Large-Scale LiDAR Consistent Mapping using Hierarchical LiDAR Bundle Adjustment (Supplementary)

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

Remark. *Our code has been open-sourced on GitHub*[1].

A. Final Factor Graph Construction

The cost item in the i-th layer factor between node $\mathbf{T}_j^i$ and $\mathbf{T}_{j+1}^i$ is defined as

$$\begin{aligned} \mathbf{e}_{j,j+1}^i &= \text{Log}\left(\left(\mathbf{T}_{j,j+1}^{i*}\right)^{-1}\left(\mathbf{T}_j^i\right)^{-1}\mathbf{T}_{j+1}^i\right)^\vee \\ c\left(\mathbf{T}_j^i, \mathbf{T}_{j+1}^i\right) &= \left(\mathbf{e}_{j,j+1}^i\right)^T\left(\mathbf{\Omega}_{j,j+1}^i\right)^{-1}\mathbf{e}_{j,j+1}^i \end{aligned} \tag{1}$$

where $\mathbf{\Omega}_{j,j+1}^i$ is correlation matrix between pose $\mathbf{T}_j^i$ and $\mathbf{T}_{j+1}^i$, $i \in \mathcal{L}$ and $j \in \mathcal{F}^i$, and is computed by inverting the Hessian matrix $\mathbf{H}$ obtained from the bottom-up hierarchical BA process. Since the node $\mathbf{T}_j^{i+1}$ and $\mathbf{T}_{s\cdot j}^i$ are essentially the same, we have

$$\mathbf{T}_j^i = \mathbf{T}_{s\cdot j}^{i-1} = \cdots = \mathbf{T}_{s^{i-1}\cdot j}^1, \ \forall i \in \mathcal{L}, j \in \mathcal{F}^i$$

Therefore, the cost item in (1) reduces to

$$\begin{aligned} \mathbf{e}_{j,j+1}^i &= \text{Log}\left(\left(\mathbf{T}_{j,j+1}^{i*}\right)^{-1}\left(\mathbf{T}_{s^{i-1}\cdot j}^1\right)^{-1}\mathbf{T}_{s^{i-1}\cdot(j+1)}^1\right)^\vee \\ c\left(\mathbf{T}_{s^{i-1}\cdot j}^1, \mathbf{T}_{s^{i-1}\cdot(j+1)}^1\right) &= \left(\mathbf{e}_{j,j+1}^i\right)^T\left(\mathbf{\Omega}_{j,j+1}^i\right)^{-1}\mathbf{e}_{j,j+1}^i \end{aligned} \tag{2}$$

where $i \in \mathcal{L}$ and $j \in \mathcal{F}^i$, and the original pose graph in Fig. 1 reduces to Fig. 4 (in the main manuscript).

B. Implementation Details of BA

For feature extraction and association in each layer of the bottom-up hierarchical BA, we use an adaptive voxelization method proposed in [5] (see the main manuscript), which extracts plane features of different sizes suitable for environments of different structures. To extract these various size plane features, the entire point cloud, after being transformed into the same global frame using the initial pose trajectory, is split into multiple voxels of size V, each goes through a plane test by checking the minimum and maximum eigenvalue ratio of the contained points is smaller than a threshold, i.e., $\frac{\lambda_1}{\lambda_3} < \theta$. If the plane test passes, points in the voxel will be viewed as lying on the same plane and used in the BA. Otherwise, the voxel will be recursively split until the contained points form a plane.

The above adaptive voxelization process is time-consuming when the number of points is extremely large. To mitigate this issue, we notice that points that are not considered as plane features in lower layers will not form a plane in upper layers either. We thus use only the plane feature points from each voxel in local BA to construct the keyframe for the upper layer in the bottom-up process. This procedure further saves time for next-layer adaptive-voxel map construction and improves the computation accuracy in local BA.

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