Large-Scale LiDAR Consistent Mapping using Hierarchical LiDAR Bundle Adjustment

Xiyuan Liu, Zheng Liu, Fanze Kong, and Fu Zhang

***Abstract*—Reconstructing an accurate and consistent large-scale LiDAR point cloud map is crucial for robotics applications. The existing solution, pose graph optimization, though it is time-efficient, does not directly optimize the mapping consistency. LiDAR bundle adjustment (BA) has been recently proposed to resolve this issue; however, it is too time-consuming on large-scale maps. To mitigate this problem, this paper presents a globally consistent and efficient mapping method suitable for large-scale maps. Our proposed work consists of a bottom-up hierarchical BA and a top-down pose graph optimization, which combines the advantages of both methods. With the hierarchical design, we solve multiple BA problems with a much smaller Hessian matrix size than the original BA; with the pose graph optimization, we smoothly and efficiently update the LiDAR poses. The effectiveness and robustness of our proposed approach have been validated on multiple spatially and timely large-scale public spinning LiDAR datasets, i.e., KITTI, MulRan and Newer College, and self-collected solid-state LiDAR datasets under structured and unstructured scenes. With proper setups, we demonstrate our work could generate a globally consistent map with around 12% of the sequence time.**

***Index Terms*—Mapping, SLAM, Localization.**

I. Introduction

RECONSTRUCTING a three-dimensional (3D) high-resolution map of the real world is of great significance in the fields of robotics, environmental and civil engineering. This 3D map could be used as a prior for autonomous service robots and as an information model for buildings and geographical measurements. Compared with the traditional 3D laser scanner, the light detection and ranging (LiDAR) sensor extraordinarily fits into this purpose due to its fast scanning rate. Moreover, it is more lightweight, cost-effective, and flexible to be carried on multiple platforms, e.g., ground or aerial vehicles and hand-held devices. In this paper, we focus on developing an accurate and consistent LiDAR mapping method for large-scale maps.

Rich research results have been presented on LiDAR-based mapping algorithms [1]–[3], which generate both point cloud maps and LiDAR odometry. Due to the accumulation of scan-to-map registration errors, odometry drift usually appears and further leads to divergence in the point cloud map. The most well-known method to refine the mapping quality (closing the gap) is pose graph optimization (PGO), which minimizes the relative pose errors between two LiDAR frames. In PGO, the relative pose estimation is assumed to follow Gaussian distribution. However, the PGO does not directly optimize the consistency of the point cloud. The divergence within the point cloud map might only be narrowed but not fully eliminated (or not even aware of). This phenomenon is more obvious when the wrong loops are detected or incorrect relative transformation estimations happen.

LiDAR bundle adjustment (BA) approach [4, 5] directly optimizes the mapping consistency by minimizing the overall point-to-plane distance, which leads to a high mapping quality necessary for mapping applications. In [4], the plane parameters are analytically solved first such that the final optimization problem is only related to the LiDAR pose. In [5], the plane parameters are eliminated in each iteration of the optimization by a Schur complement trick as in visual BA [6]. Either way, the resultant optimization is (at least) the dimension of the LiDAR pose number N, requiring $O(N^3)$ time to solve [7]. The cubic growth of the computation time has prohibited the BA for large-scale maps with large pose numbers.

To address the above issues, we propose a hierarchical LiDAR BA method to globally optimize the mapping consistency while maintaining time efficiency. This method constructs a pyramid structure of frame poses (see Fig. 1) and conducts a bottom-up hierarchical BA and a top-down PGO (see Fig. 2). The bottom-up process conducts a hierarchical BA within local windows from the bottom layer (local BA) to the top layer frames (global BA). Such design benefits the computation time since the process of local BA in each layer is suitable for parallel computation and the time complexity of each local BA is relatively low due to the small number of poses involved. One issue in the bottom-up process is that it neglects features co-visible across different local windows, which could lower the accuracy. To mitigate this issue, the top-down process constructs a pose graph from the top to the bottom layers and distributes the errors by PGO. The two process iterates until convergence.

With the hierarchical design, we could both directly optimize the consistency of the planar surfaces within the point cloud and avoid solving a cost function with large dimensions. With the PGO, we properly update the entire LiDAR poses towards convergence in a fast and reliable manner. To retain the smoothness between every two adjacent keyframes, we keep an overlap area between them by setting the stride size smaller than the window size. To further boost the optimization speed,

Manuscript received: September 21, 2022; Revised: November 6, 2022; Accepted: January 11, 2023. This paper was recommended for publication by Editor Civera Javier upon evaluation of the Associate Editor and reviewers' comments. This work was supported by the University Grants Committee of Hong Kong under the General Research Fund scheme (Project No. 17206421). (*Corresponding authors: Fu Zhang*)

Xiyuan Liu, Zheng Liu, Fanze Kong, and Fu Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China. (email: {xliuaa,u3007335,kongfz}@connect.hku.hk, fuzhang@hku.hk).

Digital Object Identifier (DOI): see top of this page.

we have applied a filter to remove outlier points and implemented CPU-based parallel processing when constructing the pyramid. In summary, our contributions are as follows:

- We propose a hierarchical bundle adjustment method to globally optimize the LiDAR mapping consistency and odometry accuracy. Our proposed approach improves the mapping quality given a good initial pose trajectory (e.g., from a pose graph optimization) and even closes the gap when the initial pose trajectory has large drifts.
- The effectiveness of our proposed work has been validated on multiple public mechanical spinning LiDAR datasets and our self-collected solid-state LiDAR dataset in both structured and unstructured scenes.

II. Related Works

Multiple approaches have been discussed in the literature on improving the mapping quality, which mainly divides into two categories: pose graph optimization and plane (bundle) adjustment-based methods. In PGO, the relative transformation (pose constraint) between two frames is estimated by ICP [8] or its variants [9]. This relative pose error is then weighted by the information matrix, which is usually the inverse of the corresponding Hessian [10] or simply a constant matrix [11]. The pose graph is optimized when the summed relative pose errors are minimized. Though computationally efficient, one important issue of PGO is that it does not directly optimize the consistency of the point cloud. Due to incorrect estimation or imprecise modeling of the relative pose constraint [12], the PGO might converge to a local minimum that considerable divergence within the point cloud could still exist [3, 13].

The plane adjustment (PA) method directly optimizes the consistency of the point cloud by minimizing the summed point-to-plane distance. In PA, each plane feature is represented by two parameters, i.e., the distance from its center to the viewpoint and the estimated plane normal vector [5]. In [14], authors concurrently optimize both the LiDAR poses and the geometric plane features. This method needs to maintain and update the parameters of all features during the optimization, whereas the total number of features will rapidly grow when the scale of the map enlarges, leaving a huge dimension of the cost function to solve. Though with the Schur complement technique, the optimization variables could be reduced to LiDAR poses only, this method is prone to generate glitches in pose estimation in real-world practices.

The bundle adjustment (BA) method improves the PA by eliminating the feature parameter prior to the optimization using a closed-form solution [4]. In [4], authors segment the point cloud into multiple voxels, each containing a plane feature. The original point-to-plane minimization problem is transformed into the minimization of the eigenvalue of points covariance in each voxel. Such a method needs to iterate through every point within each feature to derive the Hessian matrix, whose time complexity is the square of the number of points, causing a great computation demand. In the following-up work of [4], this problem is completely addressed. All the points of a feature observed by the same pose are aggregated into a point cluster which fundamentally removes the dependence of time complexity on the total point number [7] and further improves the computation accuracy and efficiency. Authors in [15, 16] also release this problem by fixing the positions of the co-visible plane features as anchors and only optimizing the related LiDAR poses. In all the above-mentioned PA and BA methods, it requires solving a nonlinear optimization problem of dimension $6N$ with N being the pose number. Solving this problem needs to solve a $6N$-dimensional linear system which usually takes $O(N^3)$ time with the Cholesky factorization. Despite the pre-conditioned conjugate gradient (PCG) algorithm that could solve this linear system with $O(N)$ time complexity, it requires further iterations, especially when the pre-conditioned matrix is of large size. Moreover, when the divergence in the map is larger than or approximates the maximum voxel size, these PA/BA methods might have a slow convergence rate.

Our proposed hierarchical BA approach is developed based on the latest BA work [7] and takes advantage of both BA and PGO. We use BA to directly minimize the point-to-plane distance and utilize PGO to smoothly and efficiently update the LiDAR poses to avoid glitches in pose estimation. With the hierarchical design, we could parallelly solve multiple BA problems with a much smaller Hessian matrix size compared with the original problem using [4, 7]. Moreover, we could flexibly set the BA parameters from the bottom layers to the top layers in accordance with the quality of the initial pose trajectory.

III. Methodology

A. Overview

The system workflow of our proposed method is illustrated in Fig. 2. This method consists of two processes, bottom-up (see Sec. III-B) and top-down (see Sec. III-C), which iterate until convergence. The inputs are raw or deskewed points from each LiDAR scan and the initial estimations of their corresponding pose in the global frame. The deskewed LiDAR points and scan poses could either be estimated from general LiDAR odometry or optimized by simultaneous locomotion and mapping (SLAM) algorithms. As shown in Fig. 1, a *local window* refers to a collection of a fixed number of LiDAR frames from the same layer. Points within the local window from one layer are aggregated as a keyframe for the next layer. The term *first layer*, also referred to *bottom layer* in the context below, describes the collection of the initial LiDAR frames and poses. Similarly, the term *second layer* represents the collection of the LiDAR keyframes and poses reconstructed from the first layer using the local BA. The term *top layer* means the collection of the last remaining LiDAR keyframes (in Fig. 1, the top layer refers to the third layer). The process of hierarchically creating LiDAR keyframes from the bottom layer to the top layer is called the bottom-up process. The process of updating bottom layer LiDAR poses by pose graph optimization is called the top-down process.

In the bottom-up process, a BA is performed on LiDAR frames within each local window (local BA) to construct keyframes from the bottom layer to the upper layer (see Fig. 1), e.g., from the first layer to the second layer and from the second layer to the third layer, etc. Meanwhile, the optimized

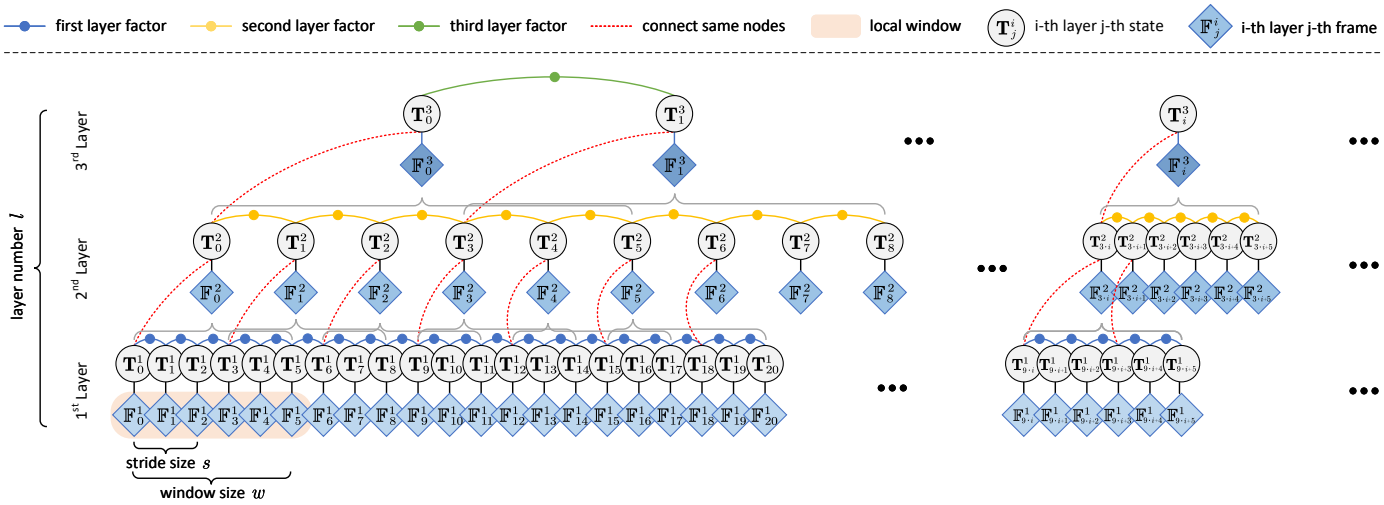

Figure 1: Pyramid structure of the proposed hierarchical bundle adjustment with *layer number l=3*, *stride size s=3* and *window size w=6*. In the bottom-up process, frames within the same local window are optimized by BA to create a keyframe for the next layer. The first frame of two adjacent local windows is *s=3* frames apart. In the top-down process, adjacent frames in the same layer are connected by factors obtained from the bottom-up BA. Frames overlapped by two (or more) local windows (e.g., nodes $\mathbf{T}_3^1$ and $\mathbf{T}_4^1$), two (or more) factors will be connected, and each is contributed from one local window BA. The red dashed line crossing two adjacent layers connects the nodes ought to be the same, e.g., $\mathbf{T}_9^1$, $\mathbf{T}_3^2$, and $\mathbf{T}_1^3$ are the same.

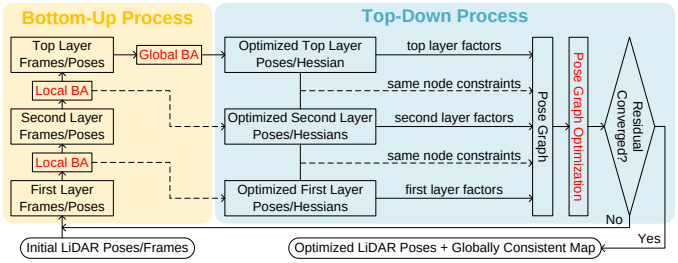

Figure 2: System overview. The light yellow region depicts the bottom-up process and the light blue region depicts the top-down process. These two processes iterate until the first layer poses are converged.

relative poses within each local window are stored for later use in the top-down process. This process is hierarchically performed until the optimal layer number is met, and then a BA is performed on the entire top layer keyframes (global BA). In the top-down process, the relative poses obtained from the bottom-up BA process are used as the constraints (factors) to construct a pose graph. The pose graph is then solved to update the first layer poses (see Fig. 4).

B. Bottom-Up Hierarchical BA

We denote $\mathbb{F}_j^i$ the j-th LiDAR frame of the i-th layer and $\mathbf{T}_j^i = (\mathbf{R}_j^i, \mathbf{t}_j^i)$, with $\mathbf{R}_j^i \in SO(3)$ and $\mathbf{t}_j^i \in \mathbb{R}^3$, is the state of the i-th layer j-th LiDAR frame. We denote $\mathbf{T}_{j,k}^i$ the relative pose between $\mathbf{T}_j^i$ and $\mathbf{T}_k^i$, i.e., $\mathbf{T}_{j,k}^i = (\mathbf{T}_j^i)^{-1} \cdot \mathbf{T}_k^i$. It is noted that points in $\mathbb{F}_j^i$ is represented in the LiDAR local frame, and $\mathbf{T}_j^i$ is in the global frame. We denote w as the local window size and s as the stride size during the bottom-up construction of the LiDAR keyframes, i.e., the starting position of every two adjacent local windows is s frames apart, and N_i the total number of LiDAR frames from the i-th layer.

In the bottom-up process, a BA is performed in each local window using the provided initial pose trajectory. For the j-th local window from the i-th layer, containing w frames $\{\mathbb{F}_{s\cdot j+k}^i \mid j = 0, \cdots, \lfloor \frac{N_i - w}{s} \rfloor; k = 0, \cdots, w-1\}$, the relative poses between the first frame and other frames in this window, i.e., $\{\mathbf{T}_{s\cdot j, s\cdot j+k}^i\}$, are optimized via a local BA [7], and the pose of the first frame is fixed to resolve the gauge freedom. The local BA constructs the Hessian matrix $\mathbf{H}$ and solves for the optimal relative poses $\{\mathbf{T}_{s\cdot j, s\cdot j+k}^{i*}\}$, with which a keyframe, denoted by $\mathbb{F}_j^{i+1}$, containing all points from this local window is constructed for the $(i+1)$-th layer (see Fig. 1 and (1a)). The pose of the keyframe, denoted by $\mathbf{T}_j^{i+1}$, is calculated by multiplying the optimal relative poses solved in all preceding local windows (see (1b)). The derived $\mathbf{H}$ from the BA in this local window is also recorded and will be used as the information matrix in the later top-down pose graph construction.

$$\mathbb{F}_j^{i+1} \triangleq \bigcup_{k=0}^{w-1} \left(\mathbf{T}_{s\cdot j, s\cdot j+k}^{i*} \cdot \mathbb{F}_{s\cdot j+k}^i \right) \tag{1a}$$

$$\mathbf{T}_j^{i+1} = \mathbf{T}_{s\cdot j}^{i*} = \prod_{k=1}^{j} \mathbf{T}_{s\cdot k-s, s\cdot k}^{i*} \tag{1b}$$

This process is repeatedly performed from the lower layer to the upper layer until the optimal layer number l is reached. It is noticed that the construction of new keyframes (local BA) does not rely on the frames outside the local window, making it suitable to concurrently use multiple local windows for parallel processing in the same layer.

Remark. *For the same frame occurring in different threads, multiple relative constraints (factors) from different local windows of different threads will be exerted on it (each contributing one factor) to ensure the pose consistency after pose graph optimization (see Fig. 1).*

Suppose we have N total number of LiDAR frames, i.e., $N = N_1$, and each time we choose to aggregate w frames from the lower layer into one frame to the upper layer with stride frame size s. Let n be the number of threads that could

be used for parallel processing. Since the computation time of BA is $O\left(M^3\right)$ with M being the number of involved poses, we could derive the overall time consumption $O\left(T_l\right)$ for a l-th layer pyramid.

The total time consumption of l-th layer pyramid includes that consumed by the local BA in each layer and that from the global BA in the top layer. For a l-th layer pyramid, the number of local windows in i-th layer ($i < l$) is $\frac{N}{s^i}$ and each local window consumes $O(w^3)$ of time. With n number of parallel threads, the total time consumption of the local BA equals the sum of the local BA in each layer which is $w^3 \cdot \left(\sum_{i=1}^{l-1} \frac{N}{s^i} \cdot \frac{1}{n}\right)$ and the global BA in the l-th layer takes $O\left(\left(\frac{N}{s^{l-1}}\right)^3\right)$ of time. In summary, $O\left(T_l\right)$ is expressed as

$$T_l = \begin{cases} N^3 & (l = 1) \\ \dfrac{w^3}{n} \cdot \displaystyle\sum_{i=1}^{l-1} \frac{N}{s^i} + \left(\frac{N}{s^{l-1}}\right)^3 & (l > 1) \end{cases} \tag{2}$$

Since T_l is a function of l, we thus calculate the optimal l^* by letting the derivative of T_l equal to zero, which leads to

$$l^* = \left\lfloor \frac{1}{2} \log_s \left(\frac{3N^2 \left(s^3 - s\right) n}{w^3} \right) \right\rfloor \tag{3}$$

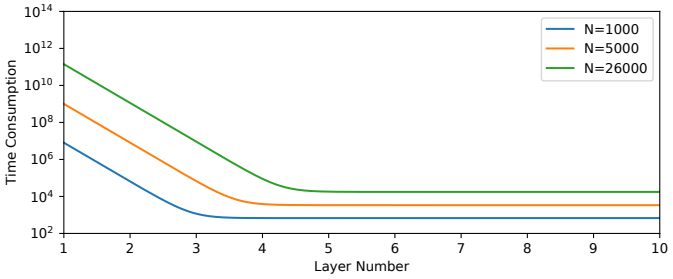

Figure 3: Example plot of time consumption T_l w.r.t. the layer number l and the pose number N using *w=10*, *s=5* and *n=8*. It is seen the overall time consumption is significantly reduced from the original BA (l=1) to our proposed hierarchical BA (l=3 or l=4).

Fig. 3 shows an example of the computation time T_l versus the layer number l at different frame numbers N. As illustrated in Fig. 3, the total computation time is greatly reduced when the layer number increases from l=1 (original BA [7]) to l^*, suggesting the effectiveness of the proposed hierarchical BA. When $l > l^*$, the computation time does not increase much and keeps almost constant, suggesting that any layer number greater than l^* will work equally well.

C. Top-Down Pose Graph Optimization

The top-down pose graph optimization process aims to reduce the pose estimation errors in the bottom-up hierarchical BA process, which considers only features co-visible in the same local windows but ignores those observed across different local windows. As shown in Fig. 1, the pose graph is constructed in a top-down manner in the pyramid structure. In each layer of the pyramid, the factors are relative poses between the adjacent two frames. Since the node $\mathbf{T}_j^{i+1}$ and $\mathbf{T}_{s\cdot j}^{i}$ are essentially the same, i.e., $\mathbf{T}_j^i = \mathbf{T}_{s\cdot j}^{i-1} = \cdots = \mathbf{T}_{s^{i-1}\cdot j}^{1}$, $\forall i \in \mathcal{L}, j \in \mathcal{F}^i$, where $\mathcal{L} = \{1, \cdots, l\}$ represents the set of l layers and $\mathcal{F}^i = \{0, \cdots, N_i - 1\}$ represents the set of N_i numbers, the original pose graph in Fig. 1 is reduced to Fig. 4 and the objective function to be minimized is thus

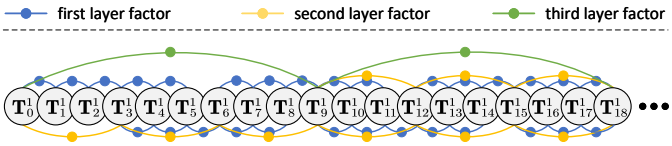

Figure 4: Final factor graph of our proposed approach with *layer number l=3*, *stride size s=3* and *window size w=6*. Factors on the same side indicate that they originate from the same local window. Nodes will be connected with multiple factors if their corresponding frames occur in multiple local windows, e.g., $\mathbf{T}_3^1$ and $\mathbf{T}_4^1$.

$$f\left(\mathbb{F}, \mathcal{T}\right) = \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{F}^i} c\left(\mathbf{T}_{s^{i-1}\cdot j}^1, \mathbf{T}_{s^{i-1}\cdot (j+1)}^1\right) \tag{4}$$

where $\mathbb{F} = \{\mathbb{F}_i^1 \mid i \in \mathcal{F}^1\}$ is the collection of all first layer frames and $\mathcal{T} = \{\mathbf{T}_i^1 \mid i \in \mathcal{F}^1\}$ represents the collection of first layer poses. The reduction of the pose graph from Fig. 1 to Fig. 4 and detailed derivation of (4) are included in Supplementary[1] Sec. A. Note that the cost function in (4) is weighted by the Hessian matrix computed in the bottom-up BA process. Finally, the factor graph is solved by the Levenberg-Marquardt method using GTSAM[2].

IV. Experiments

A. Accuracy Analysis

1) Initial Odometry with Loop Closure: In this section, we take the odometry results from the state-of-the-art (SOTA) LiDAR SLAM algorithms [1]–[3] as the input and further optimize them with our proposed work. We demonstrate that our work could both improve the mapping quality (consistency) and the pose estimation accuracy even when the initial poses have already been pose-graph optimized. The BA and pyramid parameters used in all our following experiments, without further specification, are listed in Table I (a more detailed implementation of BA is explained in Supplementary[1] Sec. B). The optimal l^* is obtained by calculation using (3) based on the actual pose number N in each sequence. For data length $N < 5 \times 10^3$ (KITTI [17], MulRan [18] and Newer College [19]), we have $l^* = 3$ and for larger sequence, i.e., $N \geq 5 \times 10^3$ (New College [20]), we have $l^* = 4$.

Table I: Hierarchical Bundle Adjustment Parameter Setting

Parameter	Value	Description
w	10	window size
s	5	stride size
n	8	threads number for parallel processing
θ_{local}	0.05	eigenvalue ratio threshold for local BA
V_{local}	4	initial voxel size for local BA
θ_{global}	0.1	eigenvalue ratio threshold for global BA
V_{global}	4	initial voxel size for global BA

We first test on the KITTI dataset [17] and use the pose estimation results from MULLS with loop closure [3] as our initial input. The RMSE of the absolute rotation (degree) and

[1]https://github.com/hku-mars/HBA/blob/main/Supplementary.pdf
[2]https://github.com/borglab/gtsam

Table II: RMSE of the ATE ($^\circ/m$) on KITTI Dataset with Loop Closure

Method	Seq. 00	Seq. 01	Seq. 02	Seq. 03	Seq. 04	Seq. 05	Seq. 06	Seq. 07	Seq. 08	Seq. 09	Seq. 10	Avg.
Proposed	**0.7/0.8**	**0.9/1.9**	**1.2**/5.1	0.7/**0.6**	**0.1**/0.8	0.5/**0.4**	**0.4/0.2**	0.4/**0.3**	1.3/2.7	0.8/1.3	0.6/1.1	**0.7/1.4**
CT-ICP [21]	**0.7**/1.7	1.0/4.2	**1.2**/4.1	**0.6**/0.7	0.2/0.7	**0.5**/0.8	**0.4**/0.3	**0.4/0.3**	1.2/2.5	**0.7/0.9**	**0.5/0.8**	**0.7**/1.5
MULLS [3]	**0.7**/1.1	**0.9/1.9**	**1.2**/5.4	0.7/0.7	0.2/0.9	0.6/1.0	**0.4**/0.3	0.4/0.4	1.3/2.9	1.0/2.1	0.6/1.1	**0.7**/1.6
LiTAMIN2 [22]	0.8/1.3	3.5/15.9	1.3/**3.2**	2.6/0.8	2.3/0.7	0.7/0.6	0.8/0.8	0.6/0.5	**0.9/2.1**	1.7/2.1	1.2/1.0	1.2/2.4
SuMa [11]	**0.7**/1.0	3.2/13.8	1.7/7.1	1.5/0.9	1.8/**0.4**	0.5/0.6	0.7/0.6	1.1/1.0	1.2/3.4	0.8/1.1	0.9/1.3	1.1/3.2
LOAM [23]	1.2/1.5	5.8/17.2	4.2/17.9	3.3/0.8	0.7/**0.4**	0.7/0.7	0.8/0.8	0.6/0.5	1.7/3.8	1.3/1.1	1.2/1.3	2.0/4.2

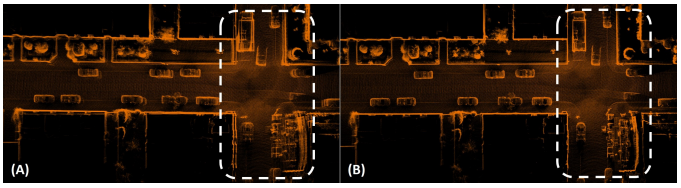

Figure 5: Mapping result from (A) MULLS [3] and (B) our proposed method on KITTI Seq. 07. The white dashed rectangle emphasizes the divergence.

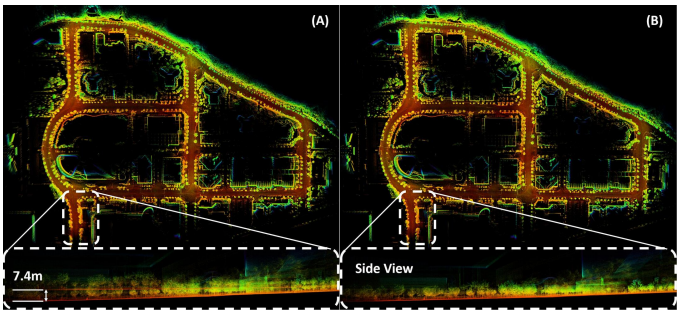

Figure 6: Point cloud mapping in sequence DCC03 of MulRan. (A): The mapping result from LIO-SAM [2] with loop closure. (B): The mapping result from our proposed method. The start (red) and ending (blue) positions have been emphasized by the white dashed rectangle and enlarged at the bottom (zoomed view is recommended).

translation (meter) errors are summarized in Table II. We choose the absolute trajectory error (ATE) as the evaluation criterion since each LiDAR frame has one unique ground truth. As can be seen, our proposed work could further improve the pose estimation accuracy, especially in translation, even when they are pose-graph optimized. Though our approach does not achieve the best result in every sequence, our work produces the optimal ATE results (0.7°/1.4m) on average compared with other SOTA methods. Moreover, since the pose graph optimization does not directly optimize the consistency of the point cloud, the divergence in the map is not fully eliminated in [3] (see Fig. 5). This divergence could be iteratively solved with our proposed hierarchical BA and pose graph optimization, which in return, reduces the ATE on pose estimations.

Table III: RMSE of the ATE (m) on MulRan Dataset with Loop Closure

Method	D01	D02	D03	K01	K02	K03	R01	R02	R03	Avg.
Proposed	**5.19**	**3.20**	**2.54**	**3.36**	**3.75**	**3.53**	**8.92**	**7.94**	**10.26**	**5.41**
LIO-SAM [2]	5.67	3.48	2.84	3.55	3.81	3.60	9.25	8.04	10.37	5.62
LEGO-LOAM [24]	6.95	5.49	6.29	5.45	5.49	5.70	19.05	16.04	30.91	11.62

D stands for DCC, K stands for KAIST and R stands for RIVERSIDE.

We then test our proposed method on another spatially large-scale spinning LiDAR dataset, MulRan [18]. The average lengths of the contained sequences DCC, KAIST and RIVERSIDE, are 4.9km, 6.1km and 6.8km, respectively. Unlike the KITTI dataset, which collects most of the data within the urban area, MulRan dataset includes more challenging scenes from the viaduct, river and woods. We choose to use the pose-graph optimized pose trajectory results from LIO-SAM [2] as our input. The RMSE of the absolute translation error has been summarized in Table III. Our proposed work could still improve the pose estimation accuracy regardless of these challenging unstructured woods scenes. However, these scenes make the LIO-SAM generate poor relative pose and covariance estimations, which further leads to partial failure in the loop closure, causing a large divergence in altitude (see Fig. 6). With our proposed hierarchical BA and pose graph optimization mechanism, this divergence could be fully eliminated, and the pose trajectory accuracy is thus further improved.

Lastly, we test our work on the sequence *long_experiment* which is the longest sequence (N=26557) in the New College dataset [20]. We choose the FAST-LIO2 [1] to provide the pose trajectory as our initial input. It is noted that these initial poses are generated without loop closure. Table IV shows the absolute translation error of our proposed and other SOTA methods. Our proposed work outperforms other loop closure-enabled SOTA methods and achieves optimal accuracy on this large time-scale sequence.

Table IV: RMSE of the ATE (m) on Newer College Dataset with Loop Closure

Method	long_experiment
Proposed	**0.26**
GICP Matching Factor [12]	0.28
FAST-LIO2 [1]	0.35
LIO-SAM [2]	0.53
CT-ICP [21]	0.58

2) Initial Odometry without Loop Closure: In this section, we demonstrate that our proposed work can converge well even when the loop is not closed in the initial pose trajectory. We first test on the KITTI dataset [17] using the initial pose trajectory estimated from MULLS [3] without loop closure. The RMSE of the absolute rotation and translation errors are summarized in Table V. As can be seen, other SOTA methods get much worse ATEs due to the lack of loop closure function, whereas our proposed work could still produce reliable pose estimations, with the absolute rotation and translation errors being largely reduced (e.g., Seq. 00 and Seq. 08) and achieving the optimal ATE on average (0.8°/1.9m). This is due to the reason that, in local BA, the strict parameters (see Table I) ensure the frames within each local window do not diverge, and the loose parameters of the top layer global BA could implicitly identify potential divergences, generating correct relative pose constraints (see Fig. 7). Though false feature

Table V: RMSE of the ATE ($^\circ/m$) on KITTI Dataset without Loop Closure

Method	Seq. 00	Seq. 01	Seq. 02	Seq. 03	Seq. 04	Seq. 05	Seq. 06	Seq. 07	Seq. 08	Seq. 09	Seq. 10	Avg.
Proposed	1.0/**1.2**	**1.0/2.4**	2.3/9.0	0.7/**0.6**	**0.2**/0.9	**0.5/0.7**	**0.3/0.2**	**0.4/0.3**	1.3/2.5	**0.7**/1.5	**0.5**/1.1	**0.8/1.9**
CT-ICP [21]	1.2/4.5	**1.0**/4.3	**1.6**/7.5	**0.6**/0.7	**0.2**/0.7	0.6/1.4	**0.3**/0.4	**0.4**/0.4	1.2/2.5	**0.7/1.3**	**0.5/0.8**	**0.8**/2.2
MULLS [3]	1.7/6.1	**1.0/2.4**	2.4/10.7	0.7/0.7	**0.2**/0.9	1.0/2.4	0.4/0.6	0.5/0.6	1.9/4.3	1.0/3.1	**0.5**/1.1	1.0/3.0
Voxel Map [25]	**0.9**/2.8	1.9/7.8	1.7/**6.1**	1.2/0.7	0.6/**0.3**	0.8/1.2	0.4/0.4	0.7/0.7	**1.1/2.3**	1.0/1.9	1.0/1.1	1.2/2.9
SuMa [11]	1.0/2.9	3.2/13.8	2.2/8.4	1.5/0.9	1.8/0.4	0.7/1.2	0.4/0.4	0.7/0.5	1.5/2.8	1.1/2.9	0.8/1.3	1.4/3.9
LiTAMIN2 [22]	1.6/5.8	3.5/15.9	2.7/10.7	2.6/0.8	2.3/0.7	1.1/2.4	1.1/0.9	1.0/0.6	1.3/2.5	1.7/2.1	1.2/1.0	1.8/5.1
LOAM [23]	1.1/2.3	4.1/17.7	7.3/37.9	3.6/0.8	0.8/0.4	0.9/2.3	0.8/0.8	0.6/0.5	1.7/3.7	1.4/1.6	1.3/1.3	2.1/6.3

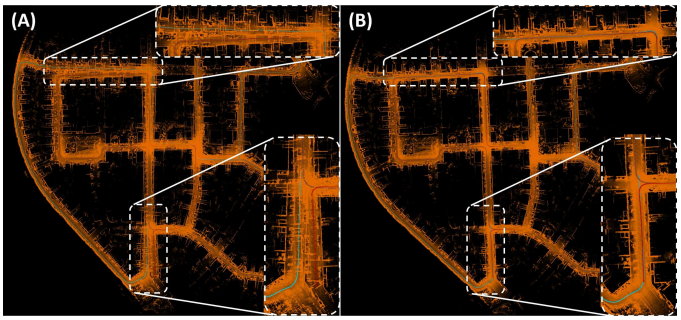

Figure 7: Closure of the gap on KITTI dataset Seq. 00 with our proposed method. The mapping result (A) is provided by MULLS [3] without loop closure, and (B) our proposed method. The odometry is colored by the moving distance from the start (red) to the end (blue). The main gaps are detailed by white dashed rectangles. The full experiment video is available on https://youtu.be/CuLnTnXVujw.

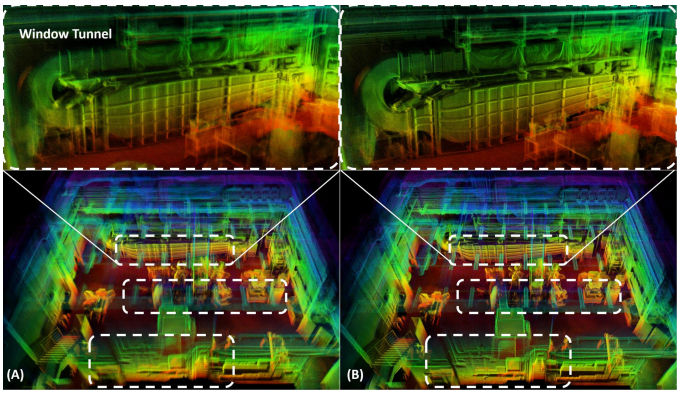

Figure 8: Reconstructed point cloud map of scene-1 using (A) FAST-LIO2 [1] (MME=-2.99) and (B) our proposed work (MME=-3.06). The main differences are emphasized by the white dashed rectangles, including the walls (bottom), ceiling lights (middle) and wind tunnel (top). Please view our experiment video for more details.

correspondence matching in global BA might happen if an incorrect top layer factor is added to the pose graph, the dense bottom layer factors ensure this incorrect factor will not drag the poses away from the correct direction.

We further validate the versatility of our proposed work on our self-collected dataset using solid-state LiDAR [26] in both structured and unstructured scenes. The first test scene is a structured indoor factory with several irregular-shaped pipelines and machines (see Fig. 8). The size of this scene is around 14×16×8m, and the length of this sequence is 7339 frames. The pyramid parameters are set the same as in Table I except that the initial voxel size is decreased due to the smaller size of the indoor scene (V_{local}=V_{global}=1m). The second test scene is an unstructured outdoor park with bush, grassland and woods around (see Fig. 9). The size of this scene is around 95×195m, and the length of this

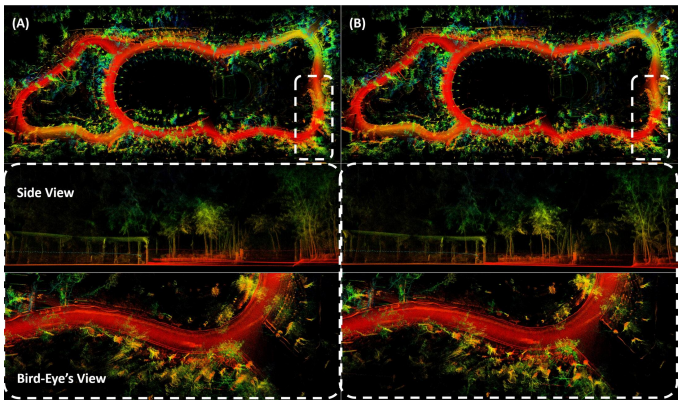

Figure 9: Reconstructed point cloud map of scene-2 using (A) FAST-LIO2 [1] (MME=-2.60) and (B) our proposed work (MME=-2.69). The main differences are emphasized by the white dashed rectangle. The second row depicts the side view of divergence in height. The third row shows the divergence from the bird-eye's view.

sequence is 3407 frames. We thus adjust the initial voxel size (V_{local}=V_{global}=2m) without changing the other pyramid parameters from Table I. For both test sequences, we use the pose estimations from FAST-LIO2 [1] without loop closure as our input. Since the measurement of the ground truth is not available, we instead use the mean map entropy (MME [27]) to evaluate the mapping quality (see Table VI). Since the MME calculates the natural logarithm of the determinant of the covariance matrix, the smaller MME is, the more consistent the point cloud is. Our proposed work further improves the mapping consistency and closes the gap in both structured and unstructured scenes regardless of the LiDAR types.

Table VI: MME on Self Collected Dataset

Method	scene-1	scene-2
FAST-LIO2 [1]	-2.99	-2.60
Proposed	**-3.06**	**-2.69**

B. Ablation Study

1) Pose Graph Optimization versus Direct Assign: In this section, we demonstrate our proposed top-down design is non-trivial. To update the bottom layer poses, a trivial method is to directly assign the optimized upper layer poses to the lower layer poses, e.g., an optimized pose of a keyframe from the upper layer is used to update the first s poses from the lower layer within the corresponding local window. For example, in Fig. 1, the poses $\mathbf{T}_0^2, \mathbf{T}_1^2, \mathbf{T}_2^2$ are updated by $\mathbf{T}_0^{3*}$ and the poses $\mathbf{T}_3^2, \mathbf{T}_4^2, \mathbf{T}_5^2$ are updated by $\mathbf{T}_1^{3*}$, respectively. We test this trivial method ("Direct Assign") with our proposed pose graph optimization mechanism on KITTI dataset [17] both using the poses generated before and after loop closure from

Table VII: RMSE of the ATE ($^\circ/m$) on KITTI Dataset

Method	Seq. 00	Seq. 01	Seq. 02	Seq. 03	Seq. 04	Seq. 05	Seq. 06	Seq. 07	Seq. 08	Seq. 09	Seq. 10	Avg.
Direct Assign*	0.68/0.81	**0.87/1.88**	1.44/5.25	0.72/**0.62**	0.19/**0.81**	**0.51**/0.52	**0.35**/0.29	**0.40/0.26**	1.28/2.76	0.86/1.59	**0.55**/1.13	0.71/1.47
Proposed*	**0.67/0.79**	**0.87**/1.91	**1.19/5.08**	**0.71**/0.63	**0.14**/0.82	0.53/**0.39**	0.36/**0.22**	**0.40**/0.30	**1.25/2.68**	**0.83/1.26**	0.57/**1.12**	**0.68/1.38**
Direct Assign	1.30/1.32	**0.97/2.37**	2.18/9.47	0.67/**0.58**	0.28/**0.87**	0.66/0.83	0.35/0.31	0.47/0.32	1.71/3.42	1.02/2.12	0.54/1.08	0.92/2.06
Proposed	**1.00/1.17**	0.98/2.41	2.26/8.95	**0.65**/0.59	**0.21**/0.90	**0.48/0.74**	**0.33/0.22**	**0.43/0.29**	**1.29/2.53**	**0.70/1.47**	**0.53/1.07**	**0.81/1.85**

* The initial pose trajectory is generated with loop closure.

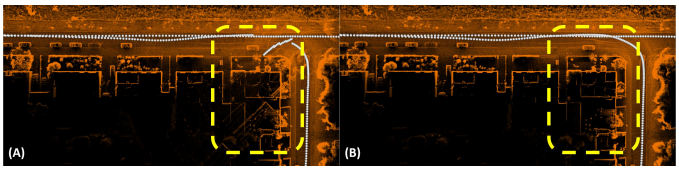

Figure 10: Mapping results (A) without and (B) with our proposed pose graph optimization in KITTI Seq. 08. The main difference is emphasized by the yellow dashed rectangle. The white dots represent the trajectory. Inconsistency occurs at the higher layer and is iteratively assigned to the bottom layer poses if no pose graph optimization is applied.

MULLS [3] as the input. The RMSE of the absolute rotation and translation errors are summarized in Table VII. As can be seen, our proposed pose graph optimization outperforms the direct assigning one both in pose estimation accuracy and mapping consistency (see Fig. 10). Though the above trivial strategy could still improve the pose estimation accuracy, such a method neglects the relative pose constraints from each local window, e.g., the pose $\mathbf{T}_3^2$ is involved in two local windows, whereas it is updated by $\mathbf{T}_1^{3*}$ only without considering the relative constraint from $\mathbf{T}_2^2$. This will lead to a mapping inconsistency between frames $\mathbb{F}_2^2$ and $\mathbb{F}_3^2$ and further between frames $\mathbb{F}_8^1$ and $\mathbb{F}_9^1$. In our proposed pose graph optimization approach, the first layer factor ensures the consistency between every adjacent frame while the second and above layer factors ensure the gap is converged towards the correct direction.

2) Hierarchical BA versus Reduced BA: In this section, we demonstrate that our proposed bottom-up design is non-trivial. To accelerate the Hessian matrix solving process, a trivial way is to keep only the block diagonal elements (of size s of the stride length) of the original Hessian matrix and solve this reduced matrix without considering the relative pose constraints among different local windows as in our method. We verify this reduced BA with the original BA [7] and our approach on the DCC sequence of the MulRan dataset [18]. The RMSE of the absolute translation error and the total optimization time of all methods are summarized in Table VIII. Our proposed work achieves a similar precision as the original BA method while drastically reducing the computation time. This is due to the reason that the original BA needs to construct an adaptive voxel map using all points (adaptive-voxel map), which quickly escalates as the involved pose number increases. Our method conducts BA in local windows, so we only have to construct an adaptive voxel map using a very small amount of points in the local window, and different local windows can be paralleled. The reduced BA actually takes longer time than the original BA due to two reasons. First, it needs to construct an adaptive voxel map similar to the original BA. Second, the reduced BA zeros the off-diagonal block elements, which leads to inaccurate Hessian estimation and significantly

Table VIII: RMSE of ATE (m) and Optimization Time on DCC Sequence of MulRan Dataset

Method	DCC01		DCC02		DCC03	
	RMSE	Time	RMSE	Time	RMSE	Time
Initial	5.67m	-	3.48m	-	2.84m	-
Original BA [7]	**5.17m**	2830.91s	**3.19m**	2631.83s	**2.54m**	3655.10s
Reduced BA	5.66m	4390.15s	3.46m	4615.90s	2.83m	7522.20s
Proposed	5.19m	**226.10s**	3.20m	**362.59s**	**2.54m**	**248.42s**

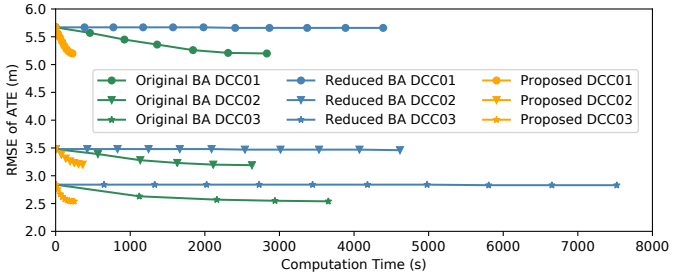

Figure 11: RMSE of ATE w.r.t. runtime of iterations of the original BA [7], reduced BA and our proposed methods on the DCC sequence of the MulRan dataset. Each marker represents one iteration.

slows down the convergence speed. In our experiments, the reduced BA may even fail to converge when the maximum iteration number is reached (*max_iter=10*) while the original BA converges within a few steps (see Fig. 11). Moreover, since we use relatively strict parameters on the local BA, factors from these layers ensure that glitches will not appear between every adjacent frame. For the simplified and the original BA, only the strict parameter could be adopted. Otherwise, false feature matching will frequently happen (points within the voxel do not form a plane feature).

C. Computation Cost

In this section, we demonstrate that our proposed approach is computationally efficient, especially on large-scale datasets. We test our proposed method with multiple setups of used layers on New College [20], and Newer College [19] datasets whose data length varies from 10^3 to 10^4 frames. The total computation time (including the adaptive-voxel map construction and BA time) and the maximum RAM memory consumption recorded for each setup at all sequences are illustrated in Fig. 12. Due to the huge time and RAM consumption of the original BA method, we do not test it on the last two sequences since the plot could already depict the trend.

It is seen for all test scenes the more layers are used, the less computation time the pyramid takes. When the pose number $N < 5 \times 10^3$, the time and RAM consumption of 3-layer and 4-layer pyramids are similar, and when $N > 5 \times 10^3$, the 4-layer pyramid becomes optimal. All these phenomenons are in accordance with our theoretical analysis shown in Fig. 3 and (3). Since the third test scene (N=2436) contains

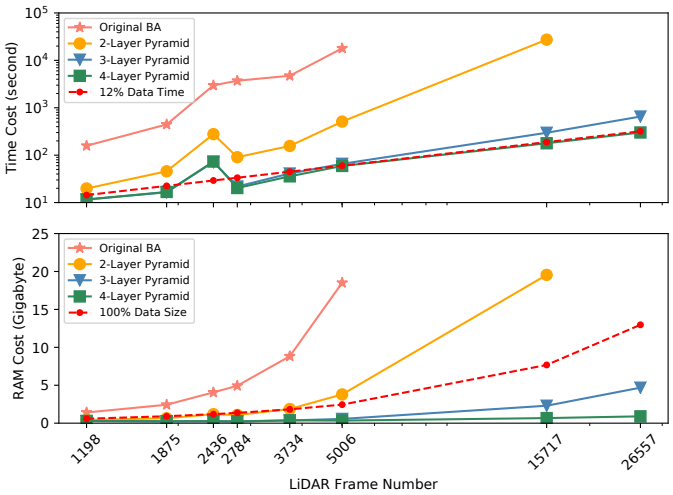

Figure 12: Comparison of computation time and RAM costs with multiple setups of layers used in the pyramid under Newer College Dataset [19, 20]. Different setups are represented using different colors. The red dashed lines represent 12% of the total data time and total point cloud size of the corresponding sequence.

a more complex environment (thus more adaptive-voxel map construction time), the total time consumption is actually larger than the latter scene. Despite this, by choosing the optimal layer setup, our work could converge within around 12% of the whole data time and consumes much smaller RAM during operation, which is suitable for practical usage.

V. Conclusion

In this paper, we propose a hierarchical BA and pose graph optimization-based work to optimize the pose estimation accuracy and mapping consistency globally for the large-scale LiDAR point cloud. With the bottom-up hierarchical BA, we parallelly solve multiple BA problems with a much smaller Hessian matrix size than the original BA method. With the top-down pose graph optimization, we smoothly and efficiently update the LiDAR poses without generating glitches. We validate the effectiveness of our work on spatially and timely large-scale LiDAR datasets with structured and unstructured scenes, given a good initial pose trajectory or with large drifts. We demonstrate our proposed work outperforms other SOTA methods in pose estimation accuracy and mapping consistency on multiple public spinning LiDAR and our self-collected solid-state LiDAR datasets. In our future work, we could combine the IMU pre-integration and LiDAR measurement noise model into our hierarchical BA work.

References

[1] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[2] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and Rus D. 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, 2020.

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

[4] Z. Liu and F. Zhang. Balm: Bundle adjustment for lidar mapping. *IEEE Robotics and Automation Letters*, 6(2):3184–3191, 2021.

[5] L. Zhou, D. Koppel, and M. Kaess. Lidar slam with plane adjustment for indoor environment. *IEEE Robotics and Automation Letters*, 6(4):7073–7080, 2021.

[6] Y. Tian, Y. Wang, M. Ouyang, and X. Shi. Hierarchical segment-based optimization for slam. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6573–6580, 2021.

[7] Z. Liu, X. Liu, and F. Zhang. Efficient and consistent bundle adjustment on lidar point clouds, 2022. doi:10.48550/arXiv.2209.08854.

[8] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[9] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2021.

[10] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[11] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[12] K. Koide, M. Yokozuka, S. Oishi, and A. Banno. Globally consistent and tightly coupled 3d lidar inertial mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5622–5628, 2022.

[13] S. Liang, Z. Cao, C. Wang, and J. Yu. A novel 3d lidar slam based on directed geometry point and sparse frame. *IEEE Robotics and Automation Letters*, 6(2):374–381, 2021.

[14] K. Ćwian, M. R. Nowicki, J. Wietrzykowski, and P. Skrzypczyński. Large-scale lidar slam with factor graph optimization on high-level geometric features. *Sensors*, 21(10):3445, May 2021.

[15] S. Liang, Z. Cao, C. Wang, and J. Yu. Hierarchical estimation-based lidar odometry with scan-to-map matching and fixed-lag smoothing. *IEEE Transactions on Intelligent Vehicles*, pages 1–1, 2022.

[16] 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*, 6(4):8269–8276, 2021.

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

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

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

[20] 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 *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360, 2020.

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

[22] M. Yokozuka, K. Koide, S. Oishi, and A. Banno. Litamin2: Ultra light lidar-based slam using geometric approximation applied with kl-divergence. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11619–11625, 2021.

[23] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Proceedings of Robotics: Science and Systems*, pages 55–63, Berkeley, USA, July 2014.

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

[25] 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*, 7(3):8518–8525, 2022.

[26] Z. Liu, F. Zhang, and X. Hong. Low-cost retina-like robotic lidars based on incommensurable scanning. *IEEE/ASME Transactions on Mechatronics*, 27(1):58–68, 2022.

[27] J. Razlaw, D. Droeschel, D. Holz, and S. Behnke. Evaluation of registration methods for sparse 3d laser scans. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2015.