$\mathcal{PLC}$-LiSLAM: LiDAR SLAM with Planes, Lines, and Cylinders

Lipu Zhou[1], Guoquan Huang[1], Yinian Mao[1], Jincheng Yu[2], Shengze Wang[3], and Michael Kaess[4]

***Abstract*—Planes, lines, and cylinders widely exist in man-made environments. This paper introduces a LiDAR simultaneous localization and mapping (SLAM) system using those three types of landmarks. Our algorithm has three components including local mapping, global mapping, and localization. The local and global mapping jointly adjust planes, lines, and cylinders with LiDAR poses to minimize the point-to-model cost, which is referred to as plane-line-cylinder adjustment (PLCA). We prove that, with some preprocessing, PLCA is independent of the number of points captured from the three types of landmarks, which makes efficiently solving a large-scale PLCA problem feasible. The localization component conducts real-time pose estimation through registering local planes, lines, and cylinders to the global ones, which is referred to as plane-line-cylinder registration (PLCR). We present an efficient solution for PLCR. The detection and data association may introduce errors. We correct these errors through checking the cost in the back-end. It is difficult for the registration-based algorithm, such as LOAM and ICP, to correct these errors, as they do not maintain the data association. Experimental results show that out algorithm outperforms the state-of-the-art LiDAR SLAM algorithms and achieves real-time performance.**

***Index Terms*—LiDAR SLAM, Optimization, Range Sensing**

I. Introduction

LIDARs have been widely used in various robots to perceive the environment. Simultaneous localization and mapping (SLAM) using LiDAR is a fundamental problem in the robotics community. This paper focuses on investigating LiDAR SLAM using planes, lines, and cylinders, and we seek to achieve real-time performance.

In our previous works [1], [2], we explored planes for LiDAR SLAM, and introduced plane adjustment (PA) to correct the drift, which is a counterpart of the bundle adjustment (BA) in visual SLAM. As only planes are considered, these methods will fail if there are not enough planes for pose estimation. This paper extends our work [2] with lines and cylinders. Specifically, our algorithm has three components including local mapping, global mapping, and localization. In local and global mapping, planes, lines, and cylinders are jointly optimized with poses. We call the resulting least-squares problems local and global plane-line-cylinder adjustment (PLCA), respectively. In localization, the pose of a new LiDAR scan is estimated by scan-to-model continuous-time registration using planes, lines, and cylinders, referred to as continuous-time plane-line-cylinder registration (PLCR). The contributions of this paper are as follows:

- We prove that, with some preprocessing, the iterative minimization of local and global PLCA is independent of the number of points captured from planes, lines, and cylinders. As these objects are unbounded, many points can be captured from them in a LiDAR scan. Our algorithm makes efficiently solving a large-scale PLCA problem feasible.
- We present an efficient solution for PLCR. We adopt the first-order Taylor expansion to approximate the rotation, under the assumption of a small rotation between two subsequent scans, which is generally true. We iterate this step if a fast rotating motion occurs.
- Our algorithm can tolerate certain detection errors. Due to the occlusion and the sparseness of the LiDAR point cloud, the detection process is prone to introducing errors, as shown in Fig. 1 and Fig. 2. It is difficult to solve this problem in the registration framework, such as LOAM and ICP, where the data association is not well maintained. Our algorithm corrects the detection errors by checking the cost in the back-end.

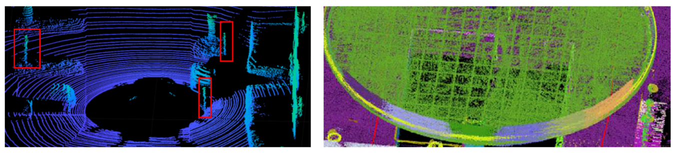
Fig. 1. Planes and lines are the most commonly used landmarks in LiDAR SLAM systems [1], [3]–[5]. But only considering the two types of landmarks may lead to problems. As the LiDAR point cloud is sparse, it is easy to misidentify a thin cylinder as a line (left image). On the other hand, a large cylinder is prone to be misidentified as small planar patches (right image). Thus it is necessary to incorporate cylinders into the LiDAR SLAM system, and develop an effective method to correct potential detection errors.

II. Related Work

LiDAR SLAM has been extensively studied. Point cloud registration is generally used for LiDAR SLAM. This category includes various ICP algorithms [4], [6]–[8], surfel-based algorithm [9], LOAM [3] and its variants [10]–[15]. These algorithms estimate the pose through aligning two point clouds. Then the 3D map is generated by simply assembling the scans using the estimated poses. This may form a vicious loop. Specifically, the pose error will degrade the quality of the map, which will in turn further reduce the quality of the pose

Manuscript received February 24, 2022; accepted May 16, 2022. Date of publication June 3, 2022; date of current version June 17, 2022. This letter was recommended for publication by Associate Editor J. Zhang and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by Tsinghua University - Meituan Joint Institute for Digital Life, and was partially done when Lipu Zhou and Shengze Wang were at Carnegie Mellon University.

[1]Lipu Zhou, Guoquan Huang, and Yinian Mao are with Meituan, Beijing 100012, China. {zhoulipu, huangguoquan, maoyinian}@meituan.com

[2]Jincheng Yu is with Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. yu-jc@tsinghua.edu.cn

[3]Shengze Wang is with Department of Computer Science, University of North Carolina, 3175 Brooks, Chapel Hill, NC 27599, USA. shengzew@cs.unc.edu

[4]Michael Kaess is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. kaess@cmu.edu

Digital Object Identifier 10.1109/LRA.2022.3180116

estimation. In addition, as the data association is calculated on the fly, it is difficult to handle the incorrect feature detection.

Jointly optimizing the landmarks and the poses can reduce the drift. Planes are widely used in previous works for this purpose [1], [2], [16]–[19]. However, only using the plane alone will limit the applicable scenarios of the algorithm. In [5], lines and planes are introduced into LiDAR SLAM. The sliding window strategy is adopted for the optimization. Assume that there are N points that are captured from a line or a plane in one scan. The computational complexity and memory consumption for calculating the Hessian matrix related to these points are $O(N^2)$. So planes and lines in [5] are divided into pieces to make the optimization feasible, which loses the ability to establish the long-term data association and may result in a suboptimal result. This paper shows that the iterative minimization process can be independent of N with some preprocessing, which can significantly reduce the computational cost.

In addition to planes and lines, cylinders are also common in various scenarios, such as trunks and lamp poles. However, the number of works on using cylinders for LiDAR SLAM is relatively small. In [20], cylinders are projected into the ground to form circles for SLAM in $SE(2)$. This algorithm cannot employ the tilted cylinder and is not suitable for SLAM in $SE(3)$. In [21] and [22], cylinders are used to model trunks in the forest environment. But, they do not focus on how to accelerate the computation. This paper seeks to address this problem.

It is known that loop closure can correct the drift of a SLAM system. Loop closure is generally considered to occur when a robot returns back to a previously visited place [1], [15], [23]–[26]. Essentially, the loop closure is to establish the data association between the current observations and previously visited landmarks. As planes, lines and cylinders are unbounded objects, it is not necessary to return back to a previous place to revisit them. Similar to [2], we trigger the global optimization when these unbounded objects are revisited. This can correct the drift at an earlier stage.

III. Notations and Preliminaries

This paper uses italic, boldfaced lowercase and boldfaced uppercase letters to represent scalars, vectors and matrices, respectively.

Pose This paper represents a pose by a transformation matrix $\mathbf{X} = \begin{bmatrix} \mathbf{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$ which converts the coordinates of a point in a local LiDAR coordinate system to the global coordinate system. Here $\mathbf{R} \in SO(3)$ is the rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is the translation vector. In PLCR and PLCA, $\mathbf{X}$ is parameterized by $\boldsymbol{x} = [\boldsymbol{\omega}; \boldsymbol{t}]$, where $\boldsymbol{\omega}$ is the angle-axis representation of $\mathbf{R}$:

$$exp([\boldsymbol{\omega}]_\times) = \mathbf{I}_3 + \frac{sin(||\boldsymbol{\omega}||_2)}{||\boldsymbol{\omega}||_2}[\boldsymbol{\omega}]_\times + \frac{1 - cos(||\boldsymbol{\omega}||_2)}{||\boldsymbol{\omega}||_2^2}[\boldsymbol{\omega}]_\times^2, \quad (1)$$

where $[\boldsymbol{\omega}]_\times$ represents the skew symmetric matrix of $\boldsymbol{\omega}$ and $||\boldsymbol{\omega}||_2$ denotes the L^2 norm of $\boldsymbol{\omega}$.

Plane We represent a plane as $\boldsymbol{\pi} = [\boldsymbol{n}; d]$, where $\boldsymbol{n}$ is the normal of the plane with $||\boldsymbol{n}||_2 = 1$, and $|d|$ equals to the distance from the origin to the plane. In PLCA, $\boldsymbol{\pi}$ is represented by the closest-point parameterization $\boldsymbol{\eta} = d\boldsymbol{n}$ [27].

Line We apply the Plücker coordinates [28] to represent a 3D line, which is a six-dimensional vector $\boldsymbol{l} = [\boldsymbol{d}; \boldsymbol{m}]$, where $\boldsymbol{d}$ is the direction of $\boldsymbol{l}$ with $||\boldsymbol{d}||_2 = 1$, and $\boldsymbol{m}$ is perpendicular to the plane determined by the origin and $\boldsymbol{l}$, and $||\boldsymbol{m}||_2$ equals to the distance from the origin to $\boldsymbol{l}$. $\boldsymbol{l}$ has four degrees of freedom. In PLCA, we adopt the method introduced in [29] to parameterize $\boldsymbol{l}$. Specifically, given $\boldsymbol{l}$, we first construct a rotation matrix $\mathbf{R}^{\boldsymbol{l}} = [\boldsymbol{d}, \frac{\boldsymbol{m}}{||\boldsymbol{m}||_2}, \boldsymbol{d} \times \frac{\boldsymbol{m}}{||\boldsymbol{m}||_2}]$, where $\times$ denotes the cross product. Assuming $\boldsymbol{\omega}^{\boldsymbol{l}}$ is the angle-axis representation of $\mathbf{R}^{\boldsymbol{l}}$ in (1), we can parameterize $\boldsymbol{l}$ by a four-dimensional vector $\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\omega}^{\boldsymbol{l}}; & ||\boldsymbol{m}||_2 \end{bmatrix}$.

Cylinder We denote a cylinder as $\boldsymbol{c} = [\boldsymbol{l}^{\boldsymbol{c}}; r]$, where $\boldsymbol{l}^{\boldsymbol{c}}$ is the Plücker coordinates of the central line of $\boldsymbol{c}$, and r is its radius. In PLCA, we parameterize $\boldsymbol{c}$ by a five-dimensional vector $\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\zeta}^{\boldsymbol{c}}; & r \end{bmatrix}$, where $\boldsymbol{\zeta}^{\boldsymbol{c}}$ is the parameterization of $\boldsymbol{l}^{\boldsymbol{c}}$ defined above.

Observation Let us use $\boldsymbol{m}$ to represent a plane, a line or a cylinder, *i.e.*, $\boldsymbol{m} \in \{\boldsymbol{\pi}, \boldsymbol{l}, \boldsymbol{c}\}$. Suppose that $\boldsymbol{m}_j$ denotes the jth landmark, and assume that $\boldsymbol{m}_j$ is observed at pose $\mathbf{X}_i$. The observations of $\boldsymbol{m}_j$ at $\mathbf{X}_i$ form a set of $N_{ij}^{\boldsymbol{m}}$ points, denoted as $\mathbb{P}_{ij}^{\boldsymbol{m}} = \{\boldsymbol{p}_{ijk}^{\boldsymbol{m}}\}_{k=1}^{N_{ij}^{\boldsymbol{m}}}$. We denote the homogeneous coordinates of $\boldsymbol{p}_{ijk}^{\boldsymbol{m}}$ as $\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{m}} = [\boldsymbol{p}_{ijk}^{\boldsymbol{m}}; 1]$. Stacking the homogeneous coordinates of the points in $\mathbb{P}_{ij}^{\boldsymbol{m}}$, we get a $N_{ij}^{\boldsymbol{m}} \times 4$ matrix

$$\mathbf{P}_{ij}^{\boldsymbol{m}} = [\cdots; (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{m}})^T; \cdots]. \quad (2)$$

As planes, lines and cylinders are unbounded objects, $N_{ij}^{\boldsymbol{m}}$ can be very large, which leads to a large-scale least-squares problem.

Optimization The Levenberg-Marquardt (LM) [30] algorithm is generally used to solve a least-squares problem. Given a least-squares problem $\hat{\boldsymbol{\chi}} = \arg\min ||\boldsymbol{\epsilon}(\boldsymbol{\chi})||_2^2$, the LM algorithm updates the solution by $\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_k + \mathit{\Delta}$ at the kth iteration. Here $\mathit{\Delta}$ is the solution of the following linear equation system

$$(\mathbf{J}_{\boldsymbol{\epsilon}}^T \mathbf{J}_{\boldsymbol{\epsilon}} + \lambda \mathbf{I})\mathit{\Delta} = -\mathbf{J}_{\boldsymbol{\epsilon}}^T \boldsymbol{\epsilon}, \quad (3)$$

where $\mathbf{J}_{\boldsymbol{\epsilon}}$ is the Jacobian matrix of $\boldsymbol{\epsilon}$ at $\boldsymbol{\chi}_k$, and λ is the parameter of the LM algorithm, which is adjusted at each iteration to ensure that the value of $||\boldsymbol{\epsilon}(\boldsymbol{\chi})||_2^2$ reduces.

To apply the LM algorithm, we generally first compute $\mathbf{J}_{\boldsymbol{\epsilon}}$ and $\boldsymbol{\epsilon}$, and then substitute them into (3) to compute $\mathit{\Delta}$. As $N_{ij}^{\boldsymbol{m}}$ can be very large, this straightforward approach may be computationally demanding. On the other hand, **it is clear that only $\mathbf{J}_{\boldsymbol{\epsilon}}^T \mathbf{J}_{\boldsymbol{\epsilon}}$ and $\mathbf{J}_{\boldsymbol{\epsilon}}^T \boldsymbol{\epsilon}$ are required to get $\mathit{\Delta}$.** We show that $\mathbf{J}_{\boldsymbol{\epsilon}}^T \mathbf{J}_{\boldsymbol{\epsilon}}$ and $\mathbf{J}_{\boldsymbol{\epsilon}}^T \boldsymbol{\epsilon}$ of PLCA have special structures, which can be used to accelerate the computation.

Quadratic Form A symmetric matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ determines a quadratic form $q_{\mathbf{A}} = \boldsymbol{y}^T \mathbf{A} \boldsymbol{y}$, where $\boldsymbol{y} = [y_1; \cdots; y_n]$. We can write $q_{\mathbf{A}}$ as

$$q_{\mathbf{A}} = \boldsymbol{y}^T \mathbf{A} \boldsymbol{y} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} y_i y_j = \boldsymbol{\alpha}^T \boldsymbol{\chi}, \quad (4)$$

where $\boldsymbol{\chi} = [\ \cdots; y_i y_j; \cdots\ ]$ $(i \leq j)$ and $\boldsymbol{\alpha} = [\ \cdots; c_{ij} a_{ij}; \cdots\ ]$ (if $i = j$, $c_{ij} = 1$ and if $(i < j)$, $c_{ij} = 2$).

Matrix Calculus Assume $\boldsymbol{z} = [z_1; \cdots; z_m]$ and $\boldsymbol{z}$ is a vector function of $\boldsymbol{y}$ defined above. Suppose $\mathbf{P} \in \mathbb{R}^{k \times m}$ is a constant $k \times m$ matrix. Let us define $\boldsymbol{f} = \mathbf{P}\boldsymbol{z}$ and denote the Jacobian matrix of $\boldsymbol{f}$ as $\mathbf{J}_{\boldsymbol{f}}$. According to the matrix calculus rule, $\mathbf{J}_{\boldsymbol{f}}$ has the form

$$\mathbf{J}_f = \mathbf{P}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}, \text{ where } \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} = \left[\frac{\partial z_j}{\partial y_i}\right] \in \mathbb{R}^{m \times n}. \tag{5}$$

Here $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}$ is a $m \times n$ matrix, and $\frac{\partial z_j}{\partial y_i}$ is the entry at the ith row and jth column of $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}$.

IV. Point-to-Model Residual

Here we describe the point-to-model residuals for PLCA. The residuals for PLCR in Section VIII can be easily derived from them.

Point-to-Plane Residual We denote the jth plane as $\boldsymbol{\pi}_j$ and its parameterization as $\boldsymbol{\eta}_j$. The residual between $\boldsymbol{\pi}_j$ and $\boldsymbol{p}_{ijk}^{\boldsymbol{\pi}} \in \mathbb{P}_{ij}^{\boldsymbol{\pi}}$ observed at $\mathbf{X}_i$ has the form

$$\delta_{ijk}^{\boldsymbol{\pi}}(\boldsymbol{\eta}_j, \boldsymbol{x}_i) = \boldsymbol{\pi}_j^T \mathbf{X}_i \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}}. \tag{6}$$

Stacking $\delta_{ijk}^{\boldsymbol{\pi}}(\boldsymbol{\eta}_j, \boldsymbol{x}_i)$ in (6) for the $N_{ij}^{\boldsymbol{\pi}}$ points in $\mathbb{P}_{ij}^{\boldsymbol{\pi}}$, we get the residual vector $\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}(\boldsymbol{\eta}_j, \boldsymbol{x}_i) = [\cdots; \delta_{ijk}^{\boldsymbol{\pi}}(\boldsymbol{\eta}_j, \boldsymbol{x}_i); \cdots]$ for $\boldsymbol{\pi}_j$ at $\mathbf{X}_i$.

Point-to-Line Residual Assume that the jth line $\boldsymbol{l}_j = [\boldsymbol{d}_j; \boldsymbol{m}_j]$ with the parameterization $\boldsymbol{\zeta}_j$ is observed at $\mathbf{X}_i$. The residual vector between $\boldsymbol{l}_j$ and $\boldsymbol{p}_{ijk}^{\boldsymbol{l}} \in \mathbb{P}_{ij}^{\boldsymbol{l}}$ can be written as

$$\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j, \boldsymbol{x}_i) = \boldsymbol{m}_j - (\mathbf{R}_i \boldsymbol{p}_{ijk}^{\boldsymbol{l}} + \boldsymbol{t}_i) \times \boldsymbol{d}_j, \tag{7}$$

where $\mathbf{R}_i$ and $\boldsymbol{t}_i$ are the rotation matrix and translation vector of $\mathbf{X}_i$, respectively. We can concatenate $\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j, \boldsymbol{x}_i)$ for the $N_{ij}^{\boldsymbol{l}}$ points in $\mathbb{P}_{ij}^{\boldsymbol{l}}$ to get the residual vector $\boldsymbol{\delta}_{ij}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j, \boldsymbol{x}_i) = [\cdots; \boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j, \boldsymbol{x}_i); \cdots]$ for $\boldsymbol{l}_j$ at $\mathbf{X}_i$.

Point-to-Cylinder Residual The residual between $\boldsymbol{c}_j$ with the parameterization $\boldsymbol{\nu}_j = [\boldsymbol{\zeta}_j^{\boldsymbol{c}}; r_j]$ and its observation $\boldsymbol{p}_{ijk}^{\boldsymbol{c}} \in \mathbb{P}_{ij}^{\boldsymbol{c}}$ at $\mathbf{X}_i$ is defined as

$$\delta_{ijk}^{\boldsymbol{c}}(\boldsymbol{\nu}_j, \boldsymbol{x}_i) = ||\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j^{\boldsymbol{c}}, \boldsymbol{x}_i)||_2^2 - r_j^2, \tag{8}$$

where $\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j^{\boldsymbol{c}}, \boldsymbol{x}_i)$ is the point-to-line residual vector defined in (7). An alternative option to define the residual between $\boldsymbol{c}_j$ and $\boldsymbol{p}_{ijk}^{\boldsymbol{c}}$ is $\varepsilon_{ijk}^{\boldsymbol{c}} = ||\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j^{\boldsymbol{c}}, \boldsymbol{x}_i)||_2 - r_j$. Compared to $\varepsilon_{ijk}^{\boldsymbol{c}}$, $\delta_{ijk}^{\boldsymbol{c}}$ in (8) can benefit the computation. Specifically, we will show that, with some preprocessing, the computational complexity of minimizing the cost from (8) is independent of the number of points in $\mathbb{P}_{ij}^{\boldsymbol{c}}$. $\varepsilon_{ijk}^{\boldsymbol{c}}$ cannot lead to this performance. Stacking $\delta_{ijk}^{\boldsymbol{c}}(\boldsymbol{\nu}_j, \boldsymbol{x}_i)$ for the $N_{ij}^{\boldsymbol{c}}$ points in $\mathbb{P}_{ij}^{\boldsymbol{c}}$, we obtain the residual vector $\boldsymbol{\delta}_{ij}^{c}(\boldsymbol{\nu}_j, \boldsymbol{x}_i) = [\cdots; \delta_{ijk}^{\boldsymbol{c}}(\boldsymbol{\nu}_j, \boldsymbol{x}_i); \cdots]$ for $\boldsymbol{c}_j$ at $\mathbf{X}_i$.

Residual with Fixed Pose For local PLCA, poses out of the sliding window are fixed. Suppose that a landmark $\boldsymbol{m}_j$ ($\boldsymbol{m} \in \{\boldsymbol{\pi}, \boldsymbol{l}, \boldsymbol{c}\}$) is visible within the sliding window, and was also observed at a pose $\mathbf{X}_i$ that is out of the sliding window. Let us denote the estimated value of $\mathbf{X}_i$ and its parameterization as $\hat{\mathbf{X}}_i$ and $\hat{\boldsymbol{x}}_i$, respectively. We define the residual vector for points in $\mathbb{P}_{ij}^{\boldsymbol{m}}$ as

$$\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}(\boldsymbol{m}_j) = \boldsymbol{\delta}_{ij}^{\boldsymbol{m}}(\hat{\boldsymbol{x}}_i, \boldsymbol{m}_j). \tag{9}$$

In the following description, we remove the unknowns of the residuals for simplicity (*e.g.*, $\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}(\boldsymbol{\zeta}_j, \boldsymbol{x}_i) \to \boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}$).

V. System Overview

Our system includes a front-end and a back-end. The front-end detects planes, lines and cylinders, and establishes the local-to-global data association for real-time pose estimation. It also determines when a new keyframe should be created. The back-end includes local and global PLCA. They are implemented in two different threads. The local PLCA optimizes poses within a sliding window and the landmarks observed by these poses. We conduct local PLCA when a new keyframe is created. The global PLCA optimizes all the landmarks and all the poses except for the first pose. We trigger the global PLCA when landmarks are revisited. As planes, lines, and cylinders are unbounded objects, the robot does not need to return back to a previously visited place to trigger the global PLCA.

VI. Front-End

A. Landmark Detection

Split Scan Lines The LiDAR point cloud is comprised of multiple scan lines. Each scan line forms one row of the range image. We split a scan line into segments. Specifically, we adopt the method introduced in LOAM [3] to measure the smoothness of each point. Let us assume that $\boldsymbol{p}_{ij}$ is the jth point in the ith scan line, and $\mathbb{S}_{ij}$ contains the points around $\boldsymbol{p}_{ij}$ in the ith row. We compute

$$c = \frac{1}{|\mathbb{S}_{ij}| \cdot ||\boldsymbol{p}_{ij}||_2} || \sum_{\boldsymbol{p}_{ik} \in \mathbb{S}_{ij}} \boldsymbol{p}_{ij} - \boldsymbol{p}_{ik}||_2. \tag{10}$$

Let us denote as c_m the median of the c values of a scan line. The points whose c values are above the 95*th* percentile or larger than $5c_m$ are considered as on edges. If some edge points are close to each other, we only keep the one with the largest c. These points form a set $\mathbb{E}$, and split a scan line into segments. The points between the endpoints form a set $\mathbb{F}$. We adopt a region growing method to detect the landmarks.

Plane and Cylinder Detection The detection of planes and cylinders is similar to [31]. We start the detection from the first scan line. Given a segment $\mathbb{P}$ in the ith scan line, for each point in $\mathbb{P}$, we first find its k nearest points which belong to $\mathbb{F}$ but are not in the ith scan line. We then fit a plane to these points using the RANSAC algorithm. If the average point-to-plane distance of the inliers is larger than a threshold or the number of inliers is smaller than a threshold, we try to fit a cylinder to these points using the RANSAC algorithm. If the cylinder assumption results in a larger number of inliers and a smaller average distance, we treat these points belonging to a cylinder. We repeat this step until no points can be added into the cluster.

Line Detection We adopt a similar method for line detection. For a point $\boldsymbol{p}_0 \in \mathbb{E}$ that belongs to the ith scan line, we first find the nearest point $\boldsymbol{p}_1 \in \mathbb{E}$ in the $(i+1)$th scan line. A line $\boldsymbol{l}$ is computed using $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. We then find the nearest point $\boldsymbol{p}_2 \in \mathbb{E}$ of $\boldsymbol{p}_1$ in the $(i+2)$th scan line. If the distance between $\boldsymbol{p}_2$ and $\boldsymbol{l}$ is smaller than a distance, we fit a new line using $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. This process is repeated until no points can be added.

After the detection, we seek to merge the landmarks with similar parameters. We only detect planes, lines and cylinders in the whole scan at the first scan. For other keyframes, these landmarks are detected in the non-tracked points. Fig. 2 demonstrates the results of our algorithm. As illustrated in Fig. 2, the landmarks may be misidentified in this stage. We will correct the misidentifications in the back-end, as described in Section VII-C.

B. Forward ICP Flow

We adopt the forward ICP flow introduced in [2] to establish the local-to-global data association. Let us assume that $\mathbb{S}_{i+1}$ is the subsequent scan of $\mathbb{S}_i$. We first compute the set $\mathbb{E}$ and $\mathbb{F}$ for $\mathbb{S}_{i+1}$, as described in section VI-A. We build two KD trees for $\mathbb{E}$ and $\mathbb{F}$, respectively. Assume that $\mathbb{P}_{ij}^{\boldsymbol{m}}$ is the observation of $\boldsymbol{m}_j$ in $\mathbb{S}_i$. For each point in $\mathbb{P}_{ij}^{\boldsymbol{m}}$, we find n nearest neighbors in $\mathbb{S}_{i+1}$ ($n = 2$ in our experiments). For a plane or a cylinder, we simply combine these points. For a line, we only keep the one with the largest c. Assume that this yields a set $\mathbb{Q}_{i+1,j}^{\boldsymbol{m}}$. We adopt the RANSAC algorithm to find the inliers $\mathbb{P}_{i+1,j}^{\boldsymbol{m}}$, and then enlarge $\mathbb{P}_{i+1,j}^{\boldsymbol{m}}$ through the region growing method. If $|\mathbb{P}_{i+1,j}^{\boldsymbol{m}}|$ is too small, we ignore this match. Otherwise, this yields a local-to-global correspondence $\boldsymbol{m}_j \leftrightarrow \mathbb{P}_{i+1,j}^{\boldsymbol{m}}$. These correspondences will then be used in PLCR for pose estimation, which will be introduced in Section VIII. We can conduct the forward ICP flow for different landmarks in parallel.

C. Create Keyframe

Motivated by ORB-SLAM [32], we initially create many keyframes to reduce the drift of PLCR through local PLCA, and then sparsify them to make the global PLCA feasible. We create a new keyframe if one of the following conditions is met:

- More than 20% of the points in the current frame are not tracked.
- The rotation angle between the current frame and the last keyframe is larger than 5°.
- The distance between the current frame and the last keyframe is larger than a threshold t_1 ($t_1 = 0.2m$ for the indoor environment and $t_1 = 0.5m$ for the outdoor environment).

For a new keyframe, we detect planes, lines, and cylinders in the non-tracked points using the method introduced in Section VI-A. These new local landmarks are then tried to match with the global landmarks. For a new detection $\boldsymbol{m}$, we find the global landmark which has the smallest root mean squared error (RMSE). Let $d_{min}^{\boldsymbol{m}}$ denote this smallest RMSE. If $d_{min}^{\boldsymbol{m}} < \lambda$ ($\lambda = 5cm$ in the indoor environment and $\lambda = 15cm$ in the outdoor environment), we keep this correspondence. If $d_{min}^{\boldsymbol{m}} < 3\lambda$, we perform geometric consistency checks (GCC), otherwise we add a new global landmark.

Inspired by [2], we check the geometric consistency of new correspondences through the trial-and-error method. Specifically, we add the new correspondences into PLCR, local and global PLCA, subsequently. Each new correspondence is checked individually in PLCR, and new correspondences are checked together in local and global PLCA. If the RMSE of a new correspondence is smaller than λ at one of the above three checks, we retain this correspondence.

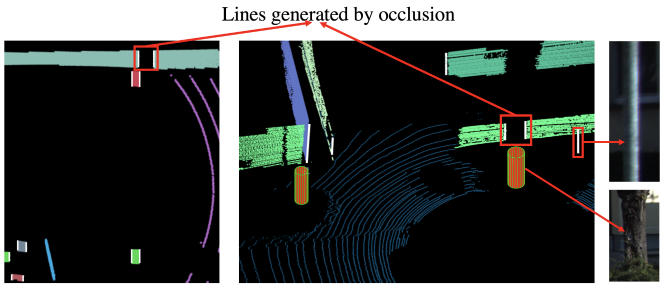

Fig. 2. Planes, lines and cylinders detected by our algorithm. Our algorithm can correctly identify most of the landmarks. But it also introduces several errors. The lines on the walls in the red rectangles are generated by the occlusion. Our algorithm also misidentifies a pole as a line. These errors will be corrected in the back-end.

When a new keyframe is created, the oldest keyframe is removed from the sliding window, if the window is full. We keep this keyframe for the global PLCA, if one of the following conditions is met

- This keyframe contains new detected landmarks.
- The rotation angle between this keyframe and the last retained keyframe is larger than 10°.
- The distance between this keyframe and the last retained keyframe is larger than t_2 ($t_2 = 2m$ for the indoor environment and $t_2 = 5m$ for the outdoor environment in our experiments).

If we keep this keyframe, we update $\mathbf{K}_j^{\boldsymbol{m}}$ according to (28) and store $\mathbf{H}_{ij}^{\boldsymbol{m}}$ that will be introduced in the next section. We will further sparsify the keyframes, if the number of keyframes is too large.

VII. Plane-Line-Cylinder Adjustment

A. Global Plane-Line-Cylinder Adjustment

Global PLCA is to jointly adjust the poses and the planes, lines, and cylinders to minimize the squared point-to-model distance. Formally, the global PLCA has the form

$$\min_{\substack{\{\boldsymbol{x}_i\},\{\boldsymbol{\eta}_j\}\\ \{\boldsymbol{\zeta}_j\},\{\boldsymbol{\nu}_j\}}} \sum_i \sum_{\boldsymbol{m}\in\{\boldsymbol{\pi},\boldsymbol{l},\boldsymbol{c}\}} \sum_{j\in\mathbb{O}_i^{\boldsymbol{m}}} ||\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}||_2^2, \tag{11}$$

where $\mathbb{O}_i^{\boldsymbol{m}}$ represents the indexes of planes, lines or cylinders observed at $\mathbf{X}_i$. The first pose is fixed during the optimization.

Let us assume that $\boldsymbol{\delta}$ and $\mathbf{J}_{\boldsymbol{\delta}}$ are the residual vector and the corresponding Jacobian matrix of (11), respectively. We adopt the LM algorithm to minimize (11). As shown in (3), instead of $\mathbf{J}_{\boldsymbol{\delta}}$ and $\boldsymbol{\delta}$, only $\mathbf{J}_{\boldsymbol{\delta}}^T\mathbf{J}_{\boldsymbol{\delta}}$ and $\mathbf{J}_{\boldsymbol{\delta}}^T\boldsymbol{\delta}$ are required by the LM algorithm. For PLCA, we can divide $\mathbf{J}_{\boldsymbol{\delta}}$ and $\boldsymbol{\delta}$ into blocks as

$$\boldsymbol{\delta} = [\cdots; \boldsymbol{\delta}_{ij}^{\boldsymbol{m}}; \cdots] \text{ and } \mathbf{J}_{\boldsymbol{\delta}} = [\cdots; \mathbf{J}_{ij}^{\boldsymbol{m}}; \cdots]. \tag{12}$$

Using (12), we get

$$\mathbf{J}_{\boldsymbol{\delta}}^T\boldsymbol{\delta} = \sum (\mathbf{J}_{ij}^{\boldsymbol{m}})^T \boldsymbol{\delta}_{ij}^{\boldsymbol{m}} \text{ and } \mathbf{J}_{\boldsymbol{\delta}}^T\mathbf{J}_{\boldsymbol{\delta}} = \sum (\mathbf{J}_{ij}^{\boldsymbol{m}})^T \mathbf{J}_{ij}^{\boldsymbol{m}}. \tag{13}$$

The following propositions provides the forms of $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\mathbf{J}_{ij}^{\boldsymbol{m}}$ for the jth plane, line and cylinder, respectively. The proofs of the propositions are in the **Appendix**.

Proposition 1: $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\mathbf{J}_{ij}^{\boldsymbol{\pi}}$ have the forms

$$(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}} = \left[\cdots \;\left(\mathbf{U}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\boldsymbol{\alpha}_{ij}\right)^T \;\cdots\; \left(\mathbf{V}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\boldsymbol{\alpha}_{ij}\right)^T \;\cdots\right]^T,$$
$$(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\mathbf{J}_{ij}^{\boldsymbol{\pi}} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \mathbf{U}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\mathbf{U}_{ij} & \cdots & \mathbf{U}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\mathbf{V}_{ij} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & \mathbf{V}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\mathbf{U}_{ij} & \cdots & \mathbf{V}_{ij}^T\mathbf{H}_{ij}^{\boldsymbol{\pi}}\mathbf{V}_{ij} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{14}$$

where $\boldsymbol{\alpha}_{ij} = \mathbf{X}_i^T\boldsymbol{\pi}_j$, $\mathbf{H}_{ij}^{\boldsymbol{\pi}} = (\mathbf{P}_{ij}^{\boldsymbol{\pi}})^T\mathbf{P}_{ij}^{\boldsymbol{\pi}}$, $\mathbf{U}_{ij} = \frac{\partial\boldsymbol{\alpha}_{ij}}{\partial\boldsymbol{\eta}_j}$ and $\mathbf{V}_{ij} = \frac{\partial\boldsymbol{\alpha}_{ij}}{\partial\boldsymbol{x}_i}$. Here $\mathbf{U}_{ij}$ and $\mathbf{V}_{ij}$ are computed according to (5).

Proposition 2: For $\boldsymbol{l}_j = [\boldsymbol{d}_j; \boldsymbol{m}_j]$, let us define $\mathbf{L}_j = \left[[\boldsymbol{d}_j]_\times, \boldsymbol{m}_j\right]$, and $\mathbf{L}_j^{[1]}$, $\mathbf{L}_j^{[2]}$, $\mathbf{L}_j^{[3]}$ are the three rows of $\mathbf{L}_j$. Then $\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}}$ in (7) can be written as:

$$\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}} = \left[(\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T\boldsymbol{\beta}_{ij}^{[1]};\; (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T\boldsymbol{\beta}_{ij}^{[2]};\; (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T\boldsymbol{\beta}_{ij}^{[3]}\right], \tag{15}$$

where $\boldsymbol{\beta}_{ij}^{[1]} = (\mathbf{L}_j^{[1]}\mathbf{X}_i)^T$, $\boldsymbol{\beta}_{ij}^{[2]} = (\mathbf{L}_j^{[2]}\mathbf{X}_i)^T$, and $\boldsymbol{\beta}_{ij}^{[3]} = (\mathbf{L}_j^{[3]}\mathbf{X}_i)^T$.

Proposition 3: $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{l}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T\mathbf{J}_{ij}^{\boldsymbol{l}}$ are similar to $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\mathbf{J}_{ij}^{\boldsymbol{\pi}}$ in (14), respectively. The two non-zero blocks of $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{l}}$ have the forms

$$\sum_{n=1}^{3}(\mathbf{A}_{ij}^{[n]})^T\mathbf{H}_{ij}^{\boldsymbol{l}}\boldsymbol{\beta}_{ij}^{[n]} \text{ and } \sum_{n=1}^{3}(\mathbf{B}_{ij}^{[n]})^T\mathbf{H}_{ij}^{\boldsymbol{l}}\boldsymbol{\beta}_{ij}^{[n]}, \tag{16}$$

where $\mathbf{H}_{ij}^{\boldsymbol{l}} = (\mathbf{P}_{ij}^{\boldsymbol{l}})^T\mathbf{P}_{ij}^{\boldsymbol{l}}$, $\mathbf{A}_{ij}^{[n]} = \frac{\partial\boldsymbol{\beta}_{ij}^n}{\partial\boldsymbol{\zeta}_j}$ and $\mathbf{B}_{ij}^{[n]} = \frac{\partial\boldsymbol{\beta}_{ij}^n}{\partial\boldsymbol{x}_i}$. The four non-zero blocks of $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T\mathbf{J}_{ij}^{\boldsymbol{l}}$ have the forms

$$\begin{gathered}\sum_{n=1}^{3}(\mathbf{A}_{ij}^{[n]})^T\mathbf{H}_{ij}^{l}\mathbf{A}_{ij}^{[n]},\; \sum_{n=1}^{3}(\mathbf{A}_{ij}^{[n]})^T\mathbf{H}_{ij}^{l}\mathbf{B}_{ij}^{[n]},\\ \sum_{n=1}^{3}(\mathbf{B}_{ij}^{[n]})^T\mathbf{H}_{ij}^{l}\mathbf{A}_{ij}^{[n]},\; \sum_{n=1}^{3}(\mathbf{B}_{ij}^{[n]})^T\mathbf{H}_{ij}^{l}\mathbf{B}_{ij}^{[n]}.\end{gathered} \tag{17}$$

Proposition 4: For $\boldsymbol{c}_j = [\boldsymbol{l}_j^{\boldsymbol{c}}; r_j]$ and its observation $\boldsymbol{p}_{ijk}^{\boldsymbol{c}} \in \mathbb{P}_{ij}^{\boldsymbol{c}}$ captured at $\mathbf{X}_i$, let us define $\tau_{ijk}^{[n]} = (\boldsymbol{\beta}_{ij}^{\boldsymbol{c}[n]})^T\boldsymbol{\Upsilon}_{ijk}\boldsymbol{\beta}_{ij}^{\boldsymbol{c}[n]}$ ($n = 1, 2, 3$) where $\boldsymbol{\Upsilon}_{ijk} = \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{c}}(\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{c}})^T \in \mathbb{R}^{4\times4}$ and $\boldsymbol{\beta}_{ij}^{\boldsymbol{c}[n]}$ is defined for $\boldsymbol{l}_j^{\boldsymbol{c}}$ according to (15). Let us expand the quadratic form $\tau_{ijk}^{[n]}$ as $\tau_{ijk}^{[n]} = \boldsymbol{q}_{ijk}^T\boldsymbol{\kappa}_{ij}^{[n]}$ according to (4), then $\boldsymbol{\delta}_{ijk}^{\boldsymbol{c}}$ in (8) has the form

$$\boldsymbol{\delta}_{ijk}^{\boldsymbol{c}} = \boldsymbol{g}_{ijk}^T\boldsymbol{\gamma}_{ij}, \text{ where } \boldsymbol{g}_{ijk} = \begin{bmatrix}\boldsymbol{q}_{ijk}\\ 1\end{bmatrix}, \boldsymbol{\gamma}_{ij} = \begin{bmatrix}\sum_{n=1}^{3}\boldsymbol{\kappa}_{ij}^{[n]}\\ -r_j^2\end{bmatrix}. \tag{18}$$

Proposition 5: $(\mathbf{J}_{ij}^{c})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{c}}$ and $(\mathbf{J}_{ij}^{c})^T\mathbf{J}_{ij}^{c}$ are similar to $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{\pi}})^T\mathbf{J}_{ij}^{\boldsymbol{\pi}}$ in (14), respectively. Using $\boldsymbol{g}_{ijk}$ in (18), we define $\mathbf{G}_{ij}^{c} = [\,\cdots; \boldsymbol{g}_{ijk}^T; \cdots]$ for points in $\mathbb{P}_{ij}^{c}$. The two non-zero blocks of $(\mathbf{J}_{ij}^{c})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{c}}$ have the forms

$$\mathbf{M}_{ij}^T\mathbf{H}_{ij}^{c}\boldsymbol{\gamma}_{ij} \text{ and } \mathbf{N}_{ij}^T\mathbf{H}_{ij}^{c}\boldsymbol{\gamma}_{ij}, \tag{19}$$

where $\mathbf{H}_{ij}^{c} = (\mathbf{G}_{ij}^{\boldsymbol{c}})^T\mathbf{G}_{ij}^{\boldsymbol{c}}$, $\mathbf{M}_{ij} = \frac{\partial\boldsymbol{\gamma}_{ij}}{\partial\boldsymbol{\nu}_j}$, $\mathbf{N}_{ij} = \frac{\partial\boldsymbol{\gamma}_{ij}}{\partial\boldsymbol{x}_i}$ and $\boldsymbol{\gamma}_{ij}$ is defined in (18). The four non-zero terms of $(\mathbf{J}_{ij}^{\boldsymbol{c}})^T\mathbf{J}_{ij}^{\boldsymbol{c}}$ have the forms

$$\mathbf{M}_{ij}^T\mathbf{H}_{ij}^{c}\mathbf{M}_{ij},\; \mathbf{M}_{ij}^T\mathbf{H}_{ij}^{c}\mathbf{N}_{ij},\; \mathbf{N}_{ij}^T\mathbf{H}_{ij}^{c}\mathbf{M}_{ij}, \text{ and } \mathbf{N}_{ij}^T\mathbf{H}_{ij}^{c}\mathbf{N}_{ij}. \tag{20}$$

Proposition 6: Given $\mathbf{H}_{ij}^{\boldsymbol{m}}$ ($\boldsymbol{m} \in \boldsymbol{\pi}, \boldsymbol{l}, \boldsymbol{c}$), the computational complexity of computing $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\mathbf{J}_{ij}^{\boldsymbol{m}}$ is $\frac{1}{N_{ij}^{\boldsymbol{m}}}$ relative to the traditional method (*i.e.*, computing $\mathbf{J}_{ij}^{\boldsymbol{m}}$ and $\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}$ individually and then multiplying them together).

According to the above propositions, given $\mathbf{H}_{ij}^{\boldsymbol{m}}$, $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{m}})^T\mathbf{J}_{ij}^{\boldsymbol{m}}$ are **independent of the number of points in** $\mathbb{P}_{ij}^{\boldsymbol{m}}$. $\mathbf{H}_{ij}^{\boldsymbol{m}}$ is a constant matrix. We only need to compute it once, and then reuse it during the iteration.

B. Local Plane-Line-Cylinder Adjustment

The local PLCA optimizes the poses within a sliding window and the landmarks observed by these poses. The poses out of the window are fixed. Formally, the cost function of local PLCA has the form

$$\min_{\substack{\mathbb{X}_w,\mathbb{P}_w\\ \mathbb{L}_w,\mathbb{C}_w}} \sum_{i\in\mathbb{W}}\sum_{\boldsymbol{m}\in\{\boldsymbol{\pi},\boldsymbol{l},\boldsymbol{c}\}}\sum_{j\in\mathbb{O}_i^{\boldsymbol{m}}}||\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}||_2^2 + \sum_{\boldsymbol{m}\in\{\boldsymbol{\pi},\boldsymbol{l},\boldsymbol{c}\}}\sum_{j\in\mathbb{V}^{\boldsymbol{m}}}\sum_{i\in\mathbb{F}_{\boldsymbol{m}_j}}||\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}||_2^2, \tag{21}$$

where $\mathbb{X}_w$ is the set of poses within the sliding window, $\mathbb{P}_w$, $\mathbb{L}_w$ and $\mathbb{C}_w$ are respectively the visible planes, lines, and cylinders at the poses in $\mathbb{X}_w$, $\mathbb{W}$ comprises the indices of the poses in $\mathbb{X}_w$, $\mathbb{V}^{\boldsymbol{m}}$ contains the indexes of visible planes, lines or cylinders, and $\mathbb{F}_{\boldsymbol{m}_j}$ includes the indexes of poses out of the sliding window that saw the landmark $\boldsymbol{m}_j$, $\mathbb{O}_i^{\boldsymbol{m}}$ is defined in (11) and $\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}$ is defined in (9).

To minimize (21), we can use the method introduced in section VII-A to handle $\boldsymbol{\delta}_{ij}^{\boldsymbol{m}}$ in (21). For $\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}$, we present a new method to reduce the computational cost of the LM algorithm. The observations of $\boldsymbol{m}_j$ at the poses out of the sliding window form a set

$$\mathbb{P}_j^{\boldsymbol{m}} = \{\mathbb{P}_{ij}^{\boldsymbol{m}} | i \in \mathbb{F}_{\boldsymbol{m}_j}\}. \tag{22}$$

Given $\hat{\mathbf{X}}_i$, we stack $\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}$ and $\mathbf{Q}_{ij}^{\boldsymbol{m}} = \mathbf{P}_{ij}^{\boldsymbol{m}}\hat{\mathbf{X}}_i^T$ for all points in $\mathbb{P}_j^{\boldsymbol{m}}$

$$\boldsymbol{\varepsilon}_j^{\boldsymbol{m}} = [\cdots; \boldsymbol{\varepsilon}_{ij}^{\boldsymbol{m}}; \cdots] \text{ and } \mathbf{Q}_j^{\boldsymbol{m}} = [\cdots; \mathbf{Q}_{ij}^{\boldsymbol{m}}; \cdots]. \tag{23}$$

Let $\mathbf{J}_j^{\boldsymbol{m}}$ denote the Jacobian matrix of $\boldsymbol{\varepsilon}_j^{\boldsymbol{m}}$. The following propositions provide the forms of $(\mathbf{J}_j^{\boldsymbol{m}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{m}}$ and $(\mathbf{J}_j^{\boldsymbol{m}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{m}}$.

Proposition 7: $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{\pi}}$ and $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\mathbf{J}_j^{\boldsymbol{\pi}}$ have the forms

$$\begin{gathered}(\mathbf{J}_j^{\boldsymbol{\pi}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{\pi}} = \left[\cdots;\; \boldsymbol{\Psi}_j^T\mathbf{K}_j^{\boldsymbol{\pi}}\boldsymbol{\pi}_j;\; \cdots\right],\\ (\mathbf{J}_j^{\boldsymbol{\pi}})^T\mathbf{J}_j^{\boldsymbol{\pi}} = \begin{bmatrix}\ddots & \vdots & \vdots\\ \cdots & \boldsymbol{\Psi}_j^T\mathbf{K}_j^{\boldsymbol{\pi}}\boldsymbol{\Psi}_j & \cdots\\ \vdots & \vdots & \ddots\end{bmatrix},\end{gathered} \tag{24}$$

where $\mathbf{K}_j^{\boldsymbol{\pi}} = (\mathbf{Q}_j^{\boldsymbol{\pi}})^T\mathbf{Q}_j^{\boldsymbol{\pi}}$ and $\boldsymbol{\Psi}_j = \frac{\partial\boldsymbol{\pi}_j}{\partial\boldsymbol{\eta}_j}$.

Proposition 8: $(\mathbf{J}_j^{\boldsymbol{l}})^T\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{l}}$ and $(\mathbf{J}_j^{\boldsymbol{l}})^T\mathbf{J}_j^{\boldsymbol{l}}$ are similar to $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{\pi}}$ and $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\mathbf{J}_j^{\boldsymbol{\pi}}$ in (24), respectively. The non-zero blocks of $(\mathbf{J}_j^{\boldsymbol{l}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{l}}$ and $(\mathbf{J}_j^{\boldsymbol{l}})^T\mathbf{J}_j^{\boldsymbol{l}}$ respectively have the forms

$$\sum_{n=1}^{3}(\boldsymbol{\Phi}_j^{[n]})^T\mathbf{K}_j^{\boldsymbol{l}}(\mathbf{L}_j^{[n]})^T \text{ and } \sum_{n=1}^{3}(\boldsymbol{\Phi}_j^{[n]})^T\mathbf{K}_j^{\boldsymbol{l}}\boldsymbol{\Phi}_j^{[n]}, \tag{25}$$

where $\mathbf{K}_j^{\boldsymbol{l}} = (\mathbf{Q}_j^{\boldsymbol{l}})^T\mathbf{Q}_j^{\boldsymbol{l}}$, $\boldsymbol{\Phi}_j^n = \frac{\partial(\mathbf{L}_j^{[n]})^T}{\partial\boldsymbol{\zeta}_j}$, and $\mathbf{L}_j^{[n]}$ is defined in Proposition 2.

Proposition 9: For $\boldsymbol{c}_j = [\boldsymbol{l}_j^{\boldsymbol{c}}; r_j]$ and its observation $\boldsymbol{p}_{ijk}^{\boldsymbol{c}} \in \mathbb{P}_{ij}^{\boldsymbol{c}}$ at a fixed pose $\hat{\mathbf{X}}_i$, let us define $\mu_{ijk}^{[n]} = (\mathbf{L}_j^{\boldsymbol{c}[n]})^T\boldsymbol{\Omega}_{ijk}\mathbf{L}_j^{\boldsymbol{c}[n]}$ ($n = 1, 2, 3$), where $\mathbf{L}_j^{\boldsymbol{c}[n]}$ is defined for $\boldsymbol{l}_j^{\boldsymbol{c}}$ according to Proposition 2, and $\boldsymbol{\Omega}_{ijk} = \hat{\mathbf{X}}_i\boldsymbol{\Upsilon}_{ijk}(\hat{\mathbf{X}}_i)^T$ where $\boldsymbol{\Upsilon}_{ijk}$ is defined in Proposition 4. The residual $\boldsymbol{\varepsilon}_{ijk}^{\boldsymbol{c}}$ for $\boldsymbol{p}_{ijk}^{\boldsymbol{c}}$ has the form

$$\boldsymbol{\varepsilon}_{ijk}^{\boldsymbol{c}} = \boldsymbol{k}_{ijk}^T\boldsymbol{\varphi}_{ij},\ \boldsymbol{k}_{ijk} = \begin{bmatrix}\boldsymbol{\omega}_{ijk}\\ 1\end{bmatrix},\ \boldsymbol{\varphi}_{ij} = \begin{bmatrix}\sum_{n=1}^{3}\boldsymbol{\rho}_{ij}^{[n]}\\ -r_j^2\end{bmatrix}, \quad (26)$$

where $\boldsymbol{\omega}_{ijk}$ and $\boldsymbol{\rho}_{ij}^{[n]}$ satisfy $\mu_{ijk}^{[n]} = \boldsymbol{\omega}_{ijk}^T\boldsymbol{\rho}_{ij}^{[n]}$ according to (4).

Proposition 10: $(\mathbf{J}_j^{\boldsymbol{c}})^T\boldsymbol{\varepsilon}_{ij}^{\boldsymbol{c}}$ and $(\mathbf{J}_j^{\boldsymbol{c}})^T\mathbf{J}_j^{\boldsymbol{c}}$ are similar to $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{\pi}}$ and $(\mathbf{J}_j^{\boldsymbol{\pi}})^T\mathbf{J}_j^{\boldsymbol{\pi}}$ in (24), respectively. Let us define $\mathbf{C}_j^{\boldsymbol{c}} = [\ \cdots; \boldsymbol{k}_{ijk}^T; \cdots]$ for the points in the set $\mathbb{P}_j^{\boldsymbol{c}}$ in (22), where $\boldsymbol{k}_{ijk}$ is defined in (26). The non-zero blocks of $(\mathbf{J}_j^{\boldsymbol{c}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{c}}$ and $(\mathbf{J}_j^{\boldsymbol{c}})^T\mathbf{J}_j^{\boldsymbol{c}}$ respectively have the forms

$$\boldsymbol{\Lambda}_j^T\mathbf{K}_j^c\boldsymbol{\varphi}_{ij} \text{ and } \boldsymbol{\Lambda}_j^T\mathbf{K}_j^c\boldsymbol{\Lambda}_j, \quad (27)$$

where $\mathbf{K}_j^{\boldsymbol{c}} = (\mathbf{C}_j^{\boldsymbol{c}})^T\mathbf{C}_j^{\boldsymbol{c}}$ and $\boldsymbol{\Lambda}_j = \frac{\partial\boldsymbol{\varphi}_{ij}}{\partial\boldsymbol{\nu}_j}$.

Incrementally Update $\mathbf{K}_j^{\boldsymbol{m}}$ From the above propositions, we know that $\mathbf{K}_j^{\boldsymbol{m}}$ is essential for computing $(\mathbf{J}_j^{\boldsymbol{m}})^T\boldsymbol{\varepsilon}_j^{\boldsymbol{m}}$ and $(\mathbf{J}_j^{\boldsymbol{m}})^T\mathbf{J}_j^{\boldsymbol{m}}$. As the number of points in $\mathbb{P}_j^{\boldsymbol{m}}$ increases, the computation for $\mathbf{K}_j^{\boldsymbol{m}}$ becomes expensive. Fortunately, we can calculate $\mathbf{K}_j^{\boldsymbol{m}}$ incrementally. Specifically, let us assume that the nth keyframe with the estimated pose $\hat{\mathbf{X}}_n$ is removed from the sliding window. The observation of $\boldsymbol{m}_j$ at $\hat{\mathbf{X}}_n$ is a set of points $\mathbb{P}_{nj}^{\boldsymbol{m}}$. For $\boldsymbol{m} \in \{\boldsymbol{\pi}, \boldsymbol{l}\}$, we first consider $\mathbf{Q}_j^{\boldsymbol{m}}$ defined in (23). Let us denote the old $\mathbf{Q}_j^{\boldsymbol{m}}$ as ${}^{old}\mathbf{Q}_j^{\boldsymbol{m}}$. According to (23), we compute $\mathbf{Q}_{nj}^{\boldsymbol{m}} = \mathbf{P}_{nj}^{\boldsymbol{m}}\hat{\mathbf{X}}_n^T$ for points in $\mathbb{P}_{nj}^{\boldsymbol{m}}$, and construct the new $\mathbf{Q}_j^{\boldsymbol{m}}$ by ${}^{new}\mathbf{Q}_j^{\boldsymbol{m}} = [{}^{old}\mathbf{Q}_j^{\boldsymbol{m}}; \mathbf{Q}_{nj}^{\boldsymbol{m}}]$. Then according to (24) and (25), we can compute ${}^{new}\mathbf{K}_j^{\boldsymbol{m}} = ({}^{new}\mathbf{Q}_j^{\boldsymbol{m}})^T\ {}^{new}\mathbf{Q}_j^{\boldsymbol{m}} = ({}^{old}\mathbf{Q}_j^{\boldsymbol{m}})^T\ {}^{old}\mathbf{Q}_j^{\boldsymbol{m}} + (\mathbf{Q}_{nj}^{\boldsymbol{m}})^T\mathbf{Q}_{nj}^{\boldsymbol{m}} = {}^{old}\mathbf{K}_j^{\boldsymbol{m}} + (\mathbf{Q}_{nj}^{\boldsymbol{m}})^T\mathbf{Q}_{nj}^{\boldsymbol{m}}$. For $\boldsymbol{m} = \boldsymbol{c}$, according to Proposition 10, we construct $\mathbf{C}_{nj}^{\boldsymbol{c}} = [\cdots; \boldsymbol{k}_{njk}; \cdots]$ for points in $\mathbb{P}_{nj}^{\boldsymbol{m}}$, and generate a new $\mathbf{C}_j^{\boldsymbol{c}}$ by ${}^{new}\mathbf{C}_j^{\boldsymbol{c}} = [{}^{old}\mathbf{C}_j^{\boldsymbol{c}}; \mathbf{C}_{nj}^{\boldsymbol{c}}]$. Then according to (27), we have ${}^{new}\mathbf{K}_j^{\boldsymbol{c}} = ({}^{new}\mathbf{C}_j^{\boldsymbol{c}})^T\ {}^{new}\mathbf{C}_j^{\boldsymbol{c}} = ({}^{old}\mathbf{C}_j^{\boldsymbol{c}})^T\ {}^{old}\mathbf{C}_j^{\boldsymbol{c}} + (\mathbf{C}_{nj}^{\boldsymbol{c}})^T\mathbf{C}_{nj}^{\boldsymbol{c}} = {}^{old}\mathbf{K}_j^{\boldsymbol{c}} + (\mathbf{C}_{nj}^{\boldsymbol{c}})^T\mathbf{C}_{nj}^{\boldsymbol{c}}$. In summary, we have

$${}^{new}\mathbf{K}_j^{\boldsymbol{m}} = {}^{old}\mathbf{K}_j^{\boldsymbol{m}} + \mathbf{K}_{nj}^{\boldsymbol{m}}, \quad (28)$$

where $\mathbf{K}_{nj}^{\boldsymbol{m}} = (\mathbf{Q}_{nj}^{\boldsymbol{m}})^T\mathbf{Q}_{nj}^{\boldsymbol{m}}$ if $\boldsymbol{m} \in \{\boldsymbol{\pi}, \boldsymbol{l}\}$, and $\mathbf{K}_{nj}^{\mathbf{c}} = (\mathbf{C}_{nj}^{\mathbf{c}})^T\mathbf{C}_{nj}^{\mathbf{c}}$, if $\boldsymbol{m} = \boldsymbol{c}$.

C. Correct the Detection Error

The detection may misidentify the landmarks. One general error is that cylinders are misrecognized as lines or planes, as shown in Fig. 1, and another one is the fake line due to occlusion, as shown in Fig. 2. We seek to correct these errors after more information is available. During the local and global PLCA, we check the RMSE of each landmark. If a RMSE is larger than a threshold, there may exist certain errors. We conduct different strategies for the three types of landmarks. For a cylinder, we directly remove this landmark. For a plane and a line, we fit a cylinder to these points. If this results in a smaller RMSE, we use the cylinder to model these points. Otherwise, we remove this landmark if it is identified as a line. Removing planes may result in insufficient constraints on a pose in some unstructured environment, as demonstrated in Fig. 3. Thus we do not remove planes. For a plane, assuming its RMSE $\bar{d}$ is larger than a threshold, we use $e^{-\bar{d}}$ to reduce its weight.

VIII. Plane-Line-Cylinder Registration

A. In-scan motion

The ith scan $\mathbb{S}_i$ is captured within the interval $(t_i^s, t_i^e]$. We adopt the pose at t_i^s as the pose of $\mathbb{S}_i$, and denote it as $\mathbf{X}_i$. Assume that the motion within $(t_i^s, t_i^e]$ results in a relative rigid body transformation $\mathbf{X}_{i,i+1}$ from $\mathbf{X}_{i+1}$ to $\mathbf{X}_i$. The relationship between $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ can be written as

$$\mathbf{X}_{i+1} = \mathbf{X}_i\mathbf{X}_{i,i+1}. \quad (29)$$

We denote the parameterization of $\mathbf{X}_{i,i+1}$ as $\boldsymbol{x}_{i,i+1} = [\boldsymbol{\omega}_{i,i+1}, \boldsymbol{t}_{i,i+1}]$, and the relative pose at $t \in (t_i^s, t_i^e]$ and its parameterization as $\mathbf{X}_{i,i+1}^t$ and $\boldsymbol{x}_{i,i+1}^t$, respectively. Let us define $s = \frac{t-t_i^s}{t_i^e-t_i^s}$. We adopt the linear interpolation to estimate $\boldsymbol{x}_{i,i+1}^t$ as done in [1]–[3], *i.e.*, $\boldsymbol{x}_{i,i+1}^t = s\boldsymbol{x}_{i,i+1}$. We use the plane-line-cylinder registration to compute $\boldsymbol{x}_{i,i+1}$ and undistort the in-scan motion as well.

B. Constraints

As the rotation component of the motion within $(t_i^s, t_i^e]$ is generally small, as done in [2], this paper adopts the first-order Taylor expansion of (1) to approximate the rotational part of $\mathbf{X}_{i,i+1}$, *i.e.*,

$$\mathbf{R}_{i,i+1}^t = \mathbf{I}_3 + s[\boldsymbol{\omega}_{i,i+1}]_\times. \quad (30)$$

Constraints from Planes Let us first consider the constraint on $\boldsymbol{x}_{i,i+1}$ from a plane. Suppose $\boldsymbol{\pi}_j$ is observed within $\mathbb{S}_i$, which generates a set of points $\mathbb{P}_{ij}^{\boldsymbol{\pi}}$. According to [2], $\boldsymbol{p}_{ijk}^{\boldsymbol{\pi}} \in \mathbb{P}_{ij}^{\boldsymbol{\pi}}$ captured at time $t_{ijk}^{\boldsymbol{\pi}}$ results in a linear constraint on $\boldsymbol{x}_{i,i+1}$. Specifically, using (29) and (6), we have

$$\epsilon_{ijk}^{\boldsymbol{\pi}}(\boldsymbol{x}_{i,i+1}) = \boldsymbol{\pi}_j^T\mathbf{X}_i\mathbf{X}_{i,i+1}^{t_{ijk}^{\boldsymbol{\pi}}}\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}} = \boldsymbol{\phi}_{ij}^{\boldsymbol{\pi}}\mathbf{X}_{i,i+1}^{t_{ijk}^{\boldsymbol{\pi}}}\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}}, \quad (31)$$

where $\boldsymbol{\phi}_{ij}^{\boldsymbol{\pi}} = \boldsymbol{\pi}_j^T\mathbf{X}_i$. Let us define $\boldsymbol{\phi}_{ij}^{\boldsymbol{\pi}} = \left[a_{ij}^{\boldsymbol{\pi}}, b_{ij}^{\boldsymbol{\pi}}, c_{ij}^{\boldsymbol{\pi}}, d_{ij}^{\boldsymbol{\pi}}\right]$, $\tilde{\boldsymbol{x}}_{i,i+1} = [\boldsymbol{x}_{i,i+1}; 1]$, and $\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}} = \left[x_{ijk}^{\boldsymbol{\pi}}, y_{ijk}^{\boldsymbol{\pi}}, z_{ijk}^{\boldsymbol{\pi}}, 1\right]^T$ and $s_{ijk}^{\boldsymbol{\pi}} = \frac{t_{ijk}^{\boldsymbol{\pi}}-t_i^s}{t_i^e-t_i^s}$. Substituting (30) into (31) and expanding it with the above notations, we obtain a linear constraint on $\boldsymbol{x}_{i,i+1}$:

$$\begin{aligned}
&\epsilon_{ijk}^{\boldsymbol{\pi}} \approx e_{ijk}^{\boldsymbol{\pi}} = \boldsymbol{w}_{ijk}^{\boldsymbol{\pi}}\tilde{\boldsymbol{x}}_{i,i+1}, \text{where } \boldsymbol{w}_{ijk}^{\boldsymbol{\pi}} = [\mathbf{a}_{ijk}^{\boldsymbol{\pi}}, b_{ijk}^{\boldsymbol{\pi}}],\\
&\mathbf{a}_{ijk}^{\boldsymbol{\pi}} = s_{ijk}^{\boldsymbol{\pi}} \cdot \left[c_{ij}^{\boldsymbol{\pi}}y_{ijk}^{\boldsymbol{\pi}} - b_{ij}^{\boldsymbol{\pi}}z_{ijk}^{\boldsymbol{\pi}},\ a_{ij}^{\boldsymbol{\pi}}z_{ijk}^{\boldsymbol{\pi}} - c_{ij}^{\boldsymbol{\pi}}x_{ijk}^{\boldsymbol{\pi}},\right.\\
&\qquad\left. b_{ij}^{\boldsymbol{\pi}}x_{ijk}^{\boldsymbol{\pi}} - a_{ij}^{\boldsymbol{\pi}}y_{ijk}^{\boldsymbol{\pi}},\ a_{ij}^{\boldsymbol{\pi}},\ b_{ij}^{\boldsymbol{\pi}},\ c_{ij}^{\boldsymbol{\pi}}\right], \quad (32)\\
&b_{ijk}^{\boldsymbol{\pi}} = -\left(a_{ij}^{\boldsymbol{\pi}}x_{ijk}^{\boldsymbol{\pi}} + b_{ij}^{\boldsymbol{\pi}}y_{ijk}^{\boldsymbol{\pi}} + c_{ij}^{\boldsymbol{\pi}}z_{ijk} + d_{ij}^{\boldsymbol{\pi}}\right).
\end{aligned}$$

Stacking all the constraints from the planes observed within $\mathbb{S}_i$, we obtain linear constraints on $\boldsymbol{x}_{i,i+1}$:

$$\boldsymbol{e}_i^{\boldsymbol{\pi}} = \mathbf{W}_i^{\boldsymbol{\pi}}\tilde{\boldsymbol{x}}_{i,i+1}. \quad (33)$$

Constraints from Lines Let us assume that a line $\boldsymbol{l}_j$ is observed within $\mathbb{S}_i$, and $\boldsymbol{p}_{ijk}^{\boldsymbol{l}}$ is point on $\boldsymbol{l}_j$ captured at time

Fig. 3. The environment of KITTI sequence 09 is not well structured. Our algorithm can still work on this sequence. But SuMa [9] slightly outperforms our algorithm on this dataset.

$t^{\boldsymbol{l}}_{ijk}$. Using (29) and (15), the nth ($n = 1, 2, 3$) element of the residual vector for $\boldsymbol{p}^{\boldsymbol{l}}_{ijk}$ has the form

$$\boldsymbol{\epsilon}^{\boldsymbol{l},[n]}_{ijk}(\boldsymbol{x}_{i,i+1}) = \mathbf{L}^{[n]}_j \mathbf{X}_i \mathbf{X}^{t^{\boldsymbol{l}}_{ijk}}_{i,i+1} \tilde{\boldsymbol{p}}^{\boldsymbol{l}}_{ijk} = \boldsymbol{\phi}^{\boldsymbol{l},[n]}_{ij} \mathbf{X}^{t^{\boldsymbol{l}}_{ijk}}_{i,i+1} \tilde{\boldsymbol{p}}^{\boldsymbol{l}}_{ijk}, \quad (34)$$

where $\boldsymbol{\phi}^{\boldsymbol{l},[n]}_{ij} = \mathbf{L}^{[n]}_j \mathbf{X}_i$. It is clear that (34) is similar to (31), which generates a linear constraint on $\boldsymbol{x}_{i,i+1}$:

$$\boldsymbol{e}^{\boldsymbol{l},[n]}_{ijk} = \mathbf{c}^{\boldsymbol{l},[n]}_{ijk} \tilde{\boldsymbol{x}}_{i,i+1}. \quad (35)$$

So stacking the constraints from the lines observed within $\mathbb{S}_i$ yields

$$\boldsymbol{e}^{\boldsymbol{l}}_i = \mathbf{W}^{\boldsymbol{l}}_i \tilde{\boldsymbol{x}}_{i,i+1}. \quad (36)$$

Constraints from Cylinders Let us assume a cylinder $\boldsymbol{c}_j = [\boldsymbol{l}^{\boldsymbol{c}}_j; r_j]$ is observed within $\mathbb{S}_i$. For a point $\boldsymbol{p}^{\boldsymbol{c}}_{ijk}$ captured from $\boldsymbol{c}_j$, we compute $\mathbf{c}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk}$ for the central line $\boldsymbol{l}^{c}_j$ of $\boldsymbol{c}_j$ according to (35). Let us define $\mathbf{E}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk} = (\mathbf{c}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk})^T \mathbf{c}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk}$. Using (8) and (35), we have

$$e^{\boldsymbol{c}}_{ijk} = \sum_{n=1}^{3} \left(\boldsymbol{e}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk} \right)^2 - r_j^2 = \underbrace{\tilde{\boldsymbol{x}}^T_{i,1+1} \left(\sum_{n=1}^{3} \mathbf{E}^{\boldsymbol{l}^{\boldsymbol{c}},[n]}_{ijk} \right) \tilde{\boldsymbol{x}}_{i,i+1}}_{\boldsymbol{q}^{\boldsymbol{c}}_{ijk} \boldsymbol{\chi}_{i,i+1}} - r_j^2$$

$$= [\boldsymbol{q}^{\boldsymbol{c}}_{ijk}, \ -r_j^2] \begin{bmatrix} \boldsymbol{\chi}_{i,i+1} \\ 1 \end{bmatrix} = \boldsymbol{y}^{\boldsymbol{c}}_{ijk} \tilde{\boldsymbol{\chi}}_{i,i+1}. \quad (37)$$

Here we apply (4) to yield $\boldsymbol{q}^{\boldsymbol{c}}_{ijk} \boldsymbol{\chi}_{i,i+1}$. Stacking the constraints from all the cylinders in $\mathbb{S}_k$, we get

$$\boldsymbol{e}^{\boldsymbol{c}}_i = \mathbf{Y}^{\boldsymbol{c}}_i \tilde{\boldsymbol{\chi}}_{i,i+1}. \quad (38)$$

The coefficients in (32), (35) and (37) can be computed in parallel.

C. Estimate $\boldsymbol{x}_{i,i+1}$

Planes, lines, and cylinders observed within $\mathbb{S}_i$ are used to compute $\boldsymbol{x}_{i,i+1}$. Let us define $\mathbf{O}_i = (\mathbf{W}^{\boldsymbol{\pi}}_i)^T \mathbf{W}^{\boldsymbol{\pi}}_i + (\mathbf{W}^{\boldsymbol{l}}_i)^T \mathbf{W}^{\boldsymbol{l}}_i$ and $\mathbf{Z}_i = (\mathbf{Y}^{\boldsymbol{c}}_i)^T \mathbf{Y}^{\boldsymbol{c}}_i$. Formally, the cost function for PLCR has the form

$$\min_{\boldsymbol{x}_{i,i+1}} \sum_{\boldsymbol{m}} ||\boldsymbol{e}^{\boldsymbol{m}}_j||_2^2 = \tilde{\boldsymbol{x}}^T_{i,i+1} \mathbf{O}_i \tilde{\boldsymbol{x}}_{i,i+1} + \tilde{\boldsymbol{\chi}}^T_{i,i+1} \mathbf{Z}_i \tilde{\boldsymbol{\chi}}_{i,i+1}. \quad (39)$$

The Hessian matrix and the gradient of (39) are easy to be computed. Thus we adopt the damped Newton's method to minimize (39). If only planes or lines are observed within $\mathbb{S}_i$, the cost (39) is a linear least-squares problem which has a closed-form solution.

Here we assume the rotation angle is small. If the rotation angle is large, equation (30) cannot well approximate the rotation matrix. We adopt the method introduced in [2] to solve this problem. Specifically, if the cost in (39) is large, we use the current $\mathbf{X}_{i,i+1}$ to transform $\boldsymbol{p}^{\boldsymbol{m}}_{ijk}$. Then we use the new $\boldsymbol{p}^{\boldsymbol{m}}_{ijk}$ to recompute a new $\mathbf{X}_{i,i+1}$. We repeat this step until it converges or the maximal number of iterations achieves. Let us denote as $\mathbf{X}^k_{i,i+1}$ the result from the kth step. Assuming m iterations were conducted, we have $\mathbf{X}_{i,i+1} = \prod_{k=1}^{m} \mathbf{X}^k_{i,i+1}$. During the iteration, we adopt the bisquare weight for robustness as done in [2], [3].

IX. Experiments

In this section, we evaluate the performance of our algorithm. All the experiments were conducted on a desktop with an Intel i9 CPU and 64G memory. Fig. 4 demonstrates the dense reconstruction from our algorithm.

A. Outdoor Environment

We first evaluate the performance of our algorithm in the outdoor environment. We use the KITTI dataset [33] for this purpose. We compare our algorithm with LiTAMIN2 [8], SuMa [9], LeGO-LOAM [10] and LOAM [3]. LiTAMIN2 and SuMa have the loop closure function, and LeGO-LOAM and LOAM do not. We also consider the performance of our algorithm without cylinder, referred to as **$\mathcal{PL}$-LiSLAM**. The KITTI dataset corrected the motion distortion of the LiDAR scan. Thus we set s in (30) to $s = 1$. We adopt the metrics introduced by the KITTI dataset to evaluate the accuracy of different algorithms. Table I lists the results. It is clear that our algorithm outperforms previous algorithms on most sequences. SuMa [9] provides a better result on sequence 09. This is because the sequence 09 is a country scene. Its environment is not well structured, as demonstrated in Fig. 3. $\mathcal{PLC}$-LiSLAM outperforms $\mathcal{PL}$-LiSLAM, as cylinders can more accurately model poles and trunks which widely exist in man-made environments.

B. Indoor Environment

To evaluate the performance of our algorithm in the indoor environment, we use the four indoor datasets in [2] and a new dataset that contains a large cylinder as shown in Fig. 4. The ground truths of these datasets are obtained from a NavVis M6 device. We compare our algorithm with BALM [5], π-LSAM [1], π-LiSLAM [2]. π-LSAM and π-LiSLAM only use planes as landmarks and are with the loop closure. BALM uses lines and planes as landmarks but without the loop closure. We use the absolute trajectory error (ATE) to evaluate the performance of different algorithms. Table II provides the results. It is clear that our algorithm outperforms other algorithms.

C. Runtime

We evaluate the runtime of our algorithm on the KITTI sequence 00 and the indoor dataset C. The KITTI dataset uses a velodyne LiDAR with 64 scan lines, and the indoor dataset uses a Velodyne LiDAR with 16 scan lines. Table III provides the results. As it takes *100 ms* for the Velodyne LiDAR to finish one scan, the front-end of our algorithm achieves real time. The runtime of the local PLCA on the

TABLE I
THE RESULTS OF DIFFERENT ALGORITHMS ON THE KITTI ODOMETRY DATASET. [33]

	Rotational error (degree per $100m$) / Translational error (%)											
Method	00	01	02	03	04	05	06	07	08	09	10	Avg
LiTAMIN2 [8]	0.28/0.70	0.46/2.10	0.32/0.98	0.48/0.96	0.52/1.05	0.25/0.45	0.34/0.59	0.32/0.44	0.29/0.95	0.40/0.69	0.47/0.80	0.38/0.88
SuMa [9]	0.22/0.64	0.47/1.77	0.41/1.23	0.46/0.57	0.27/0.39	**0.20**/0.42	0.28/0.51	0.52/0.65	0.35/1.15	**0.20/0.57**	0.27/0.69	0.33/0.78
LeGO-LOAM [10]	1.05/2.17	1.02/13.4	1.01/2.17	1.18/2.34	1.01/1.27	0.74/1.28	0.63/1.06	0.81/1.12	0.94/1.99	0.98/1.97	0.92/2.21	0.94/2.82
LOAM [3]	- /0.78	- /1.43	- /0.92	- /0.86	- /0.71	- /0.57	- /0.65	- /0.63	- /1.12	- /0.77	- /0.79	- / 0.84
$\mathcal{PL}$-LiSLAM	0.24/0.66	**0.34**/1.39	0.39/0.97	0.42/0.60	0.29/0.42	0.26/0.46	0.22/0.48	0.31/0.47	0.32/0.89	0.26/0.67	0.28/0.73	0.30/0.70
$\mathcal{PLC}$-LiSLAM	**0.20/0.61**	**0.34/1.38**	**0.31/0.82**	**0.33/0.52**	**0.21/0.36**	0.22/**0.38**	**0.16/0.42**	**0.25/0.41**	**0.23/0.73**	0.25/0.61	**0.25/0.67**	**0.25/0.63**

TABLE II
THE ATEs OF DIFFERENT ALGORITHMS ON THE FOUR INDOOR SEQUENCES FROM [2] AND A NEW COLLECTED INDOOR DATASET E THAT CONTAINS A LARGE CYLINDER.

	A	B	C	D	E
Length (m)	261.9	294.0	391.7	139.7	96.2
BALM [5]	0.34	0.21	-	0.23	0.29
π-LSAM [1]	0.082	0.16	0.13	0.11	0.18
π-LiSLAM [2]	0.039	0.042	0.033	0.048	0.073
$\mathcal{PLC}$-LiSLAM	**0.034**	**0.039**	**0.031**	**0.043**	**0.032**

TABLE III
RUNTIME (MS) OF DIFFERENT COMPONENTS OF OUR ALGORITHM. KEYFRAME CREATION IS ONLY CONDUCTED WHEN A NEW KEYFRAME IS REQUIRED. KC IS SHORT FOR KEYFRAME CREATION.

Dataset	Front-end			Back-end	
	Forward ICP Flow	PLCR	KC	Local PLCA	Global PLCA
KITTI 00	36.1 ± 6.2	32.3 ± 5.3	18.7 ± 3.1	8.1 ± 1.6	893 ± 433
Indoor C	10.3 ± 2.3	9.6 ± 1.8	5.2 ± 0.6	5.3 ± 0.9	308 ± 210

KITTI dataset is not much longer than the local PLCA on the indoor dataset. This is because the computational complexity of the LM algorithm is independent of the number of points on a landmark.

X. CONCLUSIONS

In this paper, we introduce a new LiDAR SLAM algorithm using planes, lines, and cylinders. We prove that, with some preprocessing, the minimization of local and global PLCA is independent of the number of points captured from planes, lines, and cylinders. In addition, we present an efficient solution to the PLCR problem. The detection may introduce errors, which are problematic for the method based on registration. Our algorithm checks and corrects the errors in the back-end. Experimental results show that our algorithm outperforms the state-of-the-art methods and achieves the real-time performance.

APPENDIX

A. Proof of Proposition 1

Let us define $\mathbf{X}_i^T \boldsymbol{\pi}_j = \boldsymbol{\alpha}_{ij}$. According to (6), we get

$$\delta_{ijk}^{\boldsymbol{\pi}} = \boldsymbol{\pi}_j^T \mathbf{X}_i \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}} = (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}})^T \mathbf{X}_i^T \boldsymbol{\pi}_j = (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{\pi}})^T \boldsymbol{\alpha}_{ij}. \tag{40}$$

Stacking (40) for the $N_{ij}^{\boldsymbol{\pi}}$ points in $\mathbb{P}_{ij}^{\boldsymbol{\pi}}$, we get the residual vector $\boldsymbol{\delta}_{ij}^{\pi}$ for $\boldsymbol{\pi}_j$ at $\mathbf{X}_i$

$$\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}} = \mathbf{P}_{ij}^{\boldsymbol{\pi}} \boldsymbol{\alpha}_{ij}, \tag{41}$$

where $\mathbf{P}_{ij}^{\boldsymbol{\pi}}$ is defined in (2). Here $\boldsymbol{\alpha}_{ij}$ is four-dimensional vector whose elements are functions of $\boldsymbol{\eta}_j$ and $\boldsymbol{x}_i$. According to (5), the Jacobian matrix of $\boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}$ has the form

$$\mathbf{J}_{ij}^{\boldsymbol{\pi}} = [\ \mathbf{0}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{U}_{ij}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{V}_{ij}\ \cdots \mathbf{0}\ ], \tag{42}$$

where $\mathbf{U}_{ij} = \frac{\partial \boldsymbol{\alpha}_{ij}}{\partial \boldsymbol{\eta}_j}$ and $\mathbf{V}_{ij} = \frac{\partial \boldsymbol{\alpha}_{ij}}{\partial \boldsymbol{x}_i}$. According to the block matrix multiplication rule, $(\mathbf{J}_{ij}^T)^T \boldsymbol{\delta}_{ij}^{\boldsymbol{\pi}}$ and $(\mathbf{J}_{ij}^T)^T \mathbf{J}_{ij}^{\boldsymbol{\pi}}$ have the form as (14).

B. Proof of Proposition 2

According to (7), we have

$$\begin{aligned}
\boldsymbol{\delta}_{ijk}^{\boldsymbol{l}} &= \boldsymbol{m}_j + [\boldsymbol{d}_j]_\times (\mathbf{R}_i \boldsymbol{p}_{ijk}^{\boldsymbol{l}} + \boldsymbol{t}_i) = \underbrace{[[\boldsymbol{d}_j]_\times, \boldsymbol{m}_j]}_{\mathbf{L}_j} \begin{bmatrix} \mathbf{R}_i \boldsymbol{p}_{ijk}^{\boldsymbol{l}} + \boldsymbol{t}_i \\ 1 \end{bmatrix} \\
&= \mathbf{L}_j \mathbf{X}_i \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}} = [\underbrace{\mathbf{L}_j^{[1]} \mathbf{X}_i}_{(\boldsymbol{\beta}_{ij}^{[1]})^T} \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}};\ \underbrace{\mathbf{L}_j^{[2]} \mathbf{X}_i}_{(\boldsymbol{\beta}_{ij}^{[2]})^T} \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}};\ \underbrace{\mathbf{L}_j^{[3]} \mathbf{X}_i}_{(\boldsymbol{\beta}_{ij}^{[3]})^T} \tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}}] \\
&= \left[(\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T \boldsymbol{\beta}_{ij}^{[1]};\ (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T \boldsymbol{\beta}_{ij}^{[2]};\ (\tilde{\boldsymbol{p}}_{ijk}^{\boldsymbol{l}})^T \boldsymbol{\beta}_{ij}^{[3]} \right].
\end{aligned} \tag{43}$$

C. Proof of Proposition 3

$\boldsymbol{\delta}_{ijk}^{\boldsymbol{\pi}}$ in (15) has three elements. Stacking them individually for the $N_{ij}^{\boldsymbol{l}}$ points in $\mathbb{P}_{ij}^{\boldsymbol{l}}$, we get

$$\boldsymbol{\delta}_{ij}^{\boldsymbol{l}} = [\ \mathbf{P}_{ij}^{\boldsymbol{l}} \boldsymbol{\beta}_{ij}^{[1]};\ \mathbf{P}_{ij}^{\boldsymbol{l}} \boldsymbol{\beta}_{ij}^{[2]};\ \mathbf{P}_{ij}^{\boldsymbol{l}} \boldsymbol{\beta}_{ij}^{[3]}\ ], \tag{44}$$

where $\mathbf{P}_{ij}^{\boldsymbol{l}}$ is defined in (2). According to the definition of $\boldsymbol{\beta}_{ij}^{[1]}$, $\boldsymbol{\beta}_{ij}^{[2]}$ and $\boldsymbol{\beta}_{ij}^{[3]}$ in (43), we know that their elements are functions of $\boldsymbol{\zeta}_j$ and $\boldsymbol{x}_i$. According to (5), the Jacobian matrix of $\boldsymbol{\delta}_{ij}^{\boldsymbol{l}}$ has the form

$$\mathbf{J}_{ij}^{\boldsymbol{l}} = \begin{bmatrix} \mathbf{0}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{A}_{ij}^{[1]}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{B}_{ij}^{[1]}\ \cdots \mathbf{0} \\ \mathbf{0}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{A}_{ij}^{[2]}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{B}_{ij}^{[2]}\ \cdots \mathbf{0} \\ \mathbf{0}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{A}_{ij}^{[3]}\ \cdots\ \mathbf{P}_{ij}^{\boldsymbol{\pi}} \mathbf{B}_{ij}^{[3]}\ \cdots \mathbf{0} \end{bmatrix}, \tag{45}$$

where $\mathbf{A}_{ij}^{[n]} = \frac{\partial \boldsymbol{\beta}_{ij}^n}{\partial \boldsymbol{\zeta}_j}$ and $\mathbf{B}_{ij}^{[n]} = \frac{\partial \boldsymbol{\beta}_{ij}^n}{\partial \boldsymbol{x}_i}$.

Substituting (44) and (45) into $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T \boldsymbol{\delta}_{ij}^{\boldsymbol{l}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T \mathbf{J}_{ij}^{\boldsymbol{l}}$, we can easily verify that the non-zero blocks of $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T \boldsymbol{\delta}_{ij}^{\boldsymbol{l}}$ and $(\mathbf{J}_{ij}^{\boldsymbol{l}})^T \mathbf{J}_{ij}^{\boldsymbol{l}}$ have the forms as (16) and (17), respectively.

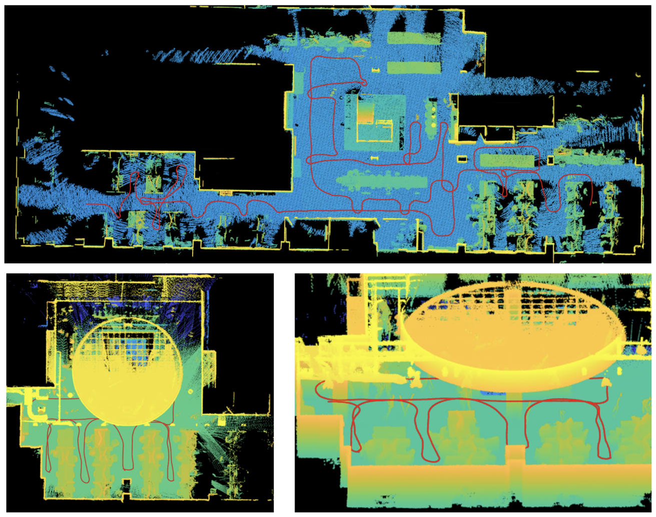
Fig. 4. The dense reconstruction result for the indoor dataset A (top) and E (bottom). The cylinder in dataset E is accurately reconstructed.

D. Proof of Proposition 4

Using (15), we can rewrite $\delta^{\boldsymbol{c}}_{ijk}$ in (8) as

$$\begin{aligned}\delta^{\boldsymbol{c}}_{ijk} &= \sum_{n=1}^{3}({\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]})^T\tilde{\boldsymbol{p}}^{\boldsymbol{c}}_{ijk}(\tilde{\boldsymbol{p}}^{\boldsymbol{c}}_{ijk})^T{\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]} - r_j^2\\ &= \sum_{n=1}^{3}\underbrace{({\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]})^T\boldsymbol{\Upsilon}_{ijk}{\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]}}_{\boldsymbol{q}^T_{ijk}\boldsymbol{\kappa}^{[n]}_{ij}} - r_j^2 = \boldsymbol{q}^T_{ijk}\sum_{n=1}^{3}\boldsymbol{\kappa}^{[n]}_{ij} - r_j^2\\ &= \underbrace{[\boldsymbol{q}^T_{ijk}, 1]}_{\boldsymbol{g}_{ijk}}\underbrace{\begin{bmatrix}\sum_{n=1}^{3}\boldsymbol{\kappa}^{[n]}_{ij}\\ -r_j^2\end{bmatrix}}_{\boldsymbol{\gamma}_{ij}} = \boldsymbol{g}_{ijk}\boldsymbol{\gamma}_{ij}\end{aligned} \tag{46}$$

where $\boldsymbol{\Upsilon}_{ijk}$ and ${\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]}$ are defined in Proposition 4 and the equation $(\boldsymbol{\beta}^{[n]}_{ij})^T\boldsymbol{\Upsilon}_{ijk}\boldsymbol{\beta}^{[n]}_{ij} = \boldsymbol{q}^T_{ijk}\boldsymbol{\kappa}^{[n]}_{ij}$ is based on (4).

E. Proof of Proposition 5

Stacking (46) for the $N^{\boldsymbol{c}}_{ij}$ points in $\mathbb{P}^{\boldsymbol{c}}_{ij}$ and using the definition $\mathbf{G}^{\boldsymbol{c}}_{ij} = [\cdots;\boldsymbol{g}_{ijk};\cdots]$, we get

$$\boldsymbol{\delta}^{\boldsymbol{c}}_{ij} = \mathbf{G}^{\boldsymbol{c}}_{ij}\boldsymbol{\gamma}_{ij}. \tag{47}$$

According to (5), the Jacobian matrix of $\boldsymbol{\delta}^{\boldsymbol{c}}_{ij}$ has the form

$$\mathbf{J}^{\boldsymbol{\pi}}_{ij} = [\ \mathbf{0}\ \cdots\ \mathbf{G}^{\boldsymbol{c}}_{ij}\mathbf{M}_{ij}\ \cdots\ \mathbf{G}^{\boldsymbol{c}}_{ij}\mathbf{N}_{ij}\ \cdots\mathbf{0}\ ], \tag{48}$$

where $\mathbf{M}_{ij} = \frac{\partial\boldsymbol{\gamma}_{ij}}{\partial\boldsymbol{\nu}_j}$ and $\mathbf{N}_{ij} = \frac{\partial\boldsymbol{\gamma}_{ij}}{\partial\boldsymbol{x}_i}$. According to the block matrix multiplication rule, the non-zero blocks of $(\mathbf{J}^T_{ij})^T\boldsymbol{\delta}^{\boldsymbol{\pi}}_{ij}$ and $(\mathbf{J}^T_{ij})^T\mathbf{J}^{\boldsymbol{\pi}}_{ij}$ have the forms as (19) and (20), respectively.

F. Proof of Proposition 6

According to Propositions 1, 3 and 5, given $\mathbf{H}^{\boldsymbol{m}}_{ij}$, the computational complexity of $(\mathbf{J}^{\boldsymbol{m}}_{ij})^T\boldsymbol{\delta}^{\boldsymbol{m}}_{ij}$ and $(\mathbf{J}^{\boldsymbol{m}}_{ij})^T\mathbf{J}^{\boldsymbol{m}}_{ij}$ is $O(1)$. Instead, the computational complexity of the traditional method is $O(N^{\boldsymbol{m}}_{ij})$. So the computational complexity of our solution is $\frac{1}{N^{\boldsymbol{m}}_{ij}}$ relative to the traditional method.

G. Proof of Proposition 7

Using (41), we can write $\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_{ij}$ in (9) as

$$\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_{ij} = \mathbf{P}^{\boldsymbol{\pi}}_{ij}\hat{\mathbf{X}}^T_i\boldsymbol{\pi}_{ij} = \mathbf{Q}^{\boldsymbol{\pi}}_{ij}\boldsymbol{\pi}_{ij}. \tag{49}$$

According to (23), $\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_j$ can be written as

$$\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_j = [\cdots;\mathbf{Q}^{\boldsymbol{\pi}}_{ij}\boldsymbol{\pi}_{ij};\cdots] = [\cdots;\mathbf{Q}^{\boldsymbol{\pi}}_{ij};\cdots]\boldsymbol{\pi}_{ij} = \mathbf{Q}^{\boldsymbol{c}}_j\boldsymbol{\pi}_j \tag{50}$$

According to (5), the Jacobian matrix of $\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_j$ has the form

$$\mathbf{J}^{\boldsymbol{\pi}}_j = [\ \mathbf{0},\ \cdots\ \mathbf{Q}^{\boldsymbol{\pi}}_j\boldsymbol{\Psi}_j,\ \cdots\ 0\ ], \tag{51}$$

where $\boldsymbol{\Psi}_j = \frac{\partial\boldsymbol{\pi}_j}{\partial\boldsymbol{\eta}_j}$. Using the rule of block matrix multiplication, we can easily verify that $(\mathbf{J}^{\boldsymbol{\pi}}_j)^T\boldsymbol{\varepsilon}^{\boldsymbol{\pi}}_j$ and $(\mathbf{J}^{\boldsymbol{\pi}}_j)^T\mathbf{J}^{\boldsymbol{\pi}}_j$ have the forms as (24).

H. Proof of Proposition 8

Using (44) and the definition of $\boldsymbol{\beta}^{[n]}_{ij}$ $(n = 1,2,3)$ in (43), we can write $\boldsymbol{\varepsilon}^{\boldsymbol{l}}_{ij}$ as

$$\begin{aligned}\boldsymbol{\varepsilon}^{\boldsymbol{l}}_{ij} &= [\ \mathbf{P}^{l}_{ij}\hat{\mathbf{X}}_i(\mathbf{L}^1_j)^T;\ \mathbf{P}^{l}_{ij}\hat{\mathbf{X}}_i(\mathbf{L}^2_j)^T;\ \mathbf{P}^{l}_{ij}\hat{\mathbf{X}}_i(\mathbf{L}^3_j)^T\ ]\\ &= [\ \mathbf{Q}^{\boldsymbol{l}}_{ij}(\mathbf{L}^1_j)^T;\ \mathbf{Q}^{\boldsymbol{l}}_{ij}(\mathbf{L}^2_j)^T;\ \mathbf{Q}^{\boldsymbol{l}}_{ij}(\mathbf{L}^3_j)^T\ ].\end{aligned} \tag{52}$$

Stacking the tree terms of (52) individually and then using the definition of $\mathbf{Q}^{\boldsymbol{l}}_j$ in (23), we get

$$\boldsymbol{\varepsilon}^{\boldsymbol{l}}_j = [\ \mathbf{Q}^{\boldsymbol{l}}_j(\mathbf{L}^1_j)^T;\ \mathbf{Q}^{\boldsymbol{l}}_j(\mathbf{L}^2_j)^T;\ \mathbf{Q}^{\boldsymbol{l}}_j(\mathbf{L}^3_j)^T\ ]. \tag{53}$$

Using (5), we can then get the Jacobian matrix of $\boldsymbol{\varepsilon}^{\boldsymbol{l}}_j$:

$$\mathbf{J}^{\boldsymbol{l}}_j = \begin{bmatrix}\mathbf{0}, & \cdots & \mathbf{Q}^{l}_j\boldsymbol{\Phi}^1_j, & \cdots & 0\\ \mathbf{0}, & \cdots & \mathbf{Q}^{l}_j\boldsymbol{\Phi}^2_j, & \cdots & 0\\ \mathbf{0}, & \cdots & \mathbf{Q}^{l}_j\boldsymbol{\Phi}^3_j, & \cdots & 0\end{bmatrix}, \tag{54}$$

where $\boldsymbol{\Phi}^n_j = \frac{\partial(\mathbf{L}^n_j)^T}{\partial\boldsymbol{\zeta}_j}$ $(n = 1,2,3)$. By substituting (53) and (54) into $(\mathbf{J}^{\boldsymbol{l}}_j)^T\boldsymbol{\varepsilon}^{\boldsymbol{l}}_j$ and $(\mathbf{J}^{\boldsymbol{l}}_j)^T\mathbf{J}^{\boldsymbol{l}}_j$, it is clear that their non-zero blocks have the forms as (25).

I. Proof of Proposition 9

Substituting ${\boldsymbol{\beta}^{\boldsymbol{c}}_{ij}}^{[n]} = ({\mathbf{L}^{\boldsymbol{c}}_j}^{[n]}\hat{\mathbf{X}}_i)^T$ into the second row of (46) and using ${\mathbf{L}^{\boldsymbol{c}}_j}^{[n]}\boldsymbol{\Omega}_{ijk}({\mathbf{L}^{\boldsymbol{c}}_j}^{[n]})^T = \boldsymbol{\omega}^T_{ijk}\boldsymbol{\rho}^{[n]}_{ij}$, we have

$$\begin{aligned}\varepsilon^{\boldsymbol{c}}_{ijk} &= \sum_{n=1}^{3}{\mathbf{L}^{\boldsymbol{c}}_j}^{[n]}\underbrace{\hat{\mathbf{X}}_i\boldsymbol{\Upsilon}_{ijk}\hat{\mathbf{X}}^T_i}_{\boldsymbol{\Omega}_{ijk}}({\mathbf{L}^{\boldsymbol{c}}_j}^{[n]})^T - r_j^2\\ &= \sum_{n=1}^{3}\underbrace{{\mathbf{L}^{\boldsymbol{c}}_j}^{[n]}\boldsymbol{\Omega}_{ijk}({\mathbf{L}^{\boldsymbol{c}}_j}^{[n]})^T}_{\boldsymbol{\omega}^T_{ijk}\boldsymbol{\rho}^{[n]}_{ij}} - r_j^2 = \boldsymbol{\omega}^T_{ijk}\sum_{n=1}^{3}\boldsymbol{\rho}^{[n]}_{ij} - r_j^2\\ &= \underbrace{[\boldsymbol{\omega}^T_{ijk}, 1]}_{\boldsymbol{k}^T_{ijk}}\underbrace{\begin{bmatrix}\sum_{n=1}^{3}\boldsymbol{\rho}^{[n]}_{ij}\\ -r_j^2\end{bmatrix}}_{\boldsymbol{\varphi}_{ij}} = \boldsymbol{k}^T_{ijk}\boldsymbol{\varphi}_{ij}.\end{aligned} \tag{55}$$

J. Proof of Proposition 10

Stacking $\varepsilon^{\boldsymbol{c}}_{ijk}$ for all points in $\mathbb{P}^{\boldsymbol{c}}_j$ defined in (22), we get

$$\boldsymbol{\varepsilon}^{\boldsymbol{c}}_j = [\cdots;\boldsymbol{k}_{ijk}\boldsymbol{\varphi}_{ij};\cdots] = [\cdots;\boldsymbol{k}_{ijk};\cdots]\boldsymbol{\varphi}_j = \mathbf{C}^{\boldsymbol{c}}_j\boldsymbol{\varphi}_{ij}. \tag{56}$$

According to (5), the Jacobian matrix of $\boldsymbol{\varepsilon}^{\boldsymbol{c}}_j$ can be written as

$$\mathbf{J}^{\boldsymbol{c}}_j = [\ \mathbf{0},\ \cdots\ \mathbf{C}^{\boldsymbol{c}}_j\boldsymbol{\Psi}_j,\ \cdots\ 0\ ]. \tag{57}$$

Using (56) and (57), we can easily verify that the non-zero blocks of $(\mathbf{J}^{\boldsymbol{c}}_j)^T\boldsymbol{\varepsilon}^{\boldsymbol{c}}_j$ and $(\mathbf{J}^{\boldsymbol{c}}_j)^T\mathbf{J}^{\boldsymbol{c}}_j$ have the forms as (27).

REFERENCES

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

[2] L. Zhou, D. Koppel, and M. Kaess, "LiDAR SLAM with plane adjustment for indoor environment," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7073–7080, 2021.

[3] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[4] F. Pomerleau, F. Colas, and R. Siegwart, *A Review of Point Cloud Registration Algorithms for Mobile Robotics*, 2015.

[5] Z. Liu and F. Zhang, "BALM: Bundle Adjustment for Lidar Mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.

[6] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[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] 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)*. IEEE, 2021, pp. 11 619–11 625.

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

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

[11] J. Lin and F. Zhang, "LOAM Livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3126–3131.

[12] X. Ji, L. Zuo, C. Zhang, and Y. Liu, "LLOAM: LiDAR odometry and mapping with loop-closure detection based correction," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2475–2480.

[13] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4390–4396.

[14] W. Xu and F. Zhang, "Fast-LIO: A Fast, Robust LiDAR-Inertial Odometry Oackage by Tightly-coupled Iterated Kalman Filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[15] S. Karam, V. Lehtola, and G. Vosselman, "Simple loop closing for continuous 6dof lidar&imu graph slam with planar features for indoor environments," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 181, pp. 413–426, 2021.

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

[17] G. Ferrer, "Eigen-factors: Plane estimation for multi-frame and time-continuous point cloud alignment," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 1278–1284.

[18] L. Zhou, D. Koppel, H. Ju, F. Steinbruecker, and M. Kaess, "An Efficient Planar Bundle Adjustment Algorithm," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2020, pp. 136–145.

[19] H. Huang, Y. Sun, J. Wu, J. Jiao, X. Hu, L. Zheng, L. Wang, and M. Liu, "On bundle adjustment for multiview point cloud registration," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8269–8276, 2021.

[20] B. Cao, R. C. Mendoza, A. Philipp, and D. Göhring, "LiDAR-based object-level SLAM for autonomous vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4397–4404.

[21] A. Kukko, R. Kaijaluoto, H. Kaartinen, V. V. Lehtola, A. Jaakkola, and J. Hyyppä, "Graph slam correction for single scanner mls forest data under boreal forest canopy," *ISPRS journal of photogrammetry and remote sensing*, vol. 132, pp. 199–209, 2017.

[22] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, X. Liu, R. A. F. Romero, and V. Kumar, "Sloam: Semantic lidar odometry and mapping for forest inventory," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 612–619, 2020.

[23] Y. Wang, Z. Sun, C.-Z. Xu, S. Sarma, J. Yang, and H. Kong, "LiDAR iris for loop-closure detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5769–5775.

[24] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *2020 International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[25] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent 3D LiDAR mapping with GPU-accelerated GICP matching cost factors," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8591–8598, 2021.

[26] J. Jiang, J. Wang, P. Wang, P. Bao, and Z. Chen, "LiPMatch: LiDAR Point Cloud Plane Based Loop-Closure," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6861–6868, 2020.

[27] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: Lidar-Inertial 3D plane SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 123–130.

[28] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] L. Zhang and R. Koch, "Structure and Motion from Line Correspondences: Representation, Projection, Initialization and Sparse Bundle Adjustment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 904–915, 2014.

[30] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[31] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast Registration Based on Noisy Planes with Unknown Correspondences for 3-D Mapping," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[32] 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.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.