2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

BALM: Bundle Adjustment for Lidar Mapping

Zheng Liu[1] and Fu Zhang[1]

***Abstract*— A local Bundle Adjustment (BA) on a sliding window of keyframes has been widely used in visual SLAM and proved to be very effective in lowering the drift. But in lidar SLAM, BA method is hardly used because the sparse feature points (e.g., edge and plane) make the exact point matching impossible. In this paper, we formulate the lidar BA as minimizing the distance from a feature point to its matched edge or plane. Unlike the visual SLAM (and prior plane adjustment method in lidar SLAM) where the feature has to be co-determined along with the pose, we show that the feature can be analytically solved and removed from the BA, the resultant BA is only dependent on the scan poses. This greatly reduces the optimization scale and allows large-scale dense plane and edge features to be used. To speedup the optimization, we derive the analytical derivatives of the cost function, up to second order, in closed form. Moreover, we propose a novel adaptive voxelization method to search feature correspondence efficiently. The proposed formulations are incorporated into a LOAM back-end for map refinement. Results show that, although as a back-end, the local BA can be solved very efficiently, even in real-time at 10Hz when optimizing 20 scans of point-cloud. The local BA also considerably lowers the LOAM drift. Our implementation of the BA optimization and LOAM are open-sourced to benefit the community[1].**

I. Introduction

Bundle adjustment (BA) is the problem of jointly solving the 3D structures (i.e., location of feature points) and camera poses [1]. It has been a fundamental problem in various visual applications, such as structure from motion (SfM) [2], visual SLAM (simultaneous localization and mapping) [3], and visual-inertial navigation [4, 5].

Similar bundle adjustment can be defined for lidar mapping where the goal is to jointly determine the lidar pose and the global 3D point-cloud map. This would be a key problem in lowering the drift in lidar SLAM. Constrained by the pairwise matching nature of existing scan registration methods, such as iterative closest points (ICP) [6], generalized ICP [7], normal distribution transform [8], and surfel-based registration [9], commonly used lidar navigation and mapping (LOAM) framework [10] and its variants [11, 12] usually build the map by incrementally registering new scans. Such an incremental mapping process would inevitably accumulate registration errors, especially in featureless environments where degeneration occurs [13] or for lidars of small FoV [14]. One way to lower such drift is performing a local BA over a sliding window of lidar scans, which allows us to re-assess the past scans based on information in new scans.

*This work was not supported by DJI (Project No. 200009538).

[1]Zheng Liu and Fu Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong. u3007335@connect.hku.hk, fuzhang@hku.hk

[1]https://github.com/hku-mars/BALM

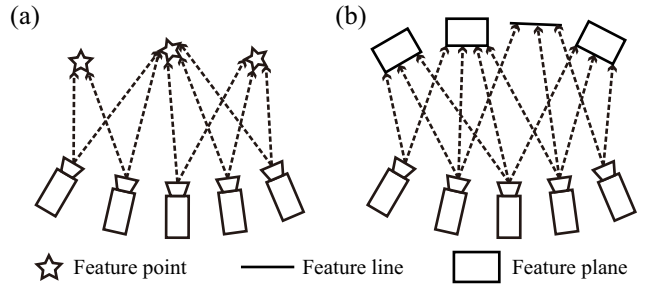


Fig. 1. Comparison of BA formulations: (a) visual BA constrains feature points to locate at the same point; (b) our proposed lidar BA constrains feature points to lie on the same edge or plane.

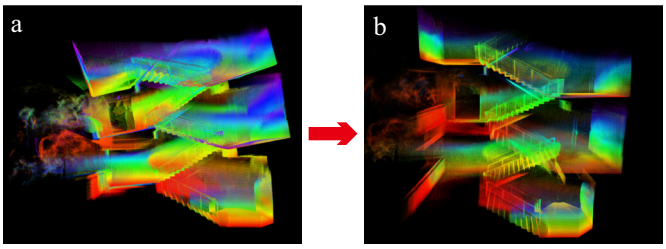


Fig. 2. (a) LOAM mapping without map refinement. (b) Refining the map using a local BA on a sliding window of lidar scans. Video available at https://youtu.be/d8R7aJmKifQ.

This method has been widely used in visual navigation and proved to be very effective [4, 5].

While lidar BA seems simpler than visual BA due to the direct depth measurements, its formulation is actually more complicated. In visual BA, the measurements are high-resolution images where each pixel corresponds to a single feature in the space (see Fig. 1(a)). Hence, a natural formulation would be to minimize the difference between the projected feature location and its actual location on the image. However, this natural formulation does not apply to lidar: lidar point-cloud is usually very sparse and even non-repetitive [12], making the exact point matching infeasible.

In this paper, we propose a formulation of lidar BA and incorporate it into a LOAM framework as the back-end to refine the incrementally built map. More specifically, our contribution is as follows: 1) We formulate the BA on sparse lidar feature points, including both edges and planes, by directly minimizing the distance from the feature point to the edge or plane (see Fig. 1(b)). Unlike visual BA which simultaneously solves the feature location and camera poses, we show that the feature (edge and plane) parameters in lidar BA can be analytically solved in closed-form solution, leading to a BA optimization over the scan poses only. Eliminating the feature parameters from the BA dramatically reduces the dimension of optimization and hence allows large-scale dense features to be optimized; 2) To enable efficient BA optimization, we analytically derive the gradient

and Hessian matrix of the cost function with respect to the scan poses; 3) We propose an adaptive voxelization to search for feature correspondence efficiently; and 4) We incorporate the proposed lidar BA into a LOAM back-end for map refinement and demonstrate its effectiveness on both spinning lidars and lidars of small FoV (e.g., Livox Horizon[2]) by comparing with existing LOAM implementations shown in Fig 2. Results show that the local BA effectively lowers the drift. Although it is designed as a back-end, the local BA runs very fast: when optimizing a sliding window of 20 scans, it runs nearly real-time at 10Hz. The BA formulation, optimization libraries, and LOAM implementations are open-sourced to the community.

II. Related work

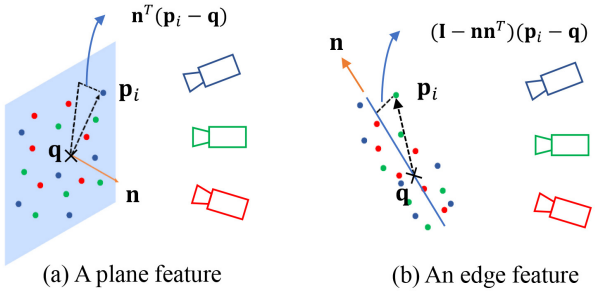
Our definition of the lidar BA is most similar to the multi-view registration. Early work in this direction [15, 16] directly extend the ICP method [6] to the multi-scans cases, where the cost function is the sum of all distance between two corresponding points in any two scans. Similarly, Neugebauer [17] uses the distance between two corresponding surfaces in any two scans. While these methods work well for dense 3D scans (e.g., depth camera), they all require exact point or surface matching that seldom exists in lidar point-cloud.

The work in [18]–[24] register any two scans sharing overlaps using standard pairwise scan registration methods. Then the obtained relative poses are used as measurements to construct a pose graph, from which the poses can be solved by graph optimization. These methods require to perform repeated pairwise scan registration among all scans having overlaps. Moreover, it does not optimize the point-cloud map directly, restricting the attainable level of mapping consistency.

The difficulty of lidar BA (or multi-view registration) lies in defining a metric that effectively evaluates the alignment quality of sparse points from all scans and, in the meantime, allows efficient optimization. The correlation (or entropy)-based scan registration in [25] naturally extends to multiple scans, however, it requires to compute the correlation between all point pairs, a computation-costly procedure that requires careful engineering [26] or GPU acceleration [27].

To lower the computation load, recent work [28]–[31] have concentrated on conducting bundle adjustment on plane features only. These work simultaneously optimize the plane parameters and scan poses, leading to an optimization of high-dimension. In contrast, our method considers both planes and edges and analytically solve both features in closed-form before the BA optimization. The resultant BA reduces to minimizing the eigenvalues over the scan poses only. With a cost function (i.e., eigenvalue) similar to [32] which uses an inefficient gradient descent optimization, we analytically derive the second order derivatives and exploit a highly efficient Gauss-Newton method to speedup the optimization. These two novel contributions, i.e., the elimination of feature (both edge and plane) parameters from the optimization, which significantly reduces the optimization dimension, and the second-order optimization, which significantly speedup the convergence, allows the BA to be conducted in nearly real-time even when optimizing very large number (e.g., a few hundreds) of features. Moreover, unlike prior methods [28]–[32]which typically require to segment planes from raw point-cloud and usually admit true plane features, we propose an adaptive voxelization to match both plane and edge features without a segmentation. This method can further adapt to various environments with both large planes (e.g., ground, wall) and small planar patches (e.g., tree crowns).

[2]https://www.livoxtech.com/horizon



(a) A plane feature (b) An edge feature

Fig. 3. A feature in space and the corresponding feature points drawn from multiple scans: (a) plane feature; (b) edge feature.

There are also some existing work on LOAM with sliding window optimization. Ye *et al.* [33] and Shan *et al* [34] optimize a sliding window of lidar scans by registering each scan in the sliding window to the map built so far. This essentially ignored all concurrent constraints among scans within the sliding window, hence leads to suboptimal solutions. Accounting for all these constraints would lead to repeated pairwise scan registration as in [35]. Droeschel *et al.* [36] uses a multi-resolution occupancy grid map which allows multi-view registration but is too costly in memory or computation [37]. Compared to these work, our method considers all constraints among all scans either in the sliding window or from the map and can be solved very efficiently.

The rest of the paper is organized as follows: In Section III, we present the theoretical framework for BA on sparse lidar points. The adaptive voxelization is presented in Section IV. We present our LOAM implementation with a local BA in Section V. The experiments are detailed in Section VI. Finally, Section VII concludes the paper and presents future work.

III. BA formulation and Derivatives

A. Direct BA formulation

Given a group of sparse feature points $\mathbf{p}_{f_i}$ $(i = 1, \cdots, N)$ drawn from M scans but all correspond to the same feature (plane or edge) (see Fig. 3). Assume the i-th feature point is drawn from the s_i-th scan, where $s_i \in \{1, \cdots, M\}$, and denote the pose of the M scans as $\mathbf{T} = (\mathbf{T}_1, \cdots, \mathbf{T}_M)$, where $\mathbf{T}_j = (\mathbf{R}_j, \mathbf{t}_j) \in SO(3) \times \mathbb{R}^3$ and $j \in \{1, \cdots, M\}$. Then, the feature point in global frame is

$$\mathbf{p}_i = \mathbf{R}_{s_i}\mathbf{p}_{f_i} + \mathbf{t}_{s_i};\ i = 1, \cdots, N. \tag{1}$$

As defined previously, the problem of lidar BA refers to jointly determining the poses of the M scans and the global 3D point-cloud map. Now the 3D map is a single feature

(edge or plane), then the BA reduces to jointly determining the poses $\mathbf{T}$ and location of the single feature, which is represented by a point $\mathbf{q}$ on the feature and a unit vector $\mathbf{n}$ ($\mathbf{n}$ is the normal vector of the plane or the direction of the edge). In case of plane feature, the direct BA formulation is to minimize the summed squared distance from each plane feature point $\mathbf{p}_i$, which depends on the pose $\mathbf{T}_{s_i}$, to the plane:

$$\begin{aligned}(\mathbf{T}^*, \mathbf{n}^*, \mathbf{q}^*) &= \arg\min_{\mathbf{T},\mathbf{n},\mathbf{q}} \frac{1}{N}\sum\nolimits_{i=1}^{N}\left(\mathbf{n}^T\left(\mathbf{p}_i-\mathbf{q}\right)\right)^2 \\ &= \arg\min_{\mathbf{T}} \underbrace{\left(\min_{\mathbf{n},\mathbf{q}} \frac{1}{N}\sum\nolimits_{i=1}^{N}\left(\mathbf{n}^T\left(\mathbf{p}_i-\mathbf{q}\right)\right)^2\right)}_{=\lambda_3(\mathbf{A});\ \text{if } \mathbf{n}^*=\mathbf{u}_3,\mathbf{q}^*=\bar{\mathbf{p}}},\end{aligned} \tag{2}$$

where $\lambda_k(\mathbf{A})$ denotes the k-th largest eigenvalue of matrix $\mathbf{A}$, $\mathbf{u}_k$ is the corresponding eigenvector, $\bar{\mathbf{p}}$ and $\mathbf{A}$ are:

$$\bar{\mathbf{p}} = \sum\nolimits_{i=1}^{N}\frac{\mathbf{p}_i}{N};\ \mathbf{A} = \frac{1}{N}\sum\nolimits_{i=1}^{N}\left(\mathbf{p}_i-\bar{\mathbf{p}}\right)\left(\mathbf{p}_i-\bar{\mathbf{p}}\right)^T \tag{3}$$

Similar to the plane feature, the direct BA formulation for an edge feature is to minimize the summed squared distance from each edge feature point $\mathbf{p}_i$ to the edge:

$$\begin{aligned}(\mathbf{T}^*, \mathbf{n}^*, \mathbf{q}^*) &= \arg\min_{\mathbf{T},\mathbf{n},\mathbf{q}} \frac{1}{N}\sum\nolimits_{i=1}^{N}\left\|\left(\mathbf{I}-\mathbf{n}\mathbf{n}^T\right)\left(\mathbf{p}_i-\mathbf{q}\right)\right\|_2^2 \\ &= \arg\min_{\mathbf{T}} \underbrace{\left(\min_{\mathbf{n},\mathbf{q}} \frac{1}{N}\sum\nolimits_{i=1}^{N}\left\|\left(\mathbf{I}-\mathbf{n}\mathbf{n}^T\right)\left(\mathbf{p}_i-\mathbf{q}\right)\right\|_2^2\right)}_{=\mathrm{Tr}(\mathbf{A})-\lambda_1(\mathbf{A})=\lambda_2(\mathbf{A})+\lambda_3(\mathbf{A});\ \text{if } \mathbf{n}^*=\mathbf{u}_1,\mathbf{q}^*=\bar{\mathbf{p}}},\end{aligned} \tag{4}$$

where $\mathrm{Tr}(\mathbf{A}) = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{p}_i-\bar{\mathbf{p}}\|_2^2$ denotes the trace of $\mathbf{A}$.

Note that in (2) and (4), the optimal point $\mathbf{q}^*$ is not unique, as the point is free to move within the plane (or along the edge). However, this has no effect on the resultant cost function to be optimized. Furthermore, (2) and (4) imply that the optimal feature (plane or edge) parameter can be analytically obtained before the BA, and the resultant BA problem is only dependent on the poses $\mathbf{T}$. This agrees well to our intuition that the 3D point-cloud map (hence the plane or edge features) are determined once the scan poses are known. Moreover, the optimization on the poses $\mathbf{T}$ reduces to minimizing the eigenvalues of the matrix $\mathbf{A}$ in (3). i.e., the BA leads to minimizing

$$\lambda_k(\mathbf{p}(\mathbf{T})), \tag{5}$$

over $\mathbf{T}$, where $\mathbf{p} = [\mathbf{p}_1^T \cdots \mathbf{p}_N^T]^T$ is the vector of all feature points corresponding to the same feature.

To allow efficient optimization with the cost in (5), we analytically derive the closed-form derivatives, up to second order, with respect to the pose $\mathbf{T}$. Due to the chain rule, we derive the derivatives with respect to the point vector $\mathbf{p}$ first.

B. The Derivatives

Theorem 1: For a group of points, $\mathbf{p}_i$ $(i = 1, \cdots, N)$ and the covariance matrix $\mathbf{A}$ defined in (3). Assume $\mathbf{A}$ has eigenvalues λ_k corresponding to eigenvectors $\mathbf{u}_k$ $(k = 1, 2, 3)$, then

$$\frac{\partial \lambda_k}{\partial \mathbf{p}_i} = \frac{2}{N}(\mathbf{p}_i-\bar{\mathbf{p}})^T\mathbf{u}_k\mathbf{u}_k^T, \tag{6}$$

where the $\bar{\mathbf{p}}$ is the average of the N points as in (3).

Theorem 2: For a group of points, $\mathbf{p}_i$ $(i = 1, \cdots, N)$ and the covariance matrix $\mathbf{A}$ defined in (3). Assume $\mathbf{A}$ has eigenvalues λ_k corresponding to eigenvectors $\mathbf{u}_k$ $(k = 1, 2, 3)$. Moreover, $\lambda_i \neq \lambda_k$ when $i \neq k$, then

$$\frac{\partial^2 \lambda_k}{\partial \mathbf{p}_j \partial \mathbf{p}_i} = \begin{cases} \frac{2}{N}\Big(\frac{N-1}{N}\mathbf{u}_k\mathbf{u}_k^T + \mathbf{u}_k(\mathbf{p}_i-\bar{\mathbf{p}})^T\mathbf{U}\mathbf{F}_k^{\mathbf{p}_j} \\ \qquad +\mathbf{U}\mathbf{F}_k^{\mathbf{p}_j}\left(\mathbf{u}_k^T(\mathbf{p}_i-\bar{\mathbf{p}})\right)\Big), & i=j \\ \frac{2}{N}\Big(-\frac{1}{N}\mathbf{u}_k\mathbf{u}_k^T + \mathbf{u}_k(\mathbf{p}_i-\bar{\mathbf{p}})^T\mathbf{U}\mathbf{F}_k^{\mathbf{p}_j} \\ \qquad +\mathbf{U}\mathbf{F}_k^{\mathbf{p}_j}\left(\mathbf{u}_k^T(\mathbf{p}_i-\bar{\mathbf{p}})\right)\Big), & i\neq j \end{cases} \tag{7}$$

$$\mathbf{F}_k^{\mathbf{p}_j} = \begin{bmatrix} \mathbf{F}_{1,k}^{\mathbf{p}_j} \\ \mathbf{F}_{2,k}^{\mathbf{p}_j} \\ \mathbf{F}_{3,k}^{\mathbf{p}_j} \end{bmatrix} \in \mathbb{R}^{3\times 3},\quad \mathbf{U} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix},$$

$$\mathbf{F}_{m,n}^{\mathbf{p}_j} = \begin{cases} \frac{(\mathbf{p}_j-\bar{\mathbf{p}})^T}{N(\lambda_n-\lambda_m)}(\mathbf{u}_m\mathbf{u}_n^T+\mathbf{u}_n\mathbf{u}_m^T), & m\neq n \\ \mathbf{0}_{1\times 3}, & m=n \end{cases}$$

C. Second order approximation

With the first and second order derivatives in previous sections, we can approximate the cost function (5) by its second order approximation as below:

$$\lambda_k(\mathbf{p}+\delta\mathbf{p}) \approx \lambda_k(\mathbf{p}) + \mathbf{J}(\mathbf{p})\delta\mathbf{p} + \frac{1}{2}\delta\mathbf{p}^T\mathbf{H}(\mathbf{p})\delta\mathbf{p}, \tag{8}$$

where $\mathbf{J}(\mathbf{p})$ is the Jacobian matrix with i-th elements in (6) and $\mathbf{H}(\mathbf{p})$ is the Hessian matrix with i-th row, j-th column elements in (7).

Recall that the point vector $\mathbf{p}$ is further dependent on the scan poses $\mathbf{T}$ as in (1). Perturbing a pose $\mathbf{T}_j$ in its tangent plane $\delta\mathbf{T}_j = \begin{bmatrix}\boldsymbol{\phi}_j^T & \delta\mathbf{t}_j^T\end{bmatrix}^T$ using the $\boxplus$ operation defined in [38], we have

$$\mathbf{T}_j = (\mathbf{R}_j, \mathbf{t}_j);\ \mathbf{T}_j \boxplus \delta\mathbf{T}_j = (\mathbf{R}_j\exp\left(\boldsymbol{\phi}_j^\wedge\right), \mathbf{t}_j+\delta\mathbf{t}_j) \tag{9}$$

and

$$\mathbf{p}_i = \mathbf{R}_{s_i}\exp(\boldsymbol{\phi}_{s_i}^\wedge)\mathbf{p}_{f_i} + \mathbf{t}_{s_i};\ \frac{\delta\mathbf{p}_i}{\delta\mathbf{T}_{s_i}} = \begin{bmatrix}-\mathbf{R}_{s_i}(\mathbf{p}_{f_i})^\wedge & \mathbf{I}\end{bmatrix} \tag{10}$$

$$\mathbf{D} = \frac{\delta\mathbf{p}}{\delta\mathbf{T}} = \begin{bmatrix} & \vdots & \\ \cdots & \mathbf{D}_{ij} & \cdots \\ & \vdots & \end{bmatrix} \in \mathbb{R}^{3N\times 6M} \tag{11}$$

$$\mathbf{D}_{ij} = \begin{cases} \frac{\delta\mathbf{p}_i}{\delta\mathbf{T}_{s_i}} & \text{for } j=s_i\in\{1,\cdots,M\} \\ \mathbf{0}_{3\times 6} & \text{for else} \end{cases} \tag{12}$$

Substituting (12) into (8) leads to

$$\lambda_k(\mathbf{T}\boxplus\delta\mathbf{T}) \approx \lambda_k + \underbrace{\mathbf{J}\mathbf{D}}_{\bar{\mathbf{J}}}\delta\mathbf{T} + \frac{1}{2}\delta\mathbf{T}^T\underbrace{\mathbf{D}^T\mathbf{H}\mathbf{D}}_{\bar{\mathbf{H}}}\delta\mathbf{T} \tag{13}$$

Finally, we use a Levenberg-Marquardt (LM) method to minimize the cost λ_k by repeatedly approximating it by the second order approximation (13). In each iteration, the solution is solved from

$$(\bar{\mathbf{H}}(\mathbf{T}) + \mu\mathbf{I})\delta\mathbf{T}^* = -\bar{\mathbf{J}}(\mathbf{T})^T, \tag{14}$$

where μ is the stepsize determined from the LM method.

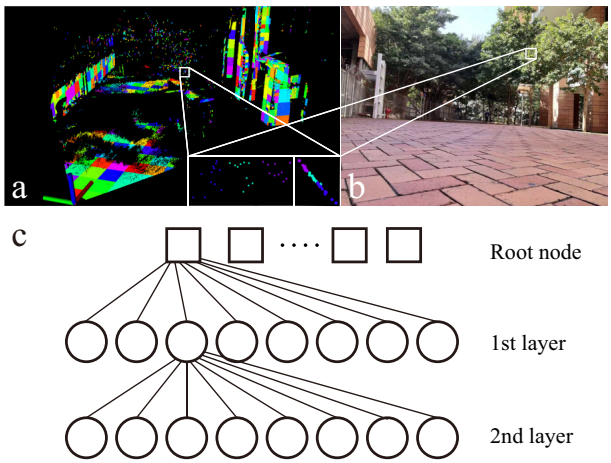


Fig. 4. (a) An exemplary voxel map, different color represents different voxels. Pictures in the lower right white box are the zoomed view of plane points on the tree crown, which contains 3 voxels with size 0.125m (left: front view; right: side view). (b) The actual environment photo. (c) All octrees are indexed in a Hash table.

IV. Adaptive Voxelization

The BA formulation in Section. III requires to find all feature points corresponding to the same feature (edge or plane). To do so, we propose a novel adaptive voxelization method: assume that a rough initial pose of different scans are available (e.g., from a LOAM odometry), we repeatedly voxelize the 3D space from a default size (e.g., 1m): if all feature points (from all scans) in the current voxel lie on a plane or edge (e.g., by examining the eigenvalue of the point covariance matrix (3)), the current voxel is kept in memory along with the contained feature points; otherwise, the current voxel breaks into eight octants and proceeds to check each octant until reaching the minimal size (e.g., 0.125m). The proposed adaptive voxelization generates a voxel map, where different voxels may have different size adapted to the environment. For each voxel, it corresponds to one feature, and hence one cost item as in (13). An exemplary voxel map is seen in Fig. 4(a).

The adaptive voxelization has many advantages: 1) It is naturally compatible with existing data structures such as octrees, hence its implementation and efficiency can be greatly facilitated; 2) It is usually more efficient than constructing a full Kd-tree on feature points [10] as early termination may occur when the contained feature points lie on the same plane or edge. Such an advantage will be more obvious when the environment has large planes or long edges; 3) A map with adaptive voxels will lower the time for searching feature correspondences in lidar odometry. It is only necessary to search the voxel a feature point lies in or near to, instead of the nearest points that require more exhaustive search [10].

In our implementation, we construct two voxel maps, one for edge features and one for planar features. The voxel map, by its construction, naturally suits to an octree structure. To reduce the depth of the octree, we use a set of octrees indexed by a Hash table (see Fig. 4(c)). Each octree corresponds a non-empty cube of the default voxel size (e.g., $1m$) in the space. Different octrees may have different depth, depending on the geometry of that cube in the space. Each leaf node (i.e., a voxel) in an octree saves feature points all corresponding to the same feature (e.g., plane or edge).

Remark 1. If a voxel contains too many points, the Hessian matrix in (8) would have a very high dimension, in this case, we could average the points from the same scan. The averaged points have fewer number and lie on the same plane determined by the raw feature points. This allows to save much computation without degrading the mapping consistency.

Remark 2. The Hessian matrix computed in Theorem 2 requires $\lambda_i \neq \lambda_k$ when $i \neq k$. For a voxel whose λ_k has algebraic multiplicity more than one, we simply skip it.

Remark 3. Although we keep saying edge features and plane features, the method naturally extends to non-planar features (e.g., curved surfaces) by constructing the voxel map at a finer level and allowing larger variance when examining whether the contained points lie on the same plane.

Remark 4. Two conditions are set to stop the recursive sub-division: one is the maximal depth of the tree and the other is the minimum number of points in a voxel.

V. LOAM with Local BA

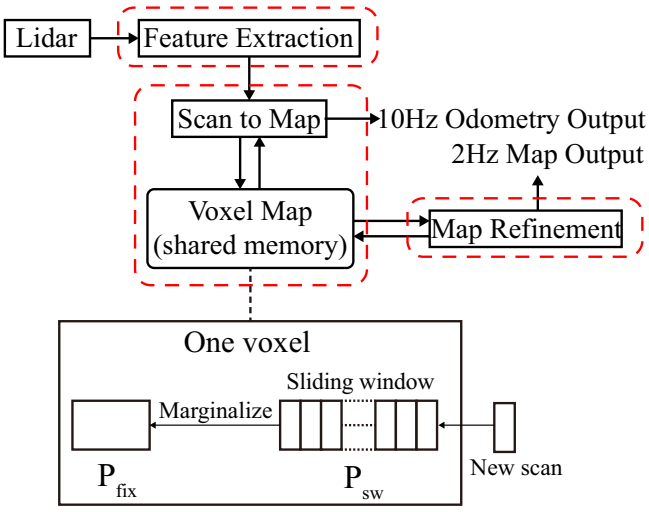


Fig. 5. Overview of LOAM with local BA.

In this section, we incorporate the proposed BA formulation and its optimization methods into a LOAM framework. The system overview is shown in Fig. 5. It consists of three parallel threads: feature extraction, odometry, and map-refinement. The feature extraction thread extracts the edge and plane features similar to [10] and [12].

Once receiving a new scan of feature points, the odometry estimates the lidar pose by registering the new scan to the existing map. Unlike the existing methods [10, 12] where each feature point is matched to some nearest points in the map, we leverage the adaptive voxel map to speedup the matching process. More specifically, when constructing the voxel map, we compute the center point and normal (or direction) vector of the plane (or edge) in a voxel. Then for a point in the new scan, we search the nearest voxel (represented by its center point) by computing the distance between the point and the plane or edge feature in the voxel.

With the odometry, the new scan can be roughly registered to the global frame and be pushed to the voxel map: for each point in the new scan, search the voxel it lies in and add this point to the leaf node of the corresponding octree. If no voxel is found in the existing map for the point in the new scan, create a new octree, index its root in the Hash table, and add this point to the root node. After all feature points of the new scan are distributed to the leaf node of existing octrees or the root node of newly created octrees, we update the voxel map as the way it is constructed: if points in a node (leaf or node) do not make a single feature (plane or edge), divide the node into eight and check each of them.

After pushing a certain number of new scans to the voxel map, a map-refinement is triggered. The map-refinement performs a local BA on a sliding window of lidar poses. Any voxel containing points within the sliding window (i.e., $\mathbf{P}_{sw}$) are used to construct cost items as (2) or (4). Then, the map-refinement repeatedly minimizes the second order approximation (13) of the total cost consisting of all relevant voxels. This refines all the lidar poses within the sliding window. The updated poses are then used to update the center points and normal vectors of all involved voxels.

Once the sliding window is full, points from older scans are merged to the map points $\mathbf{P}_{fix}$. A nice property of the point covariance matrix (3) is the existence of recursive form [39], allowing all points outside the sliding window to be summarized in a few compact matrices and vectors without saving the raw points (see lower part of Fig. 5). The merged points $\mathbf{P}_{fix}$ will be retained in the voxel map for odometry and map-refinements.

VI. Experiments

We present experimental results to verify the effectiveness of the proposed BA in LOAM. In the experiment, the lidar odometry runs at 10Hz, the map-refinement is triggered after receiving 5 scans hence running at 2Hz. We use a sliding window of 20 most recent scans. All the experiments run on a laptop computer with CPU i7-10750H and 16 GB memory. More experiment details can be found in the video available at `https://youtu.be/d8R7aJmKifQ`.

A. Livox Horizon

We test our algorithm on Livox Horizon lidar, which has a $25^\circ \times 82^\circ$ FoV, and compare its performance with that of a state-of-the-art implementation of LOAM [10] for this lidar[3]. The lidar in this experiment is handheld and moving in HKU campus (Fig. 6). The total path length is about 817m. We return to the start position after 20 minutes of walking. Fig. 6 (a) and (b) show the odometry and mapping results of BALM and LOAM, respectively. It is seen that our method successfully returns to the start point while LOAM leads to significant drift. The elevation error of our method is also much smaller (e.g., 0.27m versus 3.98m, see Fig. 6 (c) and (d)). The total translation error is summarized in Table I.

[3]`https://github.com/Livox-SDK/livox_mapping`

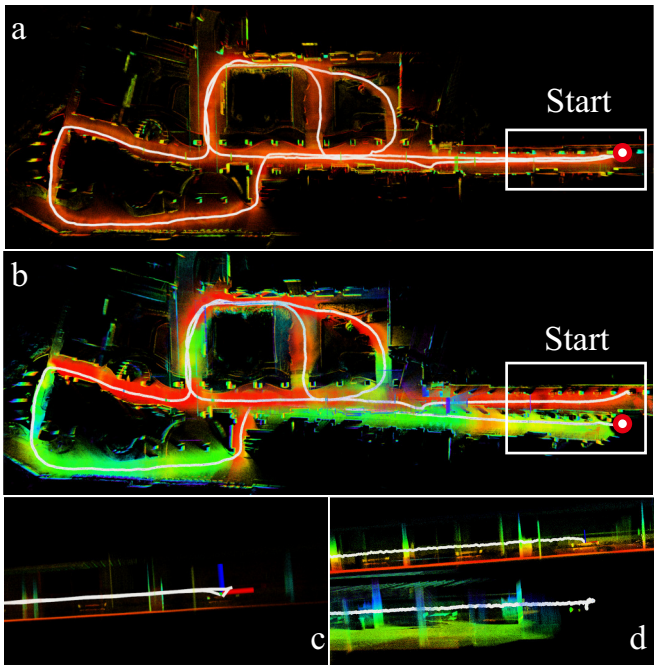


Fig. 6. Results of outdoor walking dataset: (a) the overall map built by BALM, (b) the map built by LOAM. (c) and (d) show the side view of map and odometry near to the start/end point (the white box in (a) and (b)) of our method BALM and LOAM, respectively.

TABLE I
Drift comparison on Livox Horizon lidar data.

Distance (817m)	LOAM (m)	BALM (m)
Translation error	6.228 (0.762%)	0.31 (0.038%)

We additionally conducted an indoor experiment where the sensor is handheld and moving along a stairway. Results in Fig. 2 validates the effectiveness of the proposed local BA.

B. Livox MID-40

In this experiment, we test our algorithm on Livox Mid-40 lidars[4] mounted on a UGV and compare its performance with LOAM implementation[3]. For a classic 360° spinning lidar, it is easy to turn in a corridor corner. But for the Livox Mid-40 lidar with a small 40° FoV, degeneration occurs easily because of the lessen feature points. This leads to a zigzag pose trajectory in LOAM (Fig. 7(d)). The degraded pose estimation causes map inconsistencies, which in turn worsen the following pose estimation furthermore. Hence, the LOAM odometry is falsely "raised" at the corner (Fig. 7(b)). On the other hand,although BALM produces a similarly zigzag pose trajectory due to degeneration in the front-end (i.e., scan to map) similar to LOAM, it has a much more consistent map (Fig. 7(e)), which in turn lowers the drift, due to the local BA (Fig. 7(c)).

C. Velodyne VLP-16

We further test our algorithm on Velodyne VLP-16 lidar. We use the data offered by LeGO-LOAM [11] available on Github[5], and perform comparison study. The path has the

[4]`https://www.livoxtech.com/mid-40-and-mid-100`
[5]`https://github.com/RobustFieldAutonomyLab/LeGO-LOAM`

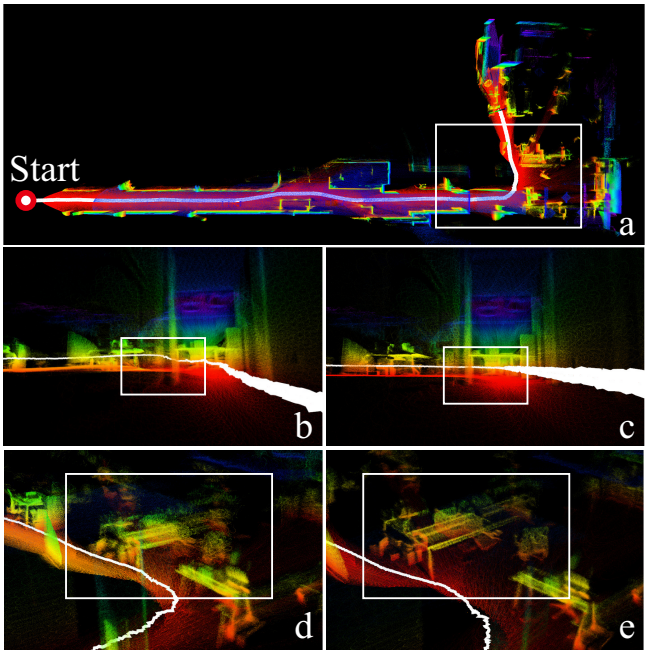


Fig. 7. Indoor mapping and odometry results: (a) the overall scene; (b) and (d) show the odometry and mapping results of LOAM; (c) and (e) show our method BLAM.

TABLE II

DRIFT COMPARISON ON VELODYNE-16 LIDAR DATA.

Distance(210m)	LOAM (cm)	LeGO-LOAM (cm)	BALM (cm)
Translation error	56.8 (0.27%)	38.5 (0.18%)	28.0 (0.13%)

same starting and end point. The scene can be seen in Fig. 8(a) and the path is colored in white. The paths of LOAM, LeGO-LOAM and BALM are shown in Fig.8(b). The drift when returning to start is given in Table II.

D. Running time

In LOAM, a feature point in new scan should find five closest points, but in BALM, the feature point just need to find the closest voxel (plane or edge), which can reduce the searching time in the scan to map. The comparison is shown in Fig. 9(a) where the running time is for building kd-tree, finding closed points/voxels and LM optimization. The data of running time is obtained by experiment A, B and C. To make a fair comparison, a fixed two-step LM optimization is used for both methods.

Finally, Fig. 9(b) shows the number of voxels for a sliding window of 20 most recent scans and the time for local BA and voxel map update. It is seen that in most cases, the local BA and voxel map update can complete in 100ms. This implies that the local BA can nearly run in real-time as the odometry (i.e., 10Hz).

VII. CONCLUSION AND FUTURE WORKS

This paper formulated a framework for lidar bundle adjustment (BA) and developed theoretical derivatives allowing efficient optimization. A novel adaptive voxelization is proposed to support the lidar BA. Then the proposed BA and optimization methods are further incorporated into a LOAM framework to serve as the back-end for map refinement.

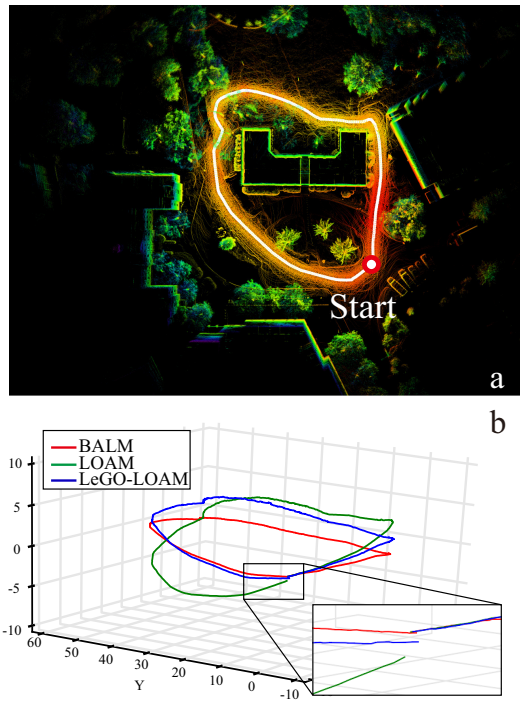


Fig. 8. Outdoor mapping and odometry results. (a) The overview of the scene; (b) The paths of LOAM, LeGO-LOAM and BALM

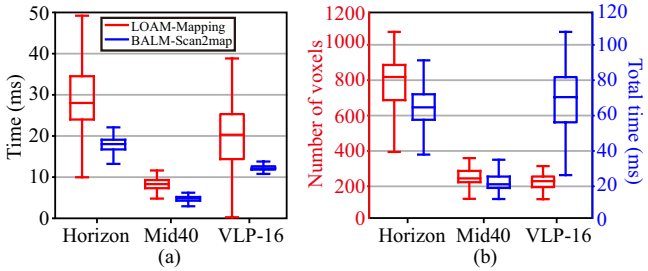


Fig. 9. (a) Time for scan to map alignment in LOAM and our method; (b) The number of voxels and computation time for a local BA over 20 scans.

Experiments on various lidars and environments validate the effectiveness of the proposed methods.

The current implementation of local BA in LOAM uses a sliding window of temporal scans, leading to redundant information in adjacent scans sharing large overlaps. Moreover, a drawback of our voxelization is the requirement of good initial poses alignment. However, the current lidar odometry uses a simple scan-to-map front-end without compensating any motion distortion or leveraging any motion model. Future works will adopt keyframes in a local sliding window and further incorporate motion models. Besides the LOAM, the proposed BA can also be used for global mapping and extrinsic calibration, which will also be explored in the future.

APPENDIX

A. Proof of theorem 1

Denote a point $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ and the eigenvector matrix $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}^T$. Further denote p an element of $\mathbf{p}_i$, p is one of x_i, y_i and z_i. Then by definition, we have

$$\mathbf{\Lambda} = \mathbf{U}^T \mathbf{A} \mathbf{U} \tag{15}$$

$$\frac{\partial \mathbf{\Lambda}}{\partial p} = \left(\frac{\partial \mathbf{U}}{\partial p}\right)^T \mathbf{A}\mathbf{U} + \mathbf{U}^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{U} + \mathbf{U}^T \mathbf{A} \frac{\partial \mathbf{U}}{\partial p} \tag{16}$$

$$\mathbf{U}^T\mathbf{A} = \mathbf{\Lambda}\mathbf{U}^T;\ \mathbf{A}\mathbf{U} = \mathbf{U}\mathbf{A} \tag{17}$$

Plugging (17) into (16) yields:

$$\frac{\partial \mathbf{\Lambda}}{\partial p} = \mathbf{U}^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{U} + \mathbf{\Lambda}\underbrace{\mathbf{U}^T \frac{\partial \mathbf{U}}{\partial p}}_{\mathbf{C}^p} + \underbrace{\left(\frac{\partial \mathbf{U}}{\partial p}\right)^T \mathbf{U}}_{(\mathbf{C}^p)^T}\mathbf{\Lambda} \tag{18}$$

As $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, partial differentiating both sides with respect to p leads to

$$\mathbf{U}^T \frac{\partial \mathbf{U}}{\partial p} + \left(\frac{\partial \mathbf{U}}{\partial p}\right)^T \mathbf{U} = \mathbf{0} \implies \mathbf{C}^p + (\mathbf{C}^p)^T = \mathbf{0}.$$

It is seen that $\mathbf{C}^p$ is a skew symmetric matrix whose diagonal elements are zeros. Moreover, since $\mathbf{\Lambda}$ is diagonal, the last two items of the right side of (18) sum to zero on diagonal positions. Only considering the diagonal elements in (18) leads to

$$\frac{\partial \lambda_k}{\partial p} = \mathbf{u}_k^T \frac{\partial \mathbf{A}}{\partial p}\mathbf{u}_k = \frac{\partial \mathbf{u}_k^T \mathbf{A}\mathbf{u}_k}{\partial p} \quad (k = 1, 2, 3)$$

where in the second equation the vector $\mathbf{u}_k$ is viewed constant. Stacking the partial differentiation of λ_k with respect to all elements of $\mathbf{p}_i$ leads to

$$\frac{\partial \lambda_k}{\partial \mathbf{p}_i} = \Big[\frac{\partial \mathbf{u}_k^T \mathbf{A}\mathbf{u}_k}{\partial x_i} \quad \frac{\partial \mathbf{u}_k^T \mathbf{A}\mathbf{u}_k}{\partial y_i} \quad \frac{\partial \mathbf{u}_k^T \mathbf{A}\mathbf{u}_k}{\partial z_i}\Big] = \frac{\partial \mathbf{u}_k^T \mathbf{A}\mathbf{u}_k}{\partial \mathbf{p}_i}$$

Recall the definition of matrix $\mathbf{A}$ in (3) and that

$$\frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_i} = \mathbf{I}, (i = j) \qquad \frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_i} = \mathbf{0}, (i \neq j),$$

Then, we can obtain

$$\begin{aligned}
\frac{\partial \lambda_k}{\partial \mathbf{p}_i} &= \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathbf{u}_k^T (\mathbf{p}_j - \bar{\mathbf{p}})(\mathbf{p}_j - \bar{\mathbf{p}})^T \mathbf{u}_k}{\partial \mathbf{p}_i} \\
&= \frac{2}{N}\sum_{j=1}^{N} (\mathbf{p}_j - \bar{\mathbf{p}})^T \mathbf{u}_k \frac{\partial \mathbf{u}_k^T (\mathbf{p}_j - \bar{\mathbf{p}})}{\partial \mathbf{p}_i} \\
&= \frac{2}{N}(\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{u}_k \mathbf{u}_k^T (\mathbf{I} - \frac{1}{N}\mathbf{I}) \\
&+ \frac{2}{N}\sum_{j=1, j\neq i}^{N} (\mathbf{p}_j - \bar{\mathbf{p}})^T \mathbf{u}_k \mathbf{u}_k^T (-\frac{1}{N}\mathbf{I}) \\
&= \frac{2}{N}(\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{u}_k \mathbf{u}_k^T. \quad \blacksquare
\end{aligned} \tag{19}$$

B. Proof of theorem 2

Consider two points, $\mathbf{p}_i = [x_i \quad y_i \quad z_i]^T$ and $\mathbf{p}_j = [x_j \quad y_j \quad z_j]^T$. Denote q a element of $\mathbf{p}_j$, q is one of x_j, y_j and z_j. Since the eigenvector matrix $\mathbf{U}$ is orthogonal, so

$$\mathbf{U}^T \frac{\partial \mathbf{U}}{\partial q} + \left(\frac{\partial \mathbf{U}}{\partial q}\right)^T \mathbf{U} = \mathbf{0}$$

Define

$$\mathbf{C}^q = \mathbf{U}^T \frac{\partial \mathbf{U}}{\partial q}, \quad \mathbf{C}^q + (\mathbf{C}^q)^T = \mathbf{0}$$

The elements on the diagonal of $\mathbf{C}^q$ is zero. Similarly with (18) and replace p with q

$$\frac{\partial \mathbf{\Lambda}}{\partial q} = \mathbf{U}^T \frac{\partial \mathbf{A}}{\partial q}\mathbf{U} + \mathbf{\Lambda}\mathbf{C}^q - \mathbf{C}^q\mathbf{\Lambda} \tag{20}$$

Since $\mathbf{\Lambda}$ is diagonal and hence $\frac{\partial \mathbf{\Lambda}}{\partial q}$, for off-diagonal elements in (20), we have

$$0 = \mathbf{u}_m^T \frac{\partial \mathbf{A}}{\partial q}\mathbf{u}_n + \lambda_m \mathbf{C}_{m,n}^q - \mathbf{C}_{m,n}^q \lambda_n$$

$\mathbf{C}_{m,n}^q$ is the m-th row and n-th column element in $\mathbf{C}^q$ as below if $\lambda_m \neq \lambda_n$

$$\mathbf{C}_{m,n}^q = \begin{cases} \dfrac{1}{\lambda_n - \lambda_m}\mathbf{u}_m^T \dfrac{\partial \mathbf{A}}{\partial q}\mathbf{u}_n, m \neq n \\ 0 \qquad\qquad\quad , m = n \end{cases} \tag{21}$$

According the definition of $\mathbf{C}^q$,

$$\frac{\partial \mathbf{u}_k}{\partial q} = \frac{\partial \mathbf{U}\mathbf{e}_k}{\partial q} = \mathbf{U}\mathbf{C}^q\mathbf{e}_k$$

where $\mathbf{e}_k$ is a 3×1 vector in which the k-th element is 1 and the rests 0. Stacking the partial differentiation of $\mathbf{u}_k$ with respect to all elements of $\mathbf{p}_j$ leads to

$$\begin{aligned}
\frac{\partial \mathbf{u}_k}{\partial \mathbf{p}_j} &= \Big[\frac{\partial \mathbf{U}\mathbf{e}_k}{\partial x_j} \quad \frac{\partial \mathbf{U}\mathbf{e}_k}{\partial y_j} \quad \frac{\partial \mathbf{U}\mathbf{e}_k}{\partial z_j}\Big] \\
&= \big[\mathbf{U}\mathbf{C}^{x_j}\mathbf{e}_k \quad \mathbf{U}\mathbf{C}^{y_j}\mathbf{e}_k \quad \mathbf{U}\mathbf{C}^{z_j}\mathbf{e}_k\big] \\
&= \mathbf{U}\big[\mathbf{C}^{x_j}\mathbf{e}_k \quad \mathbf{C}^{y_j}\mathbf{e}_k \quad \mathbf{C}^{z_j}\mathbf{e}_k\big] \\
&= \mathbf{U}\begin{bmatrix} \mathbf{C}_{1,k}^{x_j} & \mathbf{C}_{1,k}^{y_j} & \mathbf{C}_{1,k}^{z_j} \\ \mathbf{C}_{2,k}^{x_j} & \mathbf{C}_{2,k}^{y_j} & \mathbf{C}_{2,k}^{z_j} \\ \mathbf{C}_{3,k}^{x_j} & \mathbf{C}_{3,k}^{y_j} & \mathbf{C}_{3,k}^{z_j} \end{bmatrix}
\end{aligned} \tag{22}$$

Define $\mathbf{F}_{m,n}^{\mathbf{p}_j} = \big[\mathbf{C}_{m,n}^{x_j} \quad \mathbf{C}_{m,n}^{y_j} \quad \mathbf{C}_{m,n}^{z_j}\big]_{1\times 3}, m, n \in \{1, 2, 3\}$. Then stack each element $\mathbf{C}_{m,n}^{x_j}$ as in (21)

$$\mathbf{F}_{m,n}^{\mathbf{p}_j} = \begin{cases} \dfrac{1}{\lambda_n - \lambda_m}\dfrac{\partial \mathbf{u}_m^T \mathbf{A}\mathbf{u}_n}{\partial \mathbf{p}_j}, m \neq n \\ \mathbf{0} \qquad\qquad\quad , m = n \end{cases}$$

where the vector $\mathbf{u}_m$ and $\mathbf{u}_n$ are viewed constant.

By derivations similar method in (19), we can further obtain the specific form of $\mathbf{F}_{m,n}^{\mathbf{p}_j}$, as follows:

$$\mathbf{F}_{m,n}^{\mathbf{p}_j} = \begin{cases} \dfrac{(\mathbf{p}_j - \bar{\mathbf{p}})^T}{N(\lambda_n - \lambda_m)}(\mathbf{u}_m\mathbf{u}_n^T + \mathbf{u}_n\mathbf{u}_m^T), m \neq n \\ \mathbf{0} \qquad\qquad\qquad\quad , m = n \end{cases}$$

And hence (22) becomes

$$\frac{\partial \mathbf{u}_k}{\partial \mathbf{p}_j} = \mathbf{U}\begin{bmatrix} \mathbf{F}_{1,k}^{\mathbf{p}_j} \\ \mathbf{F}_{2,k}^{\mathbf{p}_j} \\ \mathbf{F}_{3,k}^{\mathbf{p}_j} \end{bmatrix} = \mathbf{U}\mathbf{F}_k^{\mathbf{p}_j} \tag{23}$$

With $\frac{\partial \lambda_k}{\partial \mathbf{p}_i}$ in (19) and $\frac{\partial \mathbf{u}_k}{\partial \mathbf{p}_j}$ in (23), we have:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{p}_j}\Big(\frac{\partial \lambda_k}{\partial \mathbf{p}_i}\Big) = &\frac{2}{N}\bigg(\mathbf{u}_k\mathbf{u}_k^T \frac{\partial (\mathbf{p}_i - \bar{\mathbf{p}})}{\partial \mathbf{p}_j} + \mathbf{u}_k(\mathbf{p}_i - \bar{\mathbf{p}})^T \frac{\partial \mathbf{u}_k}{\partial \mathbf{p}_j} \\
&+ \frac{\partial \mathbf{u}_k}{\partial \mathbf{p}_j}\Big(\mathbf{u}_k^T(\mathbf{p}_i - \bar{\mathbf{p}})\Big)\bigg)
\end{aligned}$$

$$= \frac{2}{N}\left(\mathbf{u}_k \mathbf{u}_k^T \frac{\partial(\mathbf{p}_i - \bar{\mathbf{p}})}{\partial \mathbf{p}_j} + \mathbf{u}_k (\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{U}\mathbf{F}_k^{\mathbf{p}_j} + \mathbf{U}\mathbf{F}_k^{\mathbf{p}_j}\left(\mathbf{u}_k^T(\mathbf{p}_i - \bar{\mathbf{p}})\right)\right) \tag{24}$$

It should be noted that $\mathbf{u}_k(\mathbf{p}_i - \bar{\mathbf{p}})^T$ is a matrix but $\mathbf{u}_k^T(\mathbf{p}_i - \bar{\mathbf{p}})$ is a scalar. What is more,

$$\frac{\partial(\mathbf{p}_i - \bar{\mathbf{p}})}{\partial \mathbf{p}_j} = \begin{cases} \frac{N-1}{N}\mathbf{I}, & i = j \\ -\frac{1}{N}\mathbf{I}, & i \neq j \end{cases}$$

Therefore, (24) can be rewritten as (7). ■

REFERENCES

[1] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[2] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European conference on computer vision*. Springer, 2010, pp. 29–42.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[5] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[6] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[7] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[8] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1377–1393, 2012.

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

[10] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS)*, Berkeley, CA, Jul. 2014.

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

[12] J. Lin and F. Zhang, "Loam_livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," *arXiv e-prints*, p. arXiv:1909.06700, Sep. 2019.

[13] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 809–816.

[14] Z. Liu, F. Zhang, and X. Hong, "Low-cost retina-like robotic lidars based on incommensurable scanning," *arXiv preprint arXiv:2006.11034*, 2020.

[15] G. Blais and M. D. Levine, "Registering multiview range data to create 3d computer objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 820–824, 1995.

[16] R. Benjemaa and F. Schmitt, "A solution for the registration of multiple 3d point sets using unit quaternions," in *European Conference on Computer Vision*. Springer, 1998, pp. 34–50.

[17] P. J. Neugebauer, "Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images," *International journal of shape modeling*, vol. 3, no. 01n02, pp. 71–90, 1997.

[18] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau, "Towards a general multi-view registration technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 540–547, 1996.

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

[20] K. Pulli, "Multiview registration for large data sets," in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*. IEEE, 1999, pp. 160–168.

[21] D. F. Huber and M. Hebert, "Fully automatic registration of multiple 3d data sets," *Image and Vision Computing*, vol. 21, no. 7, pp. 637–650, 2003.

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

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

[24] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 195–200.

[25] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *European conference on computer vision*. Springer, 2004, pp. 558–569.

[26] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4387–4393.

[27] W. Maddern, A. Harrison, and P. Newman, "Lost in translation (and rotation): Rapid extrinsic calibration for 2d and 3d lidars," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3096–3102.

[28] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4605–4611.

[29] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5110–5117.

[30] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "Lips: Lidar-inertial 3d plane slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 123–130.

[31] L. Zhou, D. Koppel, H. Ju, F. Steinbruecker, and M. Kaess, "An efficient planar bundle adjustment algorithm," *arXiv preprint arXiv:2006.00187*, 2020.

[32] G. Ferrer, "Eigen-factors: Plane estimation for multi-frame and time-continuous point cloud alignment." in *IROS*, 2019, pp. 1278–1284.

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

[34] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," *arXiv preprint arXiv:2007.00258*, 2020.

[35] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 181–198, 2003.

[36] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[37] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3d models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[38] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[39] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," *arXiv preprint arXiv:1909.11811*, 2019.