd-Tree, the original balance of the tree is broken, and a time-consuming tree rebuild is required. The ikd-Tree introduces box-wise deletion with lazy labels to improve point deletion efficiency and a parallel tree rebuild strategy to protect the mapping module from the rebuild delay. Besides, the residual computation mentioned in the localization module depends on ikd-Tree. An unregistered point is used as the reference point to search five neighboring points

on the ikd-Tree. These neighboring points are fitted with a plane, and the state update residual is computed as the point-plane distance. By iteratively minimizing the residual, the new scan pose can be obtained, which then registers the new scan to the ikd-Tree and also the global history map.

A key difference of our odometry from the standard FAST-LIO2 is the long-term association (LTA) employed in the mapping module. Besides merging new scan, we propose to load points from the global history map to ikd-Tree so that the global history map constrains the online state update globally. In detail, we dynamically load map points around the current robot position from the history map to the ikd-Tree. The ikd-Tree, referred to as the live map, hence contains points from both new scans and the global history map, which then registers the subsequent LiDAR scans with long-term constraints arising from the history map.

We load the points contained in history scans whose pose is in a certain range (i.e., the load range) of the current LIO pose. During loading points from the history scans, we correct the point position using the scan pose updated by the loop optimization module. Then, the history scan indices are marked as "loaded" if already considered in the previous loading process to avoid redundant operation. Therefore, the dynamic loading is performed periodically with a limited frequency: the next dynamic loading will occur when the LIO pose approaches the boundary of the last load range. Moreover, this dynamic loading strategy naturally leads to an interesting property: when the robot moves to previously explored areas, LTA will automatically take effect to merge live map with history map after loop correction, and when the robot moves to unexplored areas (no surrounding history scans), live map will grow automatically from history map without the effect of LTA.

4.2 Loop Closure Detection

We use STD-LCD as our loop closure detection module. STD-LCD can efficiently detect loop closure and provide reliable loop closure geometrical transformation. Furthermore, it can handle the challenge cases of small overlap revisits.

The detailed workflow of STD-LCD is stated in Fig. 2. In detail, submap accumulation component is to accumulate sufficient registered scans from LIO (by default, 20 moving LiDAR scans for multi-line spinning LiDAR). The pose of the accumulated submap is chosen as the scan pose (i.e., the LiDAR coordinate frame w.r.t. the global coordinate frame) of the final scan. Besides, submap indices are employed to label the scans stored in history map for future map point correction processes. Other components are explained as follows with detail.

*1)**Key points extraction and descriptors construction***: With the accumulated submap, STD-LCD voxelizes the submap and finds plane voxels by checking the on-plane point ratio. The plane voxels are merged into larger planes. The points of non-plane voxels are projected onto adjacent planes to construct images. With images, key points are extracted by finding the pixels with local maximum point-plane distance. Furthermore, a spatial non-maximum suppression is conducted to enhance extraction repeatability. Every triple of extracted key points can build up a triangle. Based on the triangle, STD-LCD designs the descriptor as a six-dimensional array, including three ascending triangular edge lengths and three projection normal vector dot product values. The projection normal vector is defined as the normal of each key point at point cloud surface.

*2)**Loop closure retrieval and verification***: Loop closure retrieval and verification consist of three parts: rough loop detection, fine loop detection, and geometrical verification. Rough loop detection can quickly propose N possible candidates by searching the matched historical descriptors in the hash table of the descriptor database. Fine loop detection verifies the proposed candidates by computing all transformation matrices between paired triangles and clustering the transformations to find the transformation with the most supporters. Fine candidates are those with maximum transformation supporters larger than 4 (by default). Finally, fine candidates' point-plane distances are computed, and the best candidate is selected as the one with the smallest distance. The best candidate is accepted as the loop closure result only when

its overall point-plane distance is smaller than a threshold. The point-plane distance computation is fast because a nice initial guess is provided by fine loop detection, and only key points (< 50) of source submap are used for the point-plane distance computation.

Remark 1: Accumulating tens of LiDAR scans in loop detection inevitably introduces a delay to the subsequent loop optimization and correction process. However, with good local accuracy and small delay, our online LIO odometry also satisfies the needs of path planning or robot control that often require local information. Furthermore, LIO odometry will also have good global accuracy at revisit places after one loop correction (see Sec. 4.4).

4.3 Loop Optimization

This subsection includes the three key parts of loop optimization: graph construction and optimization, false positive rejection, and key points factors marginalization.

4.3.1 Graph Construction and Optimization

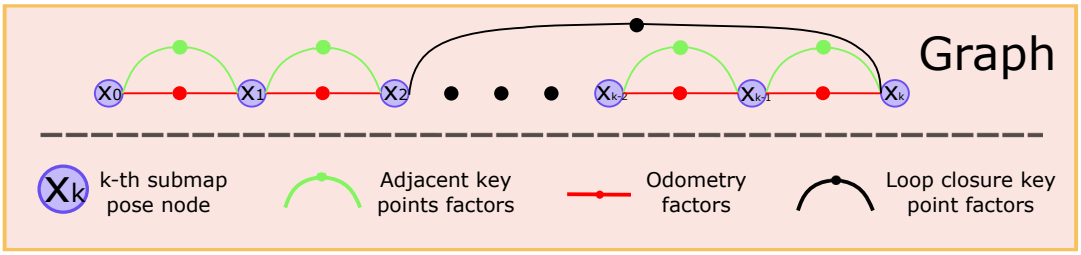

Figure 3: Graph overview.

A graph represents the relation between different constraints in the optimization problem. The graph we employed (see Fig. 3) is a variant version of a conventional pose graph. A conventional pose graph consists of pose nodes, odometry factors, and loop closure factors. The mathematical expression of pose graph optimization is shown as follows.

$$\{^{G}\mathrm{X}\} = argmin \sum_{(i,j)\in\mathcal{O}} ||\mathrm{Log}(\tilde{\mathrm{T}}_{ij}^{-1}\mathrm{T}_{ij})||^2 + \sum_{(i,j)\in\mathcal{L}} ||\mathrm{Log}(\tilde{\mathrm{T}}_{ij}^{-1}\mathrm{T}_{ij})||^2 \tag{1}$$

Where $\{^{G}\mathrm{X}\}$ denotes the collection of estimated submap pose w.r.t. global coordinate frame with index $k = [1, .., M]$, M the number of submap poses, $\tilde{\mathrm{T}}_{ij}$ and T_{ij} the measured and the estimated transformation between the i and j submap poses, $\mathcal{O}$ and $\mathcal{L}$ the sets of index pairs of the odometry constraints and loop constraints, respectively.

We employ the extracted key points from loop detection module (see Sec. 4.2) and add key point factors for adjacent pose nodes. Furthermore, we employ loop closure key point factors instead of loop closure factors. The mathematical expression of our pose graph optimization is shown as follows.

$$\{^{G}\mathrm{X}\} = argmin \sum_{(i,j)\in\mathcal{O}} ||\mathrm{Log}(\tilde{\mathrm{T}}_{ij}^{-1}\mathrm{T}_{ij})||^2 + \sum_{(i,j)\in\mathcal{K}_{\mathcal{N}}} ||^{L}\tilde{\mathrm{P}}_i - \mathrm{T}_{ij}^{L}\tilde{\mathrm{P}}_j||^2 + \sum_{(i,j)\in\mathcal{K}_{\mathcal{L}}} ||^{L}\tilde{\mathrm{P}}_i - \mathrm{T}_{ij}^{L}\tilde{\mathrm{P}}_j||^2 \tag{2}$$

Where $^{L}\tilde{\mathrm{P}}_i$ and $^{L}\tilde{\mathrm{P}}_j$ denotes the associated key point pairs of the i and j submaps respectively, $^{L}\tilde{\mathrm{P}}$ the key points w.r.t. the local submap coordinate frame, $\mathcal{K}_{\mathcal{N}}$ and $\mathcal{K}_{\mathcal{L}}$ the sets of index pairs of the neighboring and loop closure key point constraints, respectively. Neighboring key point pairs are associated by using kd-tree searching with a radius threshold in the global coordinate frame, while STD-LCD associates loop closure key

point pairs. Once loop closure key point constraints are inserted into the graph, pose graph optimization will be executed to do loop optimization.

4.3.2 False Positive Rejection

False-positive loop closures heavily impact pose graph optimization, and we introduce an FPR module to reject them. Our FPR method is based on the fact that the optimized key point factors shall output reasonable residual magnitudes. Equivalently speaking, when false-positive loop constraints are introduced to the loop optimization, an apparent separation of the same surface or edge in the resultant map will appear, causing large key point pair distances (i.e., irrational factor residuals). The detailed pseudo-code of the proposed FPR method is shown in Alg. 1. We first calculate and threshold the overlap ratio of the loop closure submap pair. Though STD-LCD can handle small overlap cases, we only trust the loop closure with an overlap ratio larger than 0.5 in our system (Lines 1 $\sim$ 3). Before each loop closure optimization, we back up the graph and its variable values (Line 8). Then, we try pose graph optimization with loop closure key point factors and consistency check (i.e., check if the outcome residuals of key point factors are smaller than a threshold. Lines 14 $\sim$ 21). When the graph becomes inconsistent, we reject this loop closure and recover the graph to the backup (Lines 9 $\sim$ 13). The threshold to control the key point factor residual can be regarded as a point-to-point distance threshold. It is a one-dimensional variable and, thus, easy to tune. However, when dealing with the first loop closure or a loop closure on the condition that the distance between current pose and loop closure pose is large (e.g., greater than 10% of the maximum range of x, y, and z on the map), we need to be more careful as this loop closure will heavily influence consequent loop closure acceptance. To better handle these cases, we maintain a container $\mathcal{L}_p$ that temporarily stores key point factors that are not added to the graph (Line 4). We propose to check two consecutive loop closures at one go in these cases because they are unlikely false-positive once both their inter-consistency and in-graph consistency are fine (Lines 5 $\sim$ 7).

Our FPR module does not execute PGO but simply recovers the graph to the previous status, avoiding introducing an extra delay to the systems as performed in RRR (Latif et al., 2013). Compared with robust estimators, it is more robust against a structurally similar scene (see Sec. 5.4) and has fast loop closing convergence.

4.3.3 Key Point Factors Marginalization

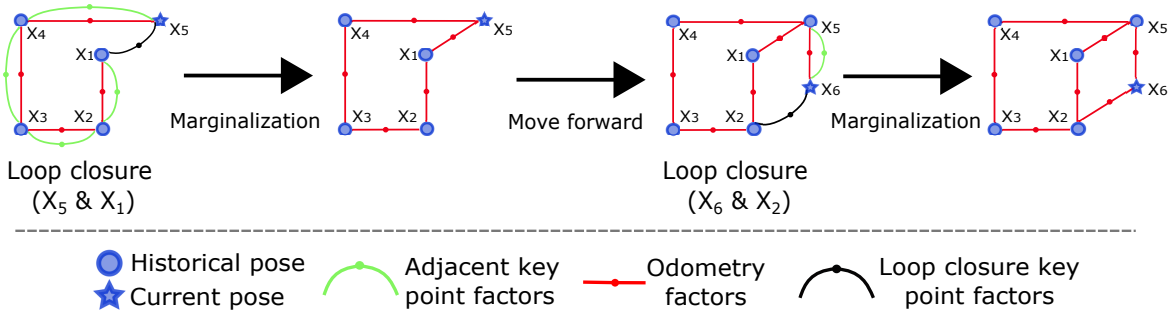

Figure 4: Illustration of key point factors marginalization.

Introducing key point factors to the loop optimization will gradually slow down the optimization because the optimization time is proportional to factor number, necessitating the marginalization of key point factors to prevent factor number from increasing unboundedly. In addition, an incremental optimization framework called ISAM2 (Kaess et al., 2012) is also employed for optimization efficiency.

The idea of key point factors marginalization is illustrated with a simple example shown in Fig. 4. Using this figure as an example, when a robot moves from the first pose to the fifth pose, a loop closure is detected

Algorithm 1: False Positive Rejection

Input : Factor graph $\mathbf{G}$;
Loop closure key point factors $\boldsymbol{l_p}$;
Overlap ratio between loop submaps $\boldsymbol{ratio_o}$;
The distance between current pose and loop closure pose $\boldsymbol{d}$;
Map range $(\boldsymbol{x_{range}}, \boldsymbol{y_{range}}, \boldsymbol{z_{range}})$;
Residual threshold $\boldsymbol{thr_r}$;
Overlap threshold $\boldsymbol{thr_o}$;
Output : Corrected factor graph $\mathbf{G}$.

if $\boldsymbol{ratio_o} < \boldsymbol{thr_o}$ **then**
  return;
end
$\mathcal{L}_p \leftarrow \mathcal{L}_p \cup \{\boldsymbol{l_p}\}$; /* Store l_p to a constraint set $\mathcal{L}_p$ */
if $\boldsymbol{l_p}$ is the first loop closure key points factor OR $\boldsymbol{d} > 0.1 * max(\boldsymbol{x_{range}}, \boldsymbol{y_{range}}, \boldsymbol{z_{range}})$ **then**
  return; /* Wait for more loop closure key points factor */
end
$G' \leftarrow G$; /* Back up graph */
Add $\mathcal{L}_p$ to G and optimize G; /* Try optimize graph with $\mathcal{L}_p$ */
if !ConsistencyCheck(G, thr_r) **then**
  $G \leftarrow G'$; /* If map not consistent, recover graph to reject the loop closure */
end
$\mathbb{L}_l \leftarrow \emptyset$;

Function ConsistencyCheck(G, thr_r)
  for $f \leftarrow$ key point factors of G **do**
    if $f.residual > thr_r$ **then**
      return false;
    end
  end
  return true;
end

and accepted by the FPR module. After the optimization, the loop-closed graph will be used to recalculate the relative transformation between consecutive poses. The recalculated relative poses are inserted into the graph as new odometry factors to replace the previous factors, including both key point factors and old odometry factors. Then, marginalization is done for the loop cycle of $\{1, 2, 3, 4, 5\}$. Similarly, when the robot moves from the fifth to the sixth pose, another loop closure is accepted, followed by marginalization for the loop cycle of $\{1, 2, 5, 6\}$.

4.4 Loop Correction

In this subsection, we first use a sequence flow diagram (see Fig. 5) to show how loop correction module works, and then we detail the four key steps, including the dynamic loading, map points correction, ikd-Tree rebuild, and scan re-registration in the pipeline (Fig. 2)

It is worth noting that we employ a separate thread called the "ikd-Tree rebuild thread" to do the extra tasks. This asynchronous operation prevents the original FAST-LIO2 thread from significant hangup. More implementation details are discussed in the following subsections.

4.4.1 Points Correction and Ikd-Tree Rebuild

Naive ikd-Tree rebuild for all history points is time-consuming, and dynamic loading of history points can alleviate this problem. After receiving the optimized odometry from the loop optimization node (see Fig. 5), the ikd-Tree rebuild thread loads the history points surrounding the current position. This dynamic loading is based on a kd-Tree of scan positions. All scan positions in optimized odometry are used to create

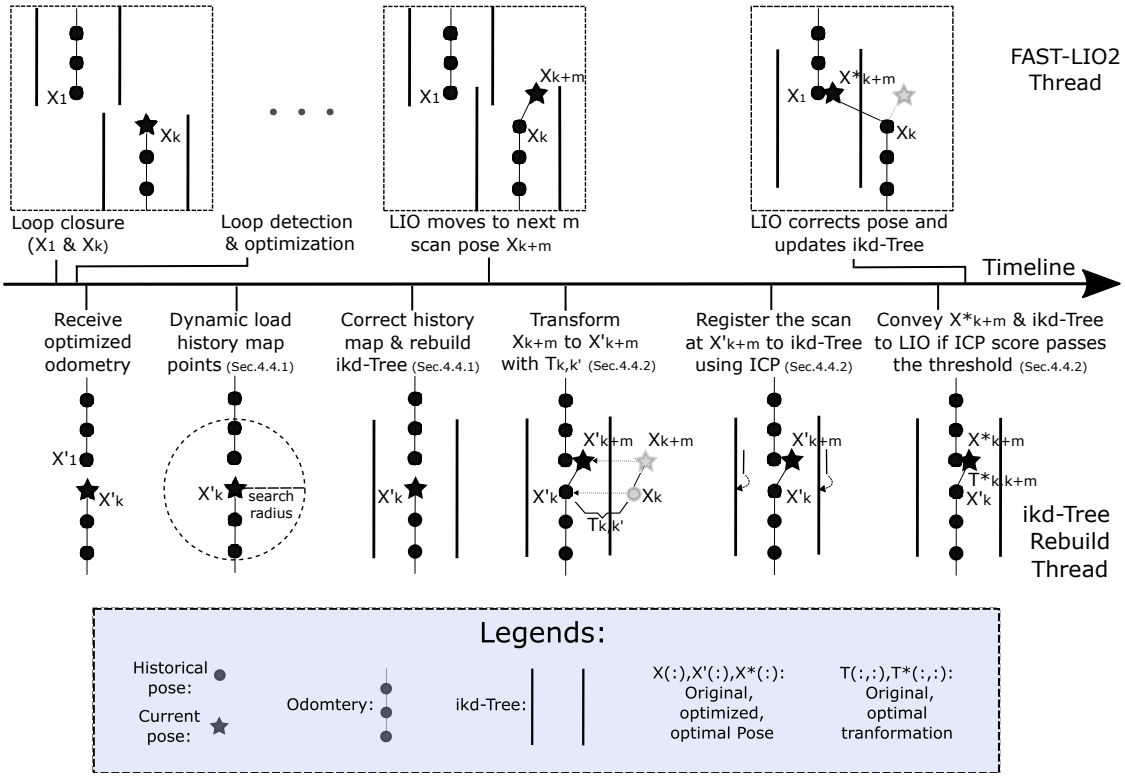

Figure 5: Workflow of loop correction.

a position kd-Tree for searching the neighboring scans in dynamic loading. The loaded points are corrected with the optimized odometry and used to rebuild a new ikd-Tree.

Remark 2: The stored submap points for points correction are the on-tree downsampling result of FAST-LIO2, and thus the point number is inherently far smaller than the raw point number. Besides, when the revisit occurs repeatedly, the downsampling ratio will increase; thus, the number of stored submap points will not rise unboundedly. The detailed time consumption is discussed in Sec. 5.2.

4.4.2 Pose Correction and Ikd-Tree Replacement

During loop detection, loop optimization, points correction, and ikd-Tree rebuild, LIO pose has moved forward from X_k to X_{k+m}. The loop optimized k-th pose $\mathrm{X}_k^{'}$ is outdated w.r.t. the up-to-date LIO pose X_{k+m} (see Fig. 5). We cannot directly reset LIO pose to be $\mathrm{X}_k^{'}$. Instead, we need to transform X_{k+m} to $\mathrm{X}_{k+m}^{'}$ with the transformation from X_k to $\mathrm{X}_k^{'}$ (denoted as $\mathrm{T}_{k',k}$). However, $\mathrm{T}_{k',k}$ is not always precise enough to fit $\mathrm{X}_{k+m}^{'}$ with the rebuilt ikd-Tree because of the imperfect optimization result. We need to do the ICP to re-register the projected scan using $\mathrm{X}_{k+m}^{'}$ to the rebuilt ikd-Tree ("Scan Re-registration" in Fig. 2), during which FAST-LIO2 must be hung up for the ICP taking average only about 8 ms (see Tab. 7). After that, the FAST-LIO2 thread corrects its pose as X_{k+m}^{*} and replaces its ikd-Tree with the rebuilt one in a short time (<1 ms) to bring LTA into service. In summary, the optimal LIO pose is calculated as follows:

$$\mathrm{X}_{k+m}^{*} = \mathrm{T}_{icp}\mathrm{T}_{k',k}\mathrm{X}_{k+m} = \mathrm{T}_{icp}\mathrm{X}_k^{'}\mathrm{X}_k^{-1}\mathrm{X}_{k+m}. \tag{3}$$

Where T_{icp} denotes the ICP result. Let pose correction transformation T_{corr} be:

$$\mathrm{T}_{corr} = \mathrm{T}_{icp}\mathrm{X}_k^{'}\mathrm{X}_k^{-1} \tag{4}$$

Then,

$$\mathrm{X}^{*}_{k+m} = \mathrm{T}_{corr}\mathrm{X}_{k+m}. \tag{5}$$

4.4.3 Maintenance Policy

As the accuracy of optimized odometry is gradually enhanced with more and more loop constraints, we execute loop correction with a frequency to allow the long-term association to utilize the newest loop optimization result. The frequency is controlled with a spatial interval (50 m by default). All the scans marked as "loaded" in the last loop correction execution will be reset as "not loaded" for the current loop correction execution.

4.4.4 Odometry Factor Recalculation

Pose correction introduces an abrupt pose state change to the LIO module, which impacts the accuracy of odometry factor calculation. We need to recalculate the impacted odometry factor. Suppose the pose correction occurs between the consecutive submap poses X_i and X_j (where $i > j$), X_j must be recalculated as:

$$\mathrm{X}^{*}_{j} = \mathrm{T}_{corr}\mathrm{X}_{j} \tag{6}$$

Then, the recalculated odometry factor measurement is:

$$\tilde{\mathrm{T}}^{*-1}_{ij} = \mathrm{X}_i\mathrm{X}^{*-1}_{j} = \mathrm{X}_i\mathrm{X}^{-1}_{j}\mathrm{T}^{-1}_{corr} \tag{7}$$

4.5 Multi-session Mode

When LTA-OM processes one data sequence, we can store the resultant odometry, map, and descriptor database as prior information for future localization, mapping, and loop detection. Loading the prior information from one sequence to process another is called multi-session mode. This mode is proven to enhance current session accuracy (see Fig. 10). Besides, this mode enables building the live map consistent with the pre-stored map (called map stitching, see Tab. 8). Multi-session mode is useful for robot navigation since it avoids rebuilding the map when the robot performs a repetitive task in a specified area.

Implementation of multi-session mode is briefly included in Fig. 2. There are four major differences between single-session mode and multi-session mode, including 1. pre-stored map is loaded to the FAST-LIO node for the LTA module, but pre-stored map points are not needed to correct; 2. pre-stored descriptor database is loaded to the loop detection node for the loop closure retrieval on the prior map, and we set its priority higher than the retrieval on the history map; 3. pre-stored submap poses are loaded to loop optimization node for prior factor calculation; 4. relocalization is implemented to correct the LIO pose onto the pre-stored map.

Remark 3: Considering the loop detection result is not always correct, we must carefully reject false-positive loop closure to avoid disastrous system breakdown. Thanks to our FPR module, we can actively reject them, which is not achievable using conventional robust estimators. When two consecutive loop closures against the prior descriptor database are detected, we can compute the corresponding prior factors for the online submap poses and try optimizing the graph with the prior factors. If they are consistent with each other, we accept them and accomplish subsequent relocalization.

5 Experiments

In this section, experiments in terms of accuracy, map consistency, and computational efficiency are conducted. We compare our system with other four state-of-the-art LiDAR SLAM systems including FAST-LIO-SC[1], LIOSAM-SC[2], LeGO-LOAM-SC[3] and LILIOM[4]. The first three LiDAR systems (Kim et al., 2022) are integrating Scan Context with FAST-LIO2, LIOSAM, and LeGOLOAM by the authors of Scan Context. Note that LeGOLOAM-SC requires IMU measurement while LeGOLOAM does not. Our experiment platform is a 2.90 GHz 16-cores Intel i7-10700 CPU and 15.5 GB RAM.

The datasets used for benchmark comparison are Mulran dataset (Kim et al., 2020) and NCLT dataset (Carlevaris-Bianco et al., 2016). Both datasets comprise 10 Hz LiDAR data, more than 100 Hz IMU data, and good quality 6 DoF ground truth trajectory with timestamps. Mulran dataset has three challenging scenarios for benchmarking LiDAR SLAMs, including DCC, KAIST, and riverside sequences. DCC and KAIST sequences have buildings, a mountain, trees, moving cars, etc. They include both same-direction and reverse revisits, and reverse revisits have low scans overlap. It is difficult for a loop detection method to detect all the loop closures in these sequences. Riverside sequences have trees, a river, bridges, and moving cars. Trees and bridges have repetitive structures, which are hard for a loop detection method to detect loop closures correctly and for LiDAR SLAM to alleviate the impact of false-positive loop closures. The NCLT dataset includes sequences collected in different seasons at the University of Michigan North Campus, a scene with bushes, trees, and campus buildings. Compared with Mulran sequences, NCLT sequences have more frequent revisits in the same and opposite directions, more significant variance in the z-direction, and larger average time duration. More details of sequences are listed in Tab. 10.

5.1 Benchmark: Trajectory Accuracy and Map Consistency

To evaluate the accuracy of LiDAR SLAMs, we use the EVO trajectory evaluation tool[5] to compute the absolute trajectory error for their result trajectories. The evaluation tool aligns the trajectory estimation to the ground truth trajectory using the Umeyama algorithm (Umeyama, 1991) and computes the estimation's root mean square error (RMSE). Furthermore, we run our system for all sequences with the factor residual threshold of FPR module = 2 and loop closure overlap ratio threshold = 0.5.

The result is displayed in Tab. 1 and 2. To obtain each number of the tables, we repeat the trial five times and take the average of the results to make the comparison more statistically persuasive. Our system outperforms the competing SLAM systems on eight out of nine data sequences. The only exception is on DCC01 where our result is slightly worse than that of FAST-LIO-SC. The result proves the advantage of our system in trajectory accuracy. We also spot that our system outperforms FAST-LIO-SC even though both systems employ the same LIO module with identical configuration files. The accuracy advantage of our system against the competing SLAM systems is due to our system being well-engineered and -designed to exert the capability of all modules and make them function harmoniously.

The comparisons of the constructed maps by LTA-OM, FAST-LIO-SC, LIOSAM-SC, and LeGOLOAM-SC on riverside02 and DCC02 sequences are shown in the right pictures of Fig. 6 and Fig. 7, respectively. It is seen that the map generated by LTA-OM is the most consistent one. An exception is in the case of riverside02, where the map constructed by LIO-SAM-SC shows similar consistency. The complete maps by LTA-OM for riverside02 and DCC02 sequences are shown in the left pictures of Fig. 6 and Fig. 7, respectively, and they show a global consistency. The map consistency advantage comes from the proposed LTA module, and Sec. 5.3 provides more evidence. Note that the LIO module of LILIOM is not working properly on Mulran dataset. Therefore, the constructed map by LILIOM is not comparable to those by the

[1]retrieved from https://github.com/gisbi-kim/FAST_LIO_SLAM on Sep. 2021
[2]retrieved from https://github.com/gisbi-kim/SC-LIO-SAM on Sep. 2021
[3]retrieved from https://github.com/irapkaist/SC-LeGO-LOAM on Sep. 2021
[4]retrieved from https://github.com/KIT-ISAS/LILIOM on Mar. 2022
[5]retrieved from https://github.com/MichaelGrupp/evo on Mar. 2022

other systems and thus not included in the comparison.

To display the global consistency of the map by LTA-OM, we compare the map with the Google Earth satellite image of the University of Michigan North Campus (see Fig. 1). The satellite image can be treated as the indirect ground truth of the map. Four zoom-in close shots spread all over the campus, showing that the map globally matches the actual outline of campus buildings. In other words, the map is globally consistent.

Table 1: The absolute trajectory error (RMSE, meters) comparison on Mulran dataset.

	R1	**R2**	**R3**	**K1**	**K2**	**K3**	**D1**	**D2**	**D3**	**Avg**
LTA-OM (proposed)	**6.98**	**6.32**	**10.38**	**3.25**	**3.10**	**3.06**	5.29	**2.94**	**2.17**	**4.83**
FAST-LIO-SC	7.96	7.21	10.67	3.44	3.90	3.12	**5.26**	5.26	4.21	5.67
LIOSAM-SC	9.27	8.05	11.62	3.77	3.81	3.59	5.77	3.50	3.02	5.82
LeGOLOAM-SC	24.6	19.21	31.85	319.58	76.49	4.74	224.86	4.13	3.58	78.78
LILIOM	>999	>999	457.69	>999	30.38	201.92	>999	34.04	x	>999

[1] × denotes that the system totally failed.

Table 2: The absolute trajectory error (RMSE, meters) comparison on NCLT dataset.

	N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**	**N9**	**Avg**
LTA-OM (proposed)	**1.46**	**1.61**	**1.38**	**1.52**	**1.76**	**1.90**	**1.79**	**0.90**	**1.75**	**1.56**
FAST-LIO-SC	2.44	2.17	2.71	1.74	2.36	2.88	6.96	1.00	3.21	2.83
LIOSAM-SC	42.03	224.25	59.67	1.76	267.13	2.85	425.00	3.36	651.49	186.39
LeGOLOAM-SC	288.59	275.61	220.54	86.87	83.42	96.83	273.89	83.51	43.81	161.45
LILIOM	>999	>999	>999	3.65	132.98	5.63	154.43	1.55	246.98	>999

5.2 Benchmark: Time Consumption

This section compares the overall time consumption per scan of LTA-OM, FAST-LIO-SC, LIOSAM-SC, LeGOLOAM-SC, and LILIOM. We record and sum up all running threads' (lidar odometry, mapping, loop optimization, etc.) execution times of these systems for each data sequence. After that, we divide the time sum by the total number of LiDAR measurements of the sequence to get the overall time consumption per scan. This time calculation method can ensure a fair comparison because systems employ different keyframe selection methods, submap accumulation policies, and thread execution frequencies. Also, this time calculation method can help us judge if a system is real-time, i.e., if its overall time consumption per scan is smaller than the one LiDAR scan duration (100 ms). The times are listed in Tab. 3 and 4.

Comparing the averages of overall times per scan in Tab. 3 shows that LTA-OM is roughly x2, x5, x3, and x2.3 more efficient than FAST-LIO-SC, LIOSAM-SC, LeGOLOAM-SC, and LILIOM on Mulran dataset, respectively. All systems can perform in real-time on Mulran dataset except LIOSAM-SC. According to Tab. 4, LTA-OM is roughly x1.5, x4, x2.7, and x4.5 more efficient than FAST-LIO-SC, LIOSAM-SC, LeGOLOAM-SC, and LILIOM on NCLT dataset, respectively. All systems can perform in real-time on NCLT dataset.

Furthermore, we experiment to compare the execution times of loop closure relevant modules, including loop detection, geometrical verification, and PGO. The execution times are listed in Tab. 5 and 6. Firstly, we can see that the average loop detection execution times of LTA-OM are 71.11 ms and 40.18 ms, significantly higher than those of other systems. The loop detection method used in FAST-LIO-SC, LIOSAM-SC, and LeGOLOAM-SC is the highly optimized version of SC where SC image augmentation is disabled, and only 10 % pixels of SC image are involved in image comparison. These tricks dramatically decrease SC execution time from about 100 milli-seconds as reported (Kim and Kim, 2018) to a few milli-seconds. At the same time,

these tricks degenerate the loop detection performance. This performance degeneration is not problematic due to the usage of ICP for further verification. The region search loop detection method used in LILIOM is efficient but also requires ICP verification. The ICP, however, takes hundreds of milli-seconds (see Tab. 5 and 6), impacting system efficiency and causing a significant delay. The geometrical verification times of LTA-OM are zeros because LTA-OM does not need ICP to verify the proposed loop candidates by loop detection module. Moreover, the average PGO execution times of LTA-OM for loop closures are the smallest among these systems except LILIOM on Mulran dataset. The PGO execution times of LILIOM are all zeros on Mulran dataset because its odometry drifting is severe, and region search does not propose any correct loop closure for optimization. In summary, taking all loop closure relevant modules into account, our system is more efficient than the other systems.

Besides, due to LTA-OM introducing loop correction module to enable LTA, we conduct a time consumption analysis for loop correction module. This module has three key steps: points correction, ikd-Tree rebuild, and scan re-registration. We test these steps on Mulran dataset, and their execution times per running are recorded as shown in Tab. 7. For each sequence, we compute the overall time of loop correction module. Besides, we compute the average step execution and overall times among nine sequences. In summary, points correction, ikd-Tree rebuild, and scan re-registration take an average of 8.68, 67.14, and 8.06 ms, and the average overall time is 83.87 ms. This overall time is smaller than one LiDAR scan duration of 100 ms; thus, this module runs in real time. In addition, this module executes at a very low frequency because the loop detection frequency is far smaller than LiDAR measurement frequency, and a space interval of 50 m is applied to control the loop correction frequency in practice (see Sec. 4.4.3).

5.3 Improvement by Long-term Association

We introduce the idea of long-term association to enable a globally consistent mapping, and here we experiment to compare the constructed maps with and without this module by turning on and off the long-term association module, respectively. We checked all the details in both cases' maps and found some map differences. The map differences are shown in Fig. 8, where we can see that the long-term association module improves the map consistency. Due to the limited space, the other examples are not displayed here.

5.4 Robustness against Structurally Similar Scene

We test our FPR module against a multi-level building scenario with similar structures at different levels. In the scene, three similar rectangular corridors of the same height and width exist. Half of the detected loop closures are false-positive as a point cloud-based loop detection module cannot distinguish the structurally similar corridors well. Our FPR works robustly and enables the system to construct a nice map shown in the left picture of Fig. 9. When Cauchy robust estimator replaces our FPR module, loop closing fails (see the right one) as the Cauchy robust estimator cannot handle high false positive rate cases. This dataset is collected with a Livox Avia Lidar and its embedded IMU. The data collection route is floor 1 $\rightarrow$ floor 2 $\rightarrow$ floor 3 $\rightarrow$ floor 2 $\rightarrow$ floor 1.

5.5 Multi-session Mode Result

In this experiment, we validate the accuracy improvement by multi-session mode and map stitching performance. For each Mulran scene, including Riverside, KAIST, and DCC, we use the sequence with the smallest RMSE to generate a prior map and the rest sequences to run multi-session mode, both by running our LTA-OM. According to the experiment result displayed in Tab. 8, multi-session has a lower absolute trajectory error than the single-session (copied from Tab. 1). This enhancement is mainly because the pre-stored map can eliminate the drift of the LIO and improve odometry factor accuracy. Besides, to further validate the fast response of the proposed multi-session mode, we also record the required scan number and time to accomplish relocalization (map switch) in Tab. 9. The result shows that our system can robustly relocalize

on the pre-stored map in seconds in our tests (the time for two submaps accumulation, loop detection, and loop correction modules). The quick map switching response enables other robot modules (e.g., autonomous navigation) to obtain global map information earlier.

The map stitching result is shown in Fig. 10, where we can see that the pre-stored map and the live map are consistent. Furthermore, when the robot enters unexplored areas (the middle part of live map), the newly appended points are stitched seamlessly with the pre-stored map without further map merging. This result is generated from the DCC01 sequence using prior information from the DCC02 sequence.

Table 3: Overall time consumption (milli-seconds per scan) comparison on Mulran dataset.

	LTA-OM (proposed)	**FAST-LIO-SC**	**LIOSAM-SC**	**LeGOLOAM-SC**	**LILIOM**
R1	**27.52**	30.60	139.49	67.64	51.43
R2	**26.74**	34.98	130.42	77.01	45.85
R3	**24.26**	45.30	124.37	79.78	42.36
K1	**33.43**	79.32	149.79	73.35	63.33
K2	**33.83**	89.08	159.58	101.10	100.28
K3	**32.37**	94.15	163.93	100.27	106.40
D1	**38.46**	79.65	157.66	108.69	95.10
D2	**32.82**	44.39	137.71	113.74	74.07
D3	**40.51**	94.94	169.88	98.62	x
Ave	**32.22**	65.82	148.09	91.13	72.35

[1] × denotes that the system totally failed, and thus the time consumption is unavailable.

Table 4: Overall time consumption (milli-seconds per scan) comparison on NCLT dataset.

	LTA-OM (proposed)	**FAST-LIO-SC**	**LIOSAM-SC**	**LeGOLOAM-SC**	**LILIOM**
N1	**23.44**	33.33	72.88	73.93	114.50
N2	**15.77**	24.36	92.87	62.09	31.80
N3	22.10	**19.22**	63.46	60.88	76.57
N4	**20.07**	36.66	107.64	45.11	119.08
N5	**23.39**	30.91	89.24	63.72	119.72
N6	**29.52**	34.75	95.07	51.20	113.81
N7	**14.71**	29.24	49.14	58.48	91.15
N8	**14.53**	26.19	62.73	32.31	66.74
N9	**22.06**	41.51	91.76	58.86	35.46
Ave	**20.62**	30.69	80.53	56.29	96.54

Table 5: The execution time (milli-seconds per running) comparison of loop closure relevant modules on the Mulran dataset. Terms "LD", "Ge", "Op", and "Ave" denote loop detection, geometrical verification, pose graph optimization, and average.

	LTA-OM (proposed)	**FAST-LIO-SC**	**LIOSAM-SC**	**LeGOLOAM-SC**	**LILIOM**
	LD/Ge/Op	**LD/ Ge/Op**	**LD/Ge/Op**	**LD/Ge/Op**	**LD/Ge/Op**
R1	60.07/0/10.72	0.68/479.80/30.95	1.12/584.51/13.06	1.22/158.83/14.36	0.18/0/0
R2	62.76/0/13.45	0.60/523.55/26.66	1.11/694.06/13.86	1.34/199.67/15.19	0.16/0/0
R3	57.05/0/20.00	0.65/557.49/43.51	1.05/498.85/11.01	1.40/177.32/17.45	0.15/0/0
K1	79.12/0/12.73	0.87/822.90/125.81	1.44/693.13/21.84	1.31/190.21/10.40	0.24/0/0
K2	72.85/0/14.53	0.83/778.78/123.61	1.57/921.62/28.67	1.61/234.31/15.07	0.19/890.72/0
K3	72.93/0/11.18	0.87/905.55/114.38	1.57/925.18/25.98	1.55/207.13/14.33	0.17/883.93/0
D1	83.85/0/10.03	0.83/727.90/52.71	1.18/743.52/6.86	1.47/276.28/12.40	0.13/777.79/0
D2	65.48/0/10.76	0.72/613.85/45.76	1.39/969.64/12.10	1.54/338.73/10.51	0.15/778.87/0
D3	85.85/0/8.98	0.94/1471.4/109.57	1.44/1280.0/13.57	1.36/272.61/7.78	x/x/x
Ave	71.11/**0**/12.49	0.78/764.58/74.77	1.32/812.28/16.33	1.42/228.34/13.05	**0.17**/832.83/**0**

[1] × denotes that the system totally failed, and thus the time consumption is unavailable.

Table 6: The execution time (milli-seconds per running) comparison of loop closure relevant modules on NCLT dataset. Terms "LD", "Ge", "Op", and "Ave" denote loop detection, geometrical verification, pose graph optimization, and average.

	LTA-OM (proposed)	**FAST-LIO-SC**	**LIOSAM-SC**	**LeGOLOAM-SC**	**LILIOM**
	LD/Ge/Op	**LD/ Ge/Op**	**LD/Ge/Op**	**LD/Ge/Op**	**LD/Ge/Op**
N1	46.45/0/72.84	0.77/1380.1/86.56	2.02/369.22/78.82	3.53/132.25/75.16	1.00/479.85/113.02
N2	29.13/0/23.53	0.66/615.99/71.43	1.60/426.95/20.45	2.70/129.38/53.09	0.22/211.75/29.85
N3	48.84/0/66.06	0.56/517.32/30.85	1.30/339.14/16.08	3.50/125.29/34.36	0.84/219.42/58.87
N4	38.51/0/12.97	0.71/1041.1/39.00	1.46/449.23/2.88	1.81/135.97/12.27	0.61/378.02/87.96
N5	40.74/0/50.13	0.65/866.52/100.73	1.64/538.12/272.93	3.33/115.65/47.77	1.83/366.72/73.85
N6	60.12/0/22.28	0.68/1609.8/53.39	1.26/491.67/3.60	2.74/112.84/14.02	0.61/353.11/42.63
N7	26.48/0/25.55	0.68/1177.3/76.65	1.30/381.13/21.66	2.45/114.16/81.29	0.84/456.35/67.69
N8	28.43/0/3.61	0.57/870.00/17.86	0.86/371.20/0.93	1.18/99.54/3.76	0.39/206.94/11.16
N9	42.92/0/18.34	0.64/1175.2/101.82	1.29/685.54/21.68	2.82/160.35/17.64	0.80/760.04/265.01
Ave	40.18/**0**/**32.81**	**0.66**/1028.1/64.25	1.41/450.24/48.78	2.67/125.05/37.71	0.79/381.35/83.34

Table 7: The execution time (milli-seconds per running) of the loop correction module on the Mulran dataset.

	Points Correction	**ikd-Tree Rebuild**	**Scan Re-registration**	**Overall**
R1	7.12	58.59	13.02	78.73
R2	6.82	70.47	9.52	86.81
R3	7.41	64.10	11.85	83.36
K1	9.04	80.01	2.74	91.79
K2	7.62	76.71	4.20	88.53
K3	10.84	60.17	9.39	80.40
D1	10.28	63.40	10.90	84.58
D2	7.43	49.93	3.94	61.30
D3	11.53	80.85	6.97	99.35
Ave	8.68	67.14	8.06	83.87

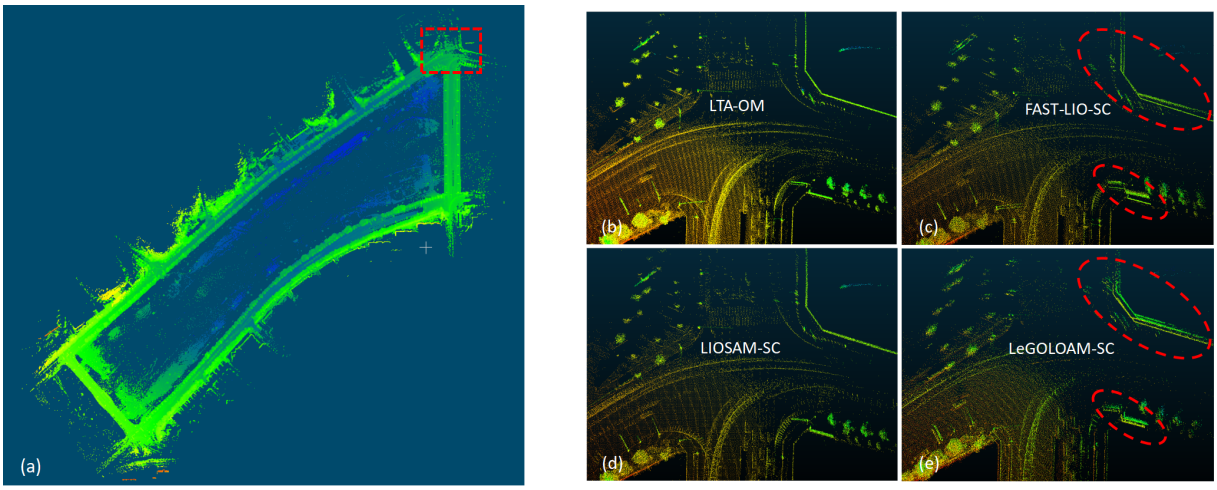

Figure 6: The full map constructed by LTA-OM with noted comparison region (a) and the maps comparison (b,c,d,e) on riverside02 sequence.

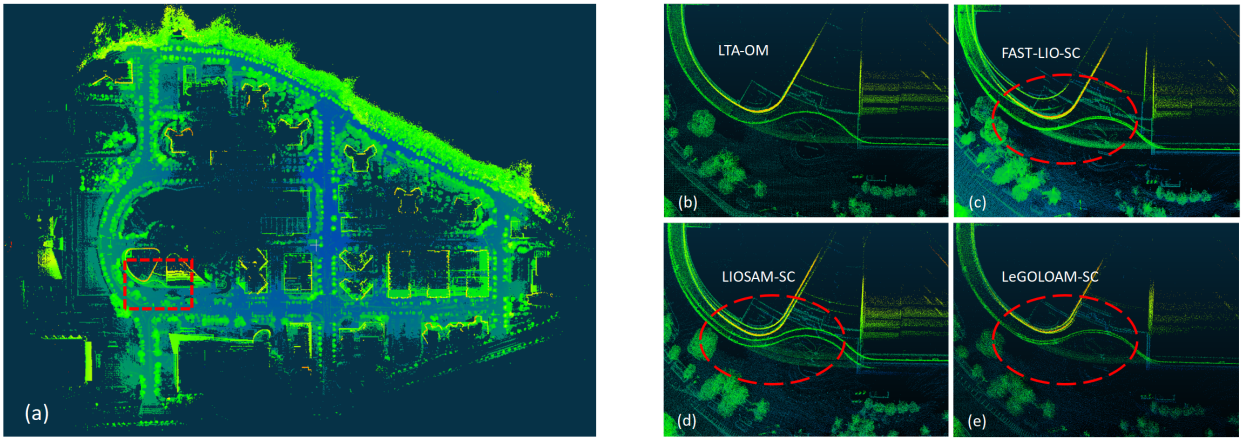

Figure 7: The full map constructed by LTA-OM with noted comparison region (a) and the maps comparison (b,c,d,e) on DCC02 sequence.

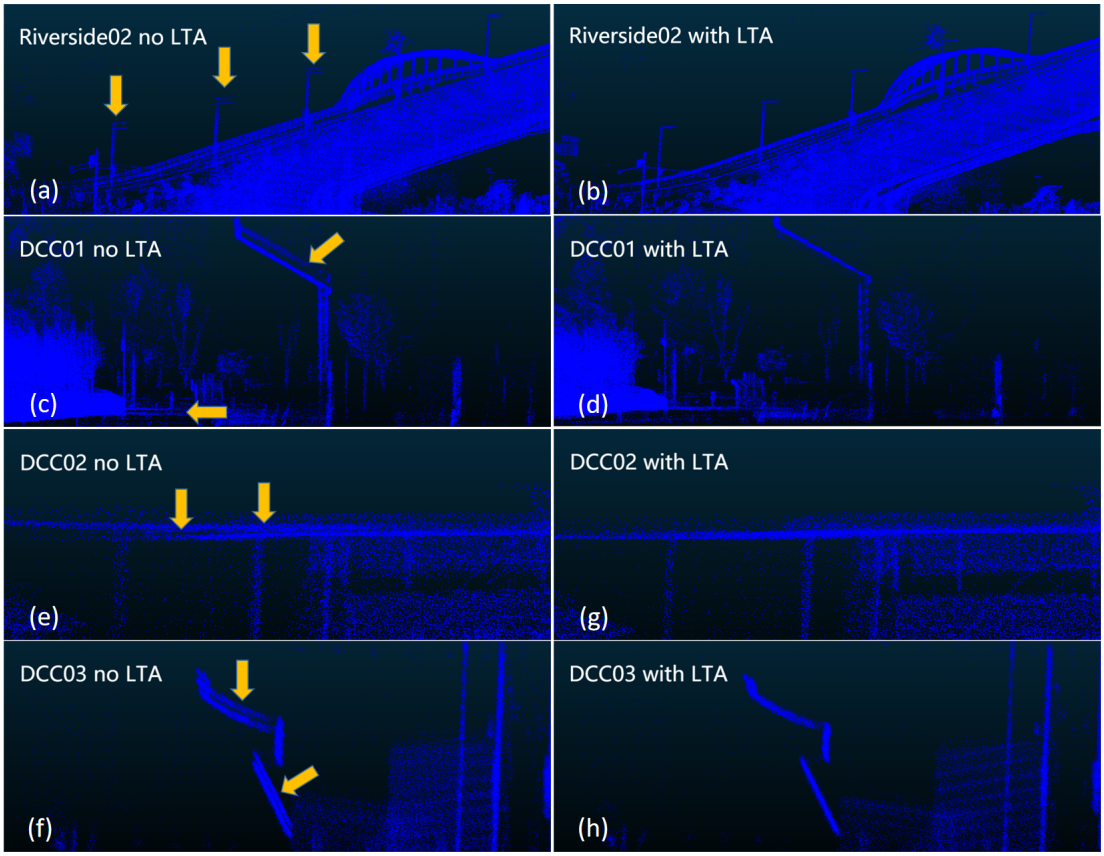

Figure 8: Map consistency improvement by long-term association. The terms "with LTA" and "no LTA" means the resultant maps using and without using long-term association, respectively.

Table 8: Multi-session absolute trajectory error (RMSE, meters) on Mulran dataset. The term "R2-1" denotes testing on R1 with prior information from R2, and other terms follow the same notation manner.

	R2-1	**R2-3**	**K3-1**	**K3-2**	**D3-1**	**D3-2**
Single-session	6.98	10.38	3.25	3.10	**5.29**	2.94
Multi-session	**6.16**	**8.36**	**3.10**	**2.99**	5.31	**2.85**

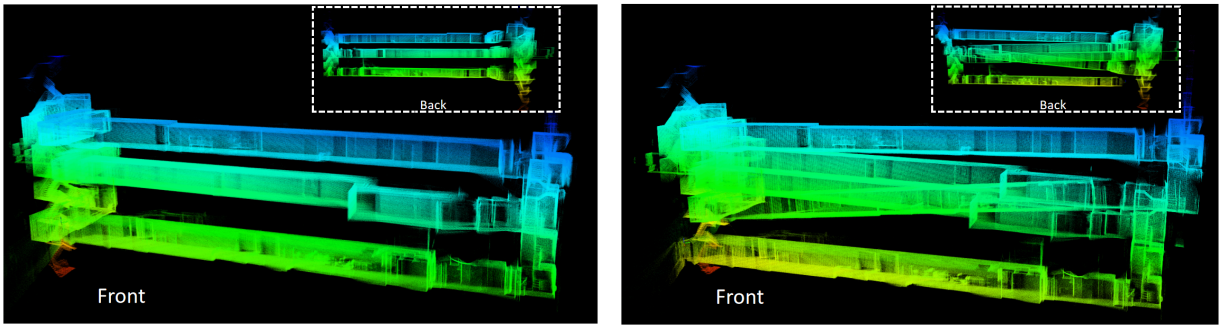

Figure 9: Comparison between the constructed map of our FPR module (the left) and Cauchy robust estimator (the right) on multi-level building dataset. There exist three similar rectangular corridors at different levels.

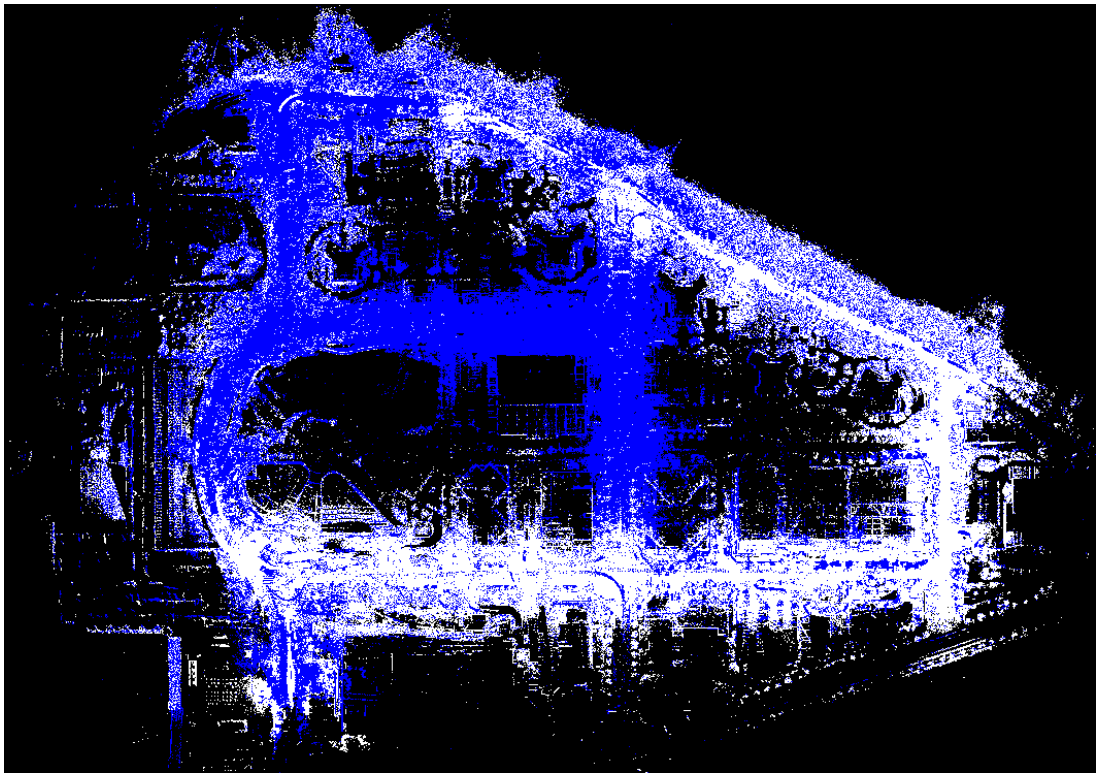

Figure 10: Map stitching result of multi-session mode. The white cloud is the pre-stored map, while the blue cloud is the live map.

Table 9: Multi-session relocalization response scan number and time (seconds) on Mulran dataset. The terms "Starting Scan" and "Finish Scan" denote the LiDAR scan indices when the platform starts moving in the pre-stored map and LTA-OM finishes relocalization, respectively.

	R2-1	**R2-3**	**K3-1**	**K3-2**	**D3-1**	**D3-2**
Starting Scan	280	78	61	63	108	80
Finish Scan	332	136	112	112	157	155
Response Scan Number	52	58	51	49	49	75
Response Time (s)	5.2	5.8	5.1	4.9	4.9	7.5

Table 10: Dataset sequences information. The sequences on the left and right are Mulran and NCLT datasets, respectively.

	Name	Duration (min:sec)	Distance (km)		**Name**	Duration (min:sec)	Distance (km)
R1	riverside01	8:59	6.5	N1	2012-01-15	111:46	7.5
R2	riverside02	10:54	6.0	N2	2012-01-22	87:19	6.1
R3	riverside03	17:05	7.3	N3	2012-02-02	98:37	6.2
K1	KAIST01	13:26	6.1	N4	2012-04-29	43:17	3.1
K2	KAIST02	14:40	6.0	N5	2012-05-11	84:32	6.0
K3	KAIST03	14:10	6.3	N6	2012-06-15	55:10	4.1
D1	DCC01	8:51	5.0	N7	2012-12-01	75:50	5.0
D2	DCC02	12:20	4.3	N8	2013-01-10	17:02	1.1
D3	DCC03	12:17	5.5	N9	2013-04-05	69:06	4.5

6 Conclusion

This paper proposes LTA-OM, an efficient, robust, and accurate LiDAR SLAM system. LTA-OM is a system-complete integration framework of FAST-LIO2, Stable Triangular Detector loop closure detection, and loop optimization. The accuracy of LTA-OM is proved to be competitive or significantly higher than those of the other state-of-the-art LiDAR systems in an exhaustive benchmark experiment. By comparing the map details, we can spot that the map consistency of LTA-OM is outstanding among the LiDAR systems. This advantage is because the proposed long-term association module brings global consistency to LTA-OM. We prove that by comparing the maps generated with and without the module. Also, the time cost of LTA-OM is significantly less than the others in the benchmark experiment. The submap accumulation policy and the elimination of ICP for geometrical verification cause this efficiency advantage.

Furthermore, LTA-OM can actively reject false-positive loop closures and construct a fine map for the scene with structurally similar places. A multi-session mode is also implemented, enabling robust relocalization, saving, and loading prior map and descriptors. This mode facilitates consistently stitching the live map with the pre-stored map and enhances extra accuracy. Besides, LTA-OM provides some excellent features for robot navigation, control, and path planning: drift-free odometry at revisited places, high-frequency and real-time odometry, and fast loop closing convergence.

Acknowledgments

The authors would also like to thank Yixi Cai for the helpful discussion.

References

Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890.

Carlevaris-Bianco, N., Ushani, A. K., and Eustice, R. M. (2016). University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035.

Chen, X., Läbe, T., Milioto, A., Röhling, T., Behley, J., and Stachniss, C. (2022). Overlapnet: a siamese network for computing lidar scan similarity with applications to loop closing and localization. *Autonomous Robots*, 46(1):61–81.

Dubé, R., Cramariuc, A., Dugas, D., Nieto, J., Siegwart, R., and Cadena, C. (2018). Segmap: 3d segment mapping using data-driven descriptors. *arXiv preprint arXiv:1804.09557*.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology.

Grisetti, G., Kümmerle, R., Strasdat, H., and Konolige, K. (2011). g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–13.

He, L., Wang, X., and Zhang, H. (2016). M2dp: A novel 3d point cloud descriptor and its application in loop closure detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 231–237. IEEE.

Huber, P. J. (1973). Robust regression: asymptotics, conjectures and monte carlo. *The annals of statistics*, pages 799–821.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235.

Kim, G., Choi, S., and Kim, A. (2021). Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Transactions on Robotics*.

Kim, G. and Kim, A. (2018). Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid.

Kim, G. and Kim, A. (2022). Lt-mapper: A modular framework for lidar-based lifelong mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7995–8002. IEEE.

Kim, G., Park, Y. S., Cho, Y., Jeong, J., and Kim, A. (2020). Mulran: Multimodal range dataset for urban place recognition. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6246–6253. IEEE.

Kim, G., Yun, S., Kim, J., and Kim, A. (2022). Sc-lidar-slam: a front-end agnostic versatile lidar slam system. *arXiv preprint arXiv:2201.06423*.

Labbe, M. and Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666. IEEE.

Latif, Y., Cadena, C., and Neira, J. (2013). Robust loop closing over time for pose graph slam. *The International Journal of Robotics Research*, 32(14):1611–1626.

Li, K., Li, M., and Hanebeck, U. D. (2021). Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 6(3):5167–5174.

Lin, J. and Zhang, F. (2020). Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3126–3131. IEEE.

Liu, Z., Suo, C., Zhou, S., Xu, F., Wei, H., Chen, W., Wang, H., Liang, X., and Liu, Y.-H. (2019). Seqlpd: Sequence matching enhanced loop-closure detection based on large-scale point cloud description for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1218–1223. IEEE.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

Pan, Y., Xiao, P., He, Y., Shao, Z., and Li, Z. (2021). Mulls: Versatile lidar slam via multi-metric linear least square. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11633–11640. IEEE.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

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

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Daniela, R. (2020). Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE.

Sünderhauf, N. and Protzel, P. (2012). Switchable constraints for robust pose graph slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380.

Xu, W., Cai, Y., He, D., Lin, J., and Zhang, F. (2022). Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*.

Ye, H., Chen, Y., and Liu, M. (2019). Tightly coupled 3d lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150. IEEE.

Yuan, C., Lin, J., Zou, Z., Hong, X., and Zhang, F. (2022). Std: Stable triangle descriptor for 3d place recognition. *arXiv preprint arXiv:2209.12435*.

Zhang, J. and Singh, S. (2014). Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA.