A Novel Sparse Geometric 3-D LiDAR Odometry Approach

Shuang Liang, Zhiqiang Cao, *Senior Member, IEEE*, Peiyu Guan, Chengpeng Wang, Junzhi Yu, *Senior Member, IEEE*, and Shuo Wang

***Abstract*—Localization is a fundamental prerequisite, no matter whether a single robot or multirobot system, where light detection and ranging (LiDAR) odometry has attracted great interest with accurate depth information and robustness to illumination variations. In this article, a novel 3-D LiDAR odometry approach based on sparse geometric information is proposed. Different from geometric map-based 3-D LiDAR odometry methods with point features, we concern significant line and plane features based on eigenvalues of neighboring points. Furthermore, line-to-line and plane-to-plane associations instead of point-to-line and point-to-plane associations are adopted, and the problem of high computation complexity for scan-to-map matching module caused by point feature is solved. The proposed approach can not only guarantee the accuracy of pose estimation but also reduce computation complexity. Experiments on the public KITTI dataset and an outdoor scenario demonstrate the effectiveness of our approach in terms of accuracy and efficiency.**

***Index Terms*—Line and plane features, line-to-line and plane-to-plane associations, sparse geometric map, 3-D light detection and ranging (LiDAR) odometry.**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is an appealing research direction for service robots, which has received much attention [1]–[3]. It has a wide variety of potential applications such as autonomous driving, domestic services, warehouse logistics, etc. Visual SLAM [4], [5] and light detection and ranging (LiDAR) SLAM [6]–[9] are two of the mainstream branches. The former mainly adopts camera that is cheap and accessible with abundant environment information; however, it is susceptible to illumination variations with an unstable performance in weakly textured environments. The latter utilizes LiDAR to perceive the environment where accurate depth information with a large field of view can be obtained. Also, it is not affected by illumination variations and can work well even in the dark night, which makes LiDAR SLAM become an indispensable means for 24-h applications. A practical issue about LiDAR is that it is more expensive than cameras. However, researchers are delighted to see that the price of LiDAR is decreasing rapidly with its popularization in recent years. Driven by the great market demands, more and more affordable and high-quality LiDARs shall be continuously available.

Generally speaking, SLAM can be separated into two parts—front end and back end. In LiDAR SLAM, the front end is known as LiDAR odometry, which performs scan-to-scan and scan-to-map matching to estimate successive poses, whereas the back end typically detects loop closure and maintains a globally consistent map. In this article, we focus on 3-D LiDAR odometry, which is denoted as LiDAR odometry for simplicity.

LiDAR odometry can be categorized according to different representations of map, including point, voxel, segmentation, and geometric feature. The first type is the point-based LiDAR odometry [10], [11] where a set of discrete points are generated from historical point clouds. Deschaud proposed a SLAM with implicit moving least square (IMLS-SLAM), which is based on a sampling method similar to [12] and IMLS [10]. In addition to the coordinate information, other point properties are also exploited for better performance. Ceriani *et al*. [11] employed the normal information of points to perform point-to-plane iterative closest point (ICP) [13]. A drawback of the point-based LiDAR odometry is its high computational cost of scan matching due to low-level abstraction of the input point cloud. The second type of LiDAR odometry employs voxel grid map. In each voxel, points can be abstracted as distribution [14], [15], occupied probability [9], or statistics property [16]. Besides, octree-based map can be viewed as a special form of voxel grid map where the resolution is varied according to point cloud density [17], [18]. Fossel *et al*. [17] proposed a SLAM based on normal octree (NOctoSLAM) by replacing kd-tree with octree for point-plane ICP. Hornung *et al*. [18] used octree to estimate the occupancy probability and then constructed a compact 3-D environmental map. For voxel grid map, a problem is the choice of resolution. Low resolution leads to poor precision whereas high

Manuscript received November 11, 2019; revised February 21, 2020 and May 3, 2020; accepted May 5, 2020. Date of publication June 1, 2020; date of current version March 9, 2021. This work was supported in part by the National Natural Science Foundation of China under Grants 61633017, 61633020, and 61836015, in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS21, and in part by the Key Research and Development Program of Shandong Province under Grant 2017CXGC0925. *(Corresponding author: Zhiqiang Cao.)*

Shuang Liang, Zhiqiang Cao, Peiyu Guan, Chengpeng Wang, and Shuo Wang are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: liangshuang2017@ia.ac.cn; zhiqiang.cao@ia.ac.cn; guanpeiyu2017@ia.ac.cn; wangchengpeng2019@ia.ac.cn; shuo.wang@ia.ac.cn).

Junzhi Yu is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the State Key Laboratory for Turbulence and Complex System, Department of Mechanics and Engineering Science, Beijing Innovation Center for Engineering Science and Advanced Technology, College of Engineering, Peking University, Beijing 100871, China (e-mail: junzhi.yu@ia.ac.cn).

Digital Object Identifier 10.1109/JSYST.2020.2995727

resolution may result in inefficiency. Furthermore, many grids in sparse environments are empty, which wastes memory space and increases computational cost. To handle the empty space problem, the segmentation-based method is considered and it separates the point cloud into several clusters with a compact and efficient map representation [19], [20]. Das *et al*. [19] first separated the ground points and then clustered other points into several normal distributions. Afterward, normal distributions transform (NDT) [14] was utilized to acquire pose estimation by matching clusters between consecutive frames. Douillard *et al*. [20] clustered point cloud and searched data associations based on clusters; then, ICP was performed between associated clusters to estimate the pose. However, for some cases such as sparse point cloud, the results are not satisfactory. Different from the aforementioned three types with the consideration of all points, the geometric feature-based LiDAR odometry mainly concerns the points with noticeable geometric information. In real-world scenarios, the geometric features are ubiquitous and they can be easily extracted from the ground, wall, tree trunk, building, etc. Existing geometric features include the point feature [6], [21], [22] as well as the plane feature [23], [24]. Weingarten and Siegwart [23] proposed segmented planar model (SPmodel), which represents planar segment with polygon sets, and tracked the pose of robot by extended Kalman filter; however, only small-scale environment is allowed because of the quadratically scaling covariance matrix. In [24], only scan-to-scan matching is considered with a slow average processing time.

Among LiDAR odometric techniques, LiDAR odometry and mapping (LOAM) with geometric point feature [21] takes the first place on the KITTI benchmark dataset. Based on the point features extracted from sharp edges and flat planes, the pose is optimized by consecutive scan-to-scan and scan-to-map matching. A weakness of LOAM is its high computation complexity of scan-to-map matching with the running time of 1 Hz. Afterward, some efforts have been made to improve the performance of LOAM. Lightweight and ground-optimized LOAM (LeGO-LOAM) improved the extraction of point feature by removing unreliable points such as those on the tree leaves [6]. Also, LeGO-LOAM reduced the computational cost of scan-to-scan matching by, respectively, optimizing ground and nonground points. However, the computation complexity of LOAM is not relieved. Qin and Cao [22] proposed A-LOAM where the efficiency of scan-to-map matching is improved to some extent by downsizing point clouds with a smaller resolution. Neither LeGO-LOAM nor A-LOAM gets rid of the discrete and dense geometric point feature, which is the potential cause of high computation complexity of scan-to-map matching.

In this article, a sparse geometric LiDAR odometry (SGLO) is proposed to solve the drawback of LOAM [21], and the main contributions of this article are as follows. An efficient scan-to-map matching based on the sparse geometric line and plane features instead of dense point feature [6], [21], [22] is proposed, where only significant line and plane features are required based on eigenvalues of neighboring points for pose estimation. More importantly, in our scan-to-map matching, the line-to-line and plane-to-plane associations are introduced instead of point-to-line and point-to-plane associations [6], [21], [22], which not only significantly decrease the association amount but also improve the accuracy of data associations. As a result, a faster and accurate scan-to-map matching is achieved, which guarantees the whole quality of the proposed LiDAR odometry SGLO.

This article is organized as follows. Section II presents the problem statement. The proposed SGLO is given in Section III. The experimental results are presented in Section IV and Section V concludes this article.

II. Problem Statement

This article is motivated by the problem of simultaneous localization and mapping in unknown environments with only a LiDAR sensor. We label the LiDAR frame at t_k $(k = 1, 2, \ldots)$ as $\{L_k\}$ whose origin O_L is located at the center of the LiDAR. The *X*-axis and *Z*-axis of $\{L_k\}$ are corresponding to the forward direction of the LiDAR and the upward direction perpendicular to the LiDAR plane, respectively. Let X_k be the point cloud in the *k*th LiDAR scan during the time interval $[t_{k-1}, t_k]$. And point $x \in X_k$ is expressed by the coordinate $x_k^L \in \mathbb{R}^3$ in $\{L_k\}$. We denote with $\{W\}$ the world frame, which is coincidence with the frame $\{L_1\}$, as shown in Fig. 1.

For a consistent global map, line and plane features can provide a more compact and robust solution with less storage memory compared to point features. It shall be noted that the points in X_k are dense and cluttered, which is inefficient to directly extract the line and plane features. Consequently, the point screening of X_k is first executed and we concern the following two types—edge points and planar points [21]. The former refers to those whose surfaces have large roughness, whereas the latter focuses on the points corresponding to small roughness. The edge and planar point feature sets are denoted as $E_k \subseteq X_k$ and $F_k \subseteq X_k$, respectively. On this basis, the lines and planes can be fitted and one obtains the line feature set S_k and the plane feature set P_k. For a line feature $s \in S_k$, its representation in $\{L_k\}$ is given by a triple $s_k^L = (c_{s,k}^L, d_{s,k}^L, P_{s,k}^L)$, where $c_{s,k}^L$ is the center coordinate of the line, $d_{s,k}^L$ refers to its unit direction vector, and $P_{s,k}^L \subseteq E_k$ is the point set to fit this line. The representation of s in $\{W\}$ is $s_k^W = (c_{s,k}^W, d_{s,k}^W, P_{s,k}^W)$. Similarly, the representation of plane feature $p \in P_k$ in $\{L_k\}$ is described as $p_k^L = (c_{p,k}^L, n_{p,k}^L, P_{p,k}^L)$, where $c_{p,k}^L$ and $n_{p,k}^L$ are the center and unit normal vector of the plane, respectively, and $P_{p,k}^L \subseteq F_k$ refers to the point set to fit this plane. Correspondingly, the plane feature p in $\{W\}$ is denoted as $p_k^W = (c_{p,k}^W, n_{p,k}^W, P_{p,k}^W)$.

The sparse geometric feature sets S_k and P_k pave the foundation of global sparse geometric map $M(k)$, which is obtained by merging line and plane features from first to *k*th LiDAR scans. The sets of the line and plane features in $M(k)$ are denoted as $S_M(k)$ and $P_M(k)$, respectively, where line features $s_M \in S_M(k)$ and plane features $p_M \in P_M(k)$ in $\{W\}$ are parameterized as $s^W = (c_s^W, d_s^W)$ and $p^W = (c_p^W, n_p^W)$, respectively. c_s^W and d_s^W refer to the center and unit direction vector of s_M whereas c_p^W and n_p^W denote the center and unit normal vector of p_M. Fig. 1 gives a schematic description with the line features $s_{M_1} - s_{M_3}$ and plane features $p_{M_1} - p_{M_5}$ stored in the global sparse map for objects $\mathrm{obj}_1 - \mathrm{obj}_3$.

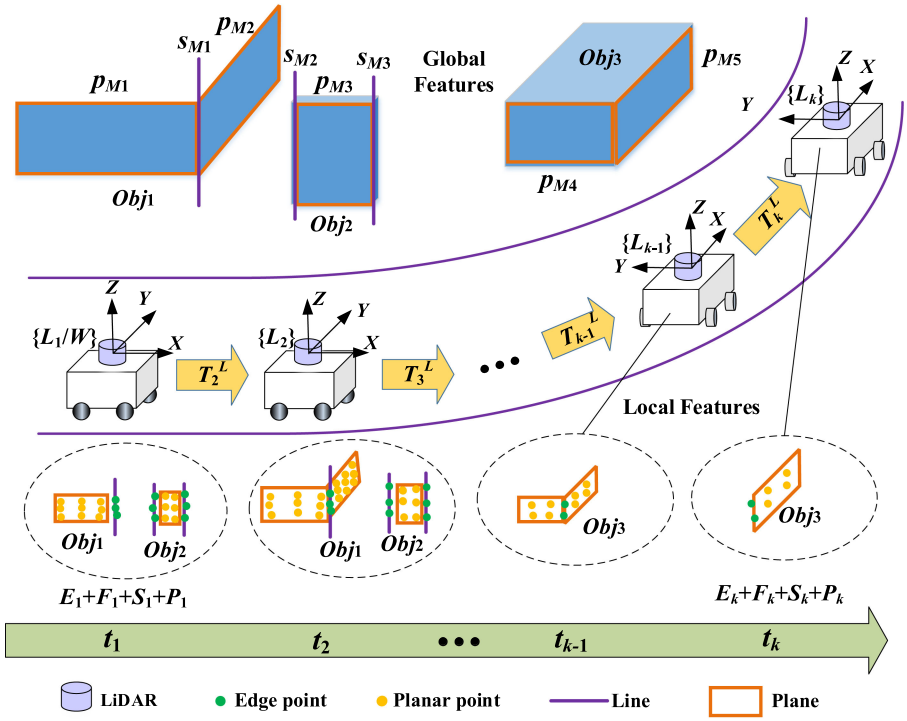

Fig. 1. Sparse geometric LiDAR odometry.

TABLE I
SYMBOLS AND NOTATIONS

Symbol	Notation
$\|\cdot\|$, $\|\cdot\|_2$	Euclidean norm operator
$\triangleq$	Equal by definition
$a \times b$	Cross product, where $a, b \in \mathbb{R}^n$, $n = 1,2,3,\cdots$
$a \cdot b$	Dot product, where $a, b \in \mathbb{R}^n$, $n = 1,2,3,\cdots$
$(\cdot)^T$	Transpose operator
$det(\cdot)$	Determinant of a matrix
$[\cdot]_\times$	Skew symmetric matrix operator
$\exp(\cdot)$	Exponential map
$\mathrm{Exp}(\cdot)$	Capital exponential map
A × B	Cartesian product, where A and B are two sets
$\oplus$	Plus operator between 3D vector space and $SO(3)$
$\boxplus$	Plus operator between 6D vector space and $\mathcal{M}$
$\partial(\cdot)$	Partial derivative
$\nabla(\cdot)$	Gradient operator
$(\cdot)^{-1}$	Inverse operator
$\lfloor\cdot\rfloor$	Floor function
$\forall(\cdot)$	Any element
$\|\cdot\|$	Number of elements in the set

For LiDAR odometry, a crucial step is to obtain the pose transformation among frames. In this article, T_k^L is denoted with the transformation from frame $\{L_{k-1}\}$ to frame $\{L_k\}$, where $k \geq 2$, and the pose transformation from $\{W\}$ to $\{L_k\}$ is denoted as T_k^W. Specifically, $T_1^W = T_1^L = I$. Symbols and notations used in this article are shown in Table I.

III. Sparse Geometric LiDAR Odometry

In this section, an SGLO is given. The whole process is described as follows. The point feature sets E_k and F_k are first extracted from the point cloud X_k. Afterward, scan-to-scan matching is performed to optimize the transformation T_k^L according to E_{k-1}, F_{k-1}, E_k, F_k, and T_{k-1}^L. Next step is the scan-to-map matching. We extract geometric feature sets S_k, P_k from E_k, F_k. With $T_1^L,\ldots, T_k^L$, T_l^W, S_k, P_k, and $M(l)$, the transformation T_k^W is optimized and then, the global sparse geometric map $M(k)$ is updated, where l is the index of the latest frame that involves in the scan-to-map matching and $1 \leq l < k$. Finally, the optimized scan-to-scan and scan-to-map poses are integrated to achieve a fast and accurate localization.

A. Point Feature Extraction

In [6], [21], and [25], edge and planar point features are adopted. Owing to the fact that there is no edge point feature in the ground plane, we first extract the ground plane in avoidance of wrong extraction of edge point feature. Zermas *et al.* [26] presented a ground plane fitting method based on region growing. Only a single ground plane is obtained and it mainly works on the flat ground. Considering the unevenness of the ground plane, we design a grid-based method that creates a 2-D grid map in *X*–*Y* plane, and for each grid, a corresponding plane is fitted using the method in [26]. With the constraint of the plane's normal vector, the ground planes are acquired. On this basis, edge and planar point features are acquired using [21], which constitute the sets E_k and F_k, respectively.

B. Scan-to-Scan Matching

A general solution for scan-to-scan matching is based on ICP framework [27], [28], which iteratively executes the data

association and pose optimization. In this section, the point feature sets are used for data associations. Based on the results of point–line and point–plane associations, the optimization problem is formulated, which is then solved by Levenberg–Marquard (LM) method based on composite manifold.

The data associations involve the respective correlations of edge and planar point features. Take planar point feature as an example. With F_{k-1} and F_k, $k \geq 2$, we first utilize F_{k-1} to construct a 3-D kd-tree before ICP iterations. For the ith ICP iteration ($1 \leq i \leq I_s$), where $I_s = 5$ is the maximum number of ICP iteration, we project each point $f \in F_k$ from frame $\{L_k\}$ to $\{L_{k-1}\}$ via function $\prod : \mathbb{R}^3 \to \mathbb{R}^3$

$$\bar{f}_k^L = \prod \left(f_k^L, {}^{(i)}T_k^L \right) \triangleq \left[I_{3\times 3} \; 0_{3\times 1} \right] {}^{(i)}T_k^L \begin{bmatrix} f_k^L \\ 1 \end{bmatrix} \quad (1)$$

where f_k^L and $\bar{f}_k^L$ are the coordinates of f in $\{L_k\}$ and $\{L_{k-1}\}$, respectively, ${}^{(i)}T_k^L$ is the pose in the optimization of ith ICP iteration, and ${}^{(1)}T_k^L$ is initialized by T_{k-1}^L. $I_{3\times 3}$ is an identity matrix of 3×3 size, and $0_{3\times 1}$ is a zero matrix with the size of 3×1. Then, the nearest neighboring searching strategy via 3-D kd-tree in [21] is employed to achieve point–plane associations where a corresponding plane for f in $(k-1)$th frame is found whose center and normal vector are denoted as $c_{f,k-1}^L$ and $n_{f,k-1}^L$, respectively. Similarly, for $e \in E_k$, point–line associations are executed, and a corresponding line in $(k-1)$th frame with its center $c_{e,k-1}^L$ and direction vector $d_{e,k-1}^L$ is obtained.

The optimization problem is thus formulated by minimizing the half of squared sum of distance vectors, which is as follows:

$$\begin{aligned} {}^{(i)}T_k^{L*} = \underset{{}^{(i)}T_k^L}{\operatorname{argmin}} \frac{1}{2} \sum_{e\in E_k} \rho \left(\|\mathrm{dis}_e\|_2^2 \right) \\ + \frac{1}{2} \sum_{f\in F_k} \rho(\|\mathrm{dis}_f\|_2^2) \end{aligned} \quad (2)$$

where $\rho(\cdot)$ is the Huber loss function; the distance vectors dis_e and dis_f refer to the distances between point and corresponding line or plane, which are given by

$$\begin{cases} \mathrm{dis}_e \triangleq \left(\bar{e}_k^L - c_{e,k-1}^L \right) \times d_{e,k-1}^L & e \in E_k \\ \mathrm{dis}_f \triangleq \left(\bar{f}_k^L - c_{f,k-1}^L \right) \cdot n_{f,k-1}^L & f \in F_k \end{cases}. \quad (3)$$

To solve (2), we employ a composite manifold-based LM method, which avoids the singularity problem caused by Euler angle-based solution [6]. Some descriptions of the manifold are given first.

Lie group $SO(3)$ is used to represent the rotation in 3-D space, which is described as follows:

$$SO(3) = \left\{ R \,|R \in \mathbb{R}^{3\times 3},\; R^T R = I,\; \det(R) = 1 \right\}. \quad (4)$$

It is obvious that $SO(3)$ is not a vector space, however, its corresponding Lie algebra $\mathfrak{so}(3)$ is a vector space [29], which is given by

$$\mathfrak{so}(3) = \left\{ [\phi]_\times | \phi \in \mathbb{R}^3 \right\} \quad (5)$$

where $[\cdot]_\times$ is skew symmetric matrix operator. The relationship between $SO(3)$ and $\mathfrak{so}(3)$ is described via exponential map [30]

$$\exp : \mathfrak{so}(3) \to SO(3); [\phi]_\times \to R = \exp\left([\phi]_\times \right). \quad (6)$$

Also, we employ the mapping from $\phi \in \mathbb{R}^3$ to $\mathrm{R} \in SO(3)$ [29]

$$\mathrm{Exp} : \mathbb{R}^3 \to SO(3); \phi \to R = \mathrm{Exp}(\phi) = \exp\left([\phi]_\times \right). \quad (7)$$

In Lie group $SO(3)$, there are multiple definitions of plus, and we utilize left plus $\oplus : \mathbb{R}^3 \times SO(3) \to SO(3)$, which is expressed as follows [30]:

$$R_2 = \phi \oplus R_1 \triangleq \mathrm{Exp}(\phi) R_1 \quad (8)$$

where $R_1, R_2 \in SO(3)$ and $\phi \in \mathbb{R}^3$.

Pose transformation is composed of not only rotation but also translation. Typically, $SE(3)$ [30] is commonly used; however, the property of its Lie algebra $\mathfrak{se}(3)$ is complex, which leads to a complicated update process of pose transformation. To solve this problem, composite manifold $\mathcal{M}$ and a plus symbol $\boxplus : \mathbb{R}^6 \times \mathcal{M} \to \mathcal{M}$ are considered in this article [29]

$$\mathcal{M} \triangleq \left\{ T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \mid R \in SO(3), t \in \mathbb{R}^3 \right\} \quad (9)$$

$$T_2 = \xi \boxplus T_1 \triangleq \begin{bmatrix} \phi \oplus R_1 & p + t_1 \\ 0 & 1 \end{bmatrix} \quad (10)$$

where $T_1, T_2 \in \mathcal{M}$, $T_1 = \left[\begin{smallmatrix} R_1 & t_1 \\ 0 & 1 \end{smallmatrix} \right]$, $T_2 = \left[\begin{smallmatrix} R_2 & t_2 \\ 0 & 1 \end{smallmatrix} \right]$, $\xi = \left[\begin{smallmatrix} \phi \\ p \end{smallmatrix} \right] \in \mathbb{R}^6$ describes the tangent vector of composite manifold $\mathcal{M}$ with rotation Lie algebra vector $\phi \in \mathbb{R}^3$ and translation vector $p \in \mathbb{R}^3$.

Then, the derivative for function $g : \mathcal{M} \to \mathbb{R}^m$ is given by

$$\frac{\partial g}{\partial T} \triangleq \lim_{\delta\xi \to 0} \frac{g(\delta\xi \boxplus T) - g(T)}{\delta\xi} \in \mathbb{R}^{m\times 6}. \quad (11)$$

With the above-mentioned definitions, (2) is solved by employing LM algorithm under the framework of composite manifold. The core is to calculate residual vector d and Jacobian matrix J, which are stacked by the distance vectors dis_e, dis_f and gradient vectors$\nabla \mathrm{dis}_e$, $\nabla \mathrm{dis}_f$ of $e \in E_k$ and $f \in F_k$

$$d = \left[\cdots \; \mathrm{dis}_e^T \; \cdots \; \cdots \; \mathrm{dis}_f^T \; \cdots \right]^T \quad (12)$$

$$J = \left[\cdots \; \nabla \mathrm{dis}_e^T \; \cdots \; \cdots \; \nabla \mathrm{dis}_f^T \; \cdots \right]^T \quad (13)$$

where $\nabla \mathrm{dis}_e \triangleq \frac{\partial \mathrm{dis}_e}{\partial {}^{(i)}T_k^L}$, $\nabla \mathrm{dis}_f \triangleq \frac{\partial \mathrm{dis}_f}{\partial {}^{(i)}T_k^L}$, and $(\cdot)^T$ is the transpose operator. With (1), (3), and (11), we have

$$\begin{cases} \nabla \mathrm{dis}_e = -\left[d_{e,k-1}^L \right]_\times \left[\; -\left[{}^{(i)}R_k^L e_k^L \right]_\times \quad I_{3\times 3} \; \right] & e \in E_k \\ \nabla dis_f = {n_{f,k-1}^L}^T \left[\; -\left[{}^{(i)}R_k^L f_k^L \right]_\times \quad I_{3\times 3} \; \right] & f \in F_k \end{cases} \quad (14)$$

where ${}^{(i)}T_k^L = \left[\begin{smallmatrix} {}^{(i)}R_k^L & {}^{(i)}t_k^L \\ 0 & 1 \end{smallmatrix} \right] \in \mathcal{M}$. Because of the limitation of article length, we skip the derivation process of (14). On this basis, optimized increment $\delta\xi$ of pose transformation can be determined based on LM algorithm [21] as follows:

$$\left(J^T J + \lambda I \right) \delta\xi = -J^T d \quad (15)$$

and the pose ${}^{(i)}T_k^L$ can be updated as follows:

$${}^{(i)}T_k^L \leftarrow \delta\xi \boxplus {}^{(i)}T_k^L. \quad (16)$$

Repeat (12)–(16) until LM algorithm converges or the iteration number reaches its maximum value O_s, where $O_s = 5$.

Algorithm 1: Scan-to-Scan Matching.

Input: E_{k-1}, F_{k-1}, E_k, F_k, T_{k-1}^L
Output: T_k^L
${}^{(1)}T_k^L \leftarrow T_{k-1}^L$;
Employ E_{k-1} and F_{k-1} to construct two kd-trees;
for $i \in [1,\ I_s]$ **do** //outer loop for data association
  Find $(c_{f,k-1}^L, n_{f,k-1}^L)$ or $(c_{e,k-1}^L, d_{e,k-1}^L)$ for each $e \in E_k$ or $f \in F_k$ via (1) and kd-trees;
  for $o \in [1,\ O_s]$ **do** //inner loop for optimization
    Calculate d and J according to (12) and (13);
    Update ${}^{(i)}T_k^L$ based on (15) and (16);
  end for
  ${}^{(i+1)}T_k^L \leftarrow {}^{(i)}T_k^L$;
end for
$T_k^L \leftarrow {}^{(I_s+1)}T_k^L$;
return

Afterward, the result of ith ICP iteration is assigned as the initial value of the next ICP iteration, i.e., ${}^{(i+1)}T_k^L = {}^{(i)}T_k^L$. After all ICP iterations finish, we obtain the optimized pose transformation $T_k^L = {}^{(I_s+1)}T_k^L$. The detailed pipeline is given in Algorithm 1.

C. Scan-to-Map Matching

Compared with scan-to-scan matching, scan-to-map matching can produce more accurate pose estimation results. In this section, global pose T_k^W is estimated and map $M(k)$ is updated. Specifically, line feature set S_k and plane feature set P_k are extracted from point feature sets E_k and F_k. Then, T_k^W is initialized and surrounding maps $\mathcal{S}_M(l)$ and $\mathcal{P}_M(l)$ are acquired. On this basis, T_k^W is estimated. Also, $M(k)$ is updated.

1) Extracting Lines and Planes: With E_k and F_k, line and plane features are extracted, respectively. To fit the planes, a 3-D kd-tree is built using all points in F_k. For $f \in F_k$ that is not involved in fitting, its surrounding points are acquired by the kd-tree, and we label the mean and covariance as $\mu_{f,k}$ and $\Sigma_{f,k}$. An eigenvalue composition for $\Sigma_{f,k}$ is performed to get the eigenvalues λ_1, λ_2, and λ_3 $(\lambda_1 \geq \lambda_2 \geq \lambda_3)$ and corresponding eigenvectors v_1, v_2, and v_3. Then, a plane is generated whose center and normal vector are $\mu_{f,k}$ and $v_3/\|v_3\|$, respectively. Similarly, for point $e \in E_k$ without being fitted, one line is obtained with its center $\mu_{e,k}$ and direction vector $v_1/\|v_1\|$. The above process requires 3-D kd-tree to search surrounding points, which is sensitive to the number of data points. To achieve the expected efficiency, we downsize E_k and F_k before fitting using voxel grid filter [31].

To further decrease computational cost of global pose estimation and map updating, the extracted line and plane features are sampled, and only significant features are kept. However, the above-mentioned voxel grid filter is not suitable because the direction vectors of lines or normal vectors of planes in the same voxel grid cannot be simply averaged. In [10], Deschaud proposed a point sampling method and it is modified in this article to sample lines and planes. Please refer to [10] for detailed description and we only focus on the critical parameter a_{2D}, where $a_{2D} = (\sigma_2 - \sigma_3)/\sigma_1$, and $\sigma_i = \sqrt{\lambda_i}$ $(1 \leq i \leq 3)$. If a_{2D} is directly utilized for plane sampling, a problem arises where it prefers the planes whose length is nearly equal to its width. This shall lead to a fact that some distant ground planes are removed because they are usually lanky due to the sparsity of LiDAR scans. To reserve these planes, the influence of the ratio of σ_2 to σ_1 on a_{2D} should be avoided. In this article, we take $a_{2D} = (\sigma_1 + \sigma_2)/(\sigma_1 + \sigma_2 + \sigma_3)$ and $a_{2D} = \sigma_1/(\sigma_1 + \sigma_2 + \sigma_3)$ for plane and line sampling, respectively. After sampling, the remaining line and plane features constitute the sets S_k and P_k, respectively.

2) Initializing Global Pose and Acquiring Surrounding Map: As defined in Section II, the latest optimized frame index of scan-to-map matching is l. Thus, we may use the optimized poses of scan-to-scan matching from lth to kth frames to compensate the corresponding pose change of scan-to-map matching. Formally, we define the pose transformation from frame $\{W\}$ to $\{L_k\}$ accumulated by scan-to-scan matching module as $T_{1:k}^L \triangleq T_{1:k-1}^L T_k^L,\ k \geq 2$, and $T_{1:1}^L \triangleq I$. Then, the initial global pose $\check{T}_k^W$ of kth LiDAR scan is acquired as follows:

$$\check{T}_k^W = T_l^W \left(T_{1:l}^L\right)^{-1} T_{1:k}^L \ \ 1 \leq l < k. \tag{17}$$

Besides, the surrounding map with line and plane features should be separated from $M(l)$, which is important to estimate global pose. To achieve fast processing, we incrementally construct a global grid map with the resolution $r_g = 10\,\mathrm{m}$. By this grip map, the line and plane features in $M(l)$ are projected to corresponding grids. For $s \in S_M(l)$, its grid coordinate g_s^W is calculated as follows:

$$g_s^W = \left(\left\lfloor \frac{c_s^W(x)}{r_g} \right\rfloor, \left\lfloor \frac{c_s^W(y)}{r_g} \right\rfloor, \left\lfloor \frac{c_s^W(z)}{r_g} \right\rfloor \right) \tag{18}$$

where $c_s^W(x),\ c_s^W(y), c_s^W(z)$ are the components of c_s^W. Similarly, the grid coordinate of $p \in P_M(l)$ is also acquired.

By searching grids around the position of $\check{T}_k^W$, the neighboring line and plane features are obtained, which form the surrounding map whose line and plane sets are denoted as $\mathcal{S}_M(l) \subseteq S_M(l)$ and $\mathcal{P}_M(l) \subseteq P_M(l)$, respectively.

3) Estimating Global Pose and Updating Map: Combining $\check{T}_k^W, S_k, P_k, \mathcal{S}_M(l)$, and $\mathcal{P}_M(l)$, the global pose T_k^W is estimated under the framework of ICP. The centers of all line or plane features in $\mathcal{S}_M(l)$ and $\mathcal{P}_M(l)$ are used to construct two 3-D kd-trees before ICP iterations. For the ith ICP iteration $(1 \leq i \leq I_W)$, where I_W is the maximum number of ICP iteration, each line $s \in S_k$ and each plane $p \in P_k$ are projected from $\{L_k\}$ to $\{W\}$, and we have

$$c_{s,k}^W = \prod \left(c_{s,k}^L; {}^{(i)}T_k^W \right) \ d_{s,k}^W = {}^{(i)}R_k^W \, d_{s,k}^L \tag{19}$$

$$c_{p,k}^W = \prod \left(c_{p,k}^L; {}^{(i)}T_k^W \right) \ n_{p,k}^W = {}^{(i)}R_k^W \, n_{p,k}^L \tag{20}$$

$$P_{s,k}^W \leftarrow \prod \left(\forall e \in P_{s,k}^L; {}^{(i)}T_k^W \right)$$

$$P_{p,k}^W \leftarrow \prod \left(\forall f \in P_{p,k}^L; {}^{(i)}T_k^W \right) \tag{21}$$

where ${}^{(i)}T_k^W = \begin{bmatrix} {}^{(i)}R_k^W & {}^{(i)}t_k^W \\ 0 & 1 \end{bmatrix} \in \mathcal{M}$ is pose transformation matrix in the optimization of ith ICP iteration with ${}^{(1)}T_k^W \triangleq \check{T}_k^W$ initially. On this basis, the best associations between S_k, P_k and $\mathcal{S}_M(l)$, $\mathcal{P}_M(l)$ are conducted. We denote with s_M^* the best

association of $s \in S_k$ in $\mathcal{S}_M(l)$, which is given as follows:

$$\begin{aligned} s_M^* = & \underset{s_M \in \mathcal{S}_M(l)}{\operatorname{argmax}} \left\| d_{s_M}^W \cdot d_{s,k}^W \right\| \\ & \text{s.t. } \left\| c_{s_M}^W - c_{s,k}^W \right\| < r_{\mathrm{ICP}} \\ & \left\| d_{s_M}^W \cdot d_{s,k}^W \right\| > \cos\theta_{\mathrm{ICP}} \end{aligned} \tag{22}$$

where r_{ICP} and θ_{ICP} are the radius and angle thresholds, respectively. In detail, (22) is solved by searching the kd-tree corresponding to $\mathcal{S}_M(l)$ within a given radius.

To formulate the line-to-line constraint, we stack the distances between all points in $P_{s,k}^W$ and the best association s_M^* to construct a distance vector as follows:

$$\mathrm{dis}_s = \begin{bmatrix} \vdots \\ \left(e_{s,k}^W - c_{s_M^*}^W \right) \times d_{s_M^*}^W \\ \vdots \end{bmatrix} \tag{23}$$

where $e_{s,k}^W$ is the coordinate of $e \in P_{s,k}^W$ in $\{W\}$.

Similarly, the best association $p_M^* \in \mathcal{P}_M(l)$ of $p \in P_k$ is shown as follows:

$$\begin{aligned} p_M^* = & \underset{p_M \in \mathcal{P}_M(l)}{\operatorname{argmin}} \left\| n_{p_M}^W \cdot \left(c_{p_M}^W - c_{p,k}^W \right) \right\| \\ & \text{s.t. } \left\| c_{p_M}^W - c_{p,k}^W \right\| < r_{\mathrm{ICP}} \\ & \left\| n_{p_M}^W \cdot \left(c_{p_M}^W - c_{p,k}^W \right) \right\| < d_{\mathrm{ICP}} \\ & \left\| n_{p_M}^W \cdot n_{p,k}^W \right\| > \cos\theta_{\mathrm{ICP}} \end{aligned} \tag{24}$$

where d_{ICP} is the distance threshold. By searching the kd-tree corresponding to $\mathcal{P}_M(l)$, p_M^* is obtained.

Using the distances between all points in $P_{p,k}^W$ and the corresponding plane p_M^*, the distance vector dis_p is defined as follows:

$$\mathrm{dis}_p = \begin{bmatrix} \vdots \\ \left(f_{p,k}^W - c_{p_M^*}^W \right) \cdot n_{p_M^*}^W \\ \vdots \end{bmatrix} \tag{25}$$

where $f_{p,k}^W$ is the coordinate of $f \in P_{p,k}^W$ in $\{W\}$.

Combining (23) and (25), the global pose optimization is formulated by minimizing the half of squared sum of distance vectors

$$\begin{aligned} {}^{(i)}T_k^{W^*} = \underset{{}^{(i)}T_k^W}{\operatorname{argmin}} \ \frac{1}{2} \Bigg(& \sum_{s \in S_k} \rho \left(\left\| \mathrm{dis}_s \right\|_2^2 \right) \\ & + \sum_{p \in P_k} \rho(\left\| \mathrm{dis}_p \right\|_2^2) \Bigg). \end{aligned} \tag{26}$$

Similar to the solution of (2), (26) is also solved by the composite manifold-based LM algorithm, and the residual vector d^W and Jacobian matrix J^W are described as follows:

$$d^W = \left[\cdots \ \mathrm{dis}_s^T \ \cdots \ \cdots \ \mathrm{dis}_p^T \ \cdots \right]^T \tag{27}$$

$$J^W = \left[\cdots \ \nabla\mathrm{dis}_s^T \ \cdots \ \cdots \ \nabla\mathrm{dis}_p^T \ \cdots \right]^T \tag{28}$$

where the line and plane gradient vectors $\nabla\mathrm{dis}_s$ and $\nabla\mathrm{dis}_p$ are given by

$$\nabla\mathrm{dis}_s = \begin{bmatrix} \vdots \\ \nabla\mathrm{dis}_{s,e} \\ \vdots \end{bmatrix} \tag{29}$$

$$\nabla\mathrm{dis}_p = \begin{bmatrix} \vdots \\ \nabla\mathrm{dis}_{p,f} \\ \vdots \end{bmatrix} \tag{30}$$

with $\nabla\mathrm{dis}_{s,e} = -[d_{s_M^*}^W]_\times [-[{}^{(i)}R_k^W e_{s,k}^W]_\times \ I_{3\times 3}] \in \mathbb{R}^{3\times 6}$ and $\nabla\mathrm{dis}_{p,f} = n_{p_M^*}^{W^T} [-[{}^{(i)}R_k^W f_{p,k}^W]_\times \ I_{3\times 3}] \in \mathbb{R}^{1\times 6}$.

Therefore, $\nabla\mathrm{dis}_s \in \mathbb{R}^{(3|P_{s,k}^W|)\times 6}$ and $\nabla\mathrm{dis}_p \in \mathbb{R}^{|P_{p,k}^W|\times 6}$. Then, one can obtain that the number of columns of J^W is six, which means that the matrix $(J^{W^T} J^W + \lambda I) \in \mathbb{R}^{6\times 6}$ and its inverse can be acquired easily. The optimized increment $\delta\xi^W$ of pose transformation can be calculated by

$$(J^{W^T} J^W + \lambda I)\delta\xi^W = -J^{W^T} d^W \tag{31}$$

and thus, the pose ${}^{(i)}T_k^W$ is updated as follows:

$${}^{(i)}T_k^W \leftarrow \delta\xi^W \boxplus {}^{(i)}T_k^W. \tag{32}$$

The aforementioned incremental optimization of ${}^{(i)}T_k^W$ is repeated until it converges or reaches a maximum iteration number O_W, where O_W is set to eight. Then, ${}^{(i+1)}T_k^W = {}^{(i)}T_k^W$ in the next ICP iteration. After the iterations are finished, the optimized global pose T_k^W is acquired via $T_k^W = {}^{(I_W+1)}T_k^W$.

After the global pose T_k^W is optimized, the global map $M(k)$ is initialized with the last global map $M(l)$. On one hand, for the line or plane feature in $M(k)$ that has associations with the elements in S_k or P_k, they shall be modified. By replacing ${}^{(i)}T_k^W$ and ${}^{(i)}R_k^W$ in (19) and (20) with T_k^W and R_k^W, lines in S_k and planes in P_k are projected from $\{L_k\}$ to $\{W\}$, where $T_k^W = \left[\begin{smallmatrix} R_k^W & t_k^W \\ 0 & 1 \end{smallmatrix} \right] \in \mathcal{M}$. Similar to the solutions of (22) and (24) with three thresholds r_{ICP}, d_{ICP}, and θ_{ICP}, data associations between $M(k)$ and S_k, P_k are searched. Also, to construct a more accurate map, the thresholds d_{ICP} and θ_{ICP} are replaced by smaller values d_{MAP} and θ_{MAP}. Suppose that $s_M^* \in \mathcal{S}_M(l)$ and $p_M^* \in \mathcal{P}_M(l)$ are the associated line and plane of $s \in S_k$ and $p \in P_k$, respectively, the modifications of s_M^* and p_M^* are as follows [32]:

$$\begin{aligned} c_{s_M^*}^W & \leftarrow \gamma c_{s_M^*}^W + (1-\gamma)\, c_{s,k}^W \\ d_{s_M^*}^W & \leftarrow \gamma d_{s_M^*}^W + (1-\gamma)\, d_{s,k}^W \\ d_{s_M^*}^W & \leftarrow \frac{d_{s_M^*}^W}{\left\| d_{s_M^*}^W \right\|} \end{aligned} \tag{33}$$

$$\begin{aligned} c_{p_M^*}^W & \leftarrow \gamma c_{p_M^*}^W + (1-\gamma)\, c_{p,k}^W \\ n_{p_M^*}^W & \leftarrow \gamma n_{p_M^*}^W + (1-\gamma)\, n_{p,k}^W \\ n_{p_M^*}^W & \leftarrow \frac{n_{p_M^*}^W}{\left\| n_{p_M^*}^W \right\|} \end{aligned} \tag{34}$$

where $0 < \gamma < 1$ and $\gamma = 0.9$ in the following experiments.

Algorithm 2: Scan-to-Map Matching.

Input: $M(l)$, E_k, F_k, $T_1^L, \ \ldots, T_k^L, T_l^W$
Output: T_k^W, $M(k)$
$S_k \leftarrow$ ExtractLines(E_k), $P_k \leftarrow$ ExtractPlanes(F_k);
Acquire $\check{T}_k^W$ via (17);
Obtain $\mathcal{S}_M(l)$ and $\mathcal{P}_M(l)$ from $M(l)$;
Construct two kd-trees by $\mathcal{S}_M(l)$ and $\mathcal{P}_M(l)$;
${}^{(1)}T_k^W \leftarrow \check{T}_k^W$;
for $i \in [1, \ I_W]$ **do //** outer loop for data association
  Obtain s_k^W and p_k^W by, respectively, projecting s_k^L and p_k^L according to (19)–(21);
  calculate s_M^* and p_M^* by (22) and (24);
  for $o \in [1, \ O_W]$ **do //** inner loop for optimization
    Calculate d^W and J^W using (27) and (28);
    Update ${}^{(i)}T_k^W$ via (31) and (32);
  end for
  ${}^{(i+1)}T_k^W \leftarrow {}^{(i)}T_k^W$;
end for
$T_k^W \leftarrow {}^{(I_W+1)}T_k^W$;
$M(k) \leftarrow$ UpdateMap$(\mathcal{S}_M(l), \ \mathcal{P}_M(l), T_k^W, S_k, P_k)$;
return

On the other hand, there exist some valuable line and plane features in S_k and P_k, which should also be added into the map $M(k)$. A way is to directly add all of them into $M(k)$, however, this shall destroy the sparsity of the global map. In this article, only the line or plane feature in S_k or P_k whose center is at least r_{MAP} away from those of the line or plane features in $M(k)$ are selected into $M(k)$, where r_{MAP} is defined as the feature resolution, which reflects the density of line and plane features.

Algorithm 2 describes the whole scan-to-map matching process, where the functions ExtractLines() and ExtractPlanes() are used to extract lines and planes, respectively, and UpdateMap() refers to the map updating.

D. Computation Complexity Analysis

In this section, the computation complexity of the proposed method, including point feature extraction, scan-to-scan matching, and scan-to-map matching modules, is given.

For the point feature extraction module, there are two parts—ground plane fitting and edge/planar point feature extraction. The complexity of the former is $\mathcal{O}(|X_k|)$ due to the fact that it needs to traverse the whole point cloud to find ground points and thus, the time cost is proportional to the number of ground points, where X_k is the point cloud of kth LiDAR scan. Considering that the sorting of points in X_k plays the major role for the latter in terms of computation, its complexity is nearly $\mathcal{O}(|X_k|\log_2|X_k|)$ based on a general sense of the sorting algorithm. Compared to $\mathcal{O}(|X_k|\log_2|X_k|)$, $\mathcal{O}(|X_k|)$ can be omitted and one can obtain the approximate computation complexity $\mathcal{O}_{fe}$ of point feature extraction by $\mathcal{O}(|X_k|\log_2|X_k|)$. After point feature extraction, edge and planar point feature sets E_k and F_k are obtained.

In scan-to-scan matching, data association and pose optimization are conducted. The first step toward data association is to build two kd-trees by E_{k-1} and F_{k-1}, respectively, whose computation complexity is considered as $\mathcal{O}(|E_{k-1}|\log_2|E_{k-1}| + |F_{k-1}|\log_2|F_{k-1}|)$. Then, for each point feature in E_k and F_k, neighborhood searching is performed in the aforementioned two kd-trees for data association, and the computation complexity is $\mathcal{O}(|E_k|\log_2|E_{k-1}| + |F_k|\log_2|F_{k-1}|)$. It is noted that the data association is performed in an iterative way. As a result, the computation complexity of I_s iterations becomes $\mathcal{O}(I_s|E_k|\log_2|E_{k-1}| + I_s|F_k|\log_2|F_{k-1}|)$. After data association, the pose optimization is executed and its complexity is obviously less than that of data association, which is therefore ignored. Based on the above analysis, the computation complexity $\mathcal{O}_{ss}$ of scan-to-scan matching can be written as $\mathcal{O}((|E_{k-1}| + I_s|E_k|)\log_2|E_{k-1}| + (|F_{k-1}| + I_s|F_k|)\log_2|F_{k-1}|)$.

In scan-to-map matching, line and plane extraction, surrounding map construction, data association, pose optimization, as well as map updating are main factors for computational analysis. First, the computational cost of line and plane extraction is proportional to the number of points in E_k and F_k. After that, line and plane feature sets S_k and P_k are obtained. Then, the cost of surrounding map construction is proportional to the number of points in line and plane sets $\mathcal{S}_M(l)$ and $\mathcal{P}_M(l)$ of the surrounding map. Next, similar to scan-to-scan matching, the computation complexity of data association and pose optimization is considered as $\mathcal{O}((|\mathcal{S}_M(l)| + I_W|S_k|)\log_2|\mathcal{S}_M(l)| + (|\mathcal{P}_M(l)| + I_W|P_k|)\log_2|\mathcal{P}_M(l)|)$, where I_W is the maximum number of iteration of scan-to-map matching. Notice that the cost of map updating is mainly occupied by data association, and its complexity can be expressed by $\mathcal{O}(|S_k|\log_2|\mathcal{S}_M(l)| + |P_k|\log_2|\mathcal{P}_M(l)|)$. From the above description, one can see that the line/plane extraction and surrounding map construction have little effect on computation complexity, which will not be taken into account. Therefore, the complexity $\mathcal{O}_{sm}$ of scan-to-map matching is approximately $\mathcal{O}((|\mathcal{S}_M(l)| + (I_W+1)|S_k|)\log_2|\mathcal{S}_M(l)| + (|\mathcal{P}_M(l)| + (I_W+1)|P_k|)\log_2|\mathcal{P}_M(l)|)$.

Overall, the computation complexity is given by $\mathcal{O}_{fe} + \mathcal{O}_{ss} + \mathcal{O}_{sm}$ and $\mathcal{O}_{fe} + \mathcal{O}_{ss}$ for nonskipped frames and skipped frames, respectively, where each skipped frame only performs feature extraction and scan-to-scan matching while each nonskipped frame also performs scan-to-map matching. The best accuracy can be reached without any skipped frame. Similarly, by replacing S_k, P_k, $\mathcal{S}_M(l)$, $\mathcal{P}_M(l)$ with E_k, F_k, $\mathcal{E}_M(l)$, $\mathcal{F}_M(l)$, respectively, one can acquire the computation complexity of LOAM, where $\mathcal{E}_M(l)$ and $\mathcal{F}_M(l)$ are surrounding maps with edge and flat point features, respectively; E_k and F_k are edge and flat point sets of kth frame, respectively. Thanks to the sparseness of line and plane features, we have $|S_k| < |E_k|$, $|P_k| < |F_k|$, $|\mathcal{S}_M(l)| < |\mathcal{E}_M(l)|$, and $|\mathcal{P}_M(l)| < |\mathcal{F}_M(l)|$. It is seen that the computational cost of our method is less than that of LOAM.

IV. Experiments

The proposed SGLO approach is verified on the KITTI dataset and an outdoor scenario. The parameters are as follows: $r_{\mathrm{ICP}} =$ 2 m, $d_{\mathrm{ICP}} = 1$ m, $\theta_{\mathrm{ICP}} = 20^\circ$, $\gamma = 0.9$, and d_{MAP}, θ_{MAP} are set to 0.5 m and 5°.

TABLE II
ACCURACY COMPARISON OF THE PROPOSED SGLO WITH EXISTING METHODS

Seq \ Methods, t_{rel}(%)	LOAM	LOAM_ Velodyne	LeGO-LOAM	A-LOAM	SGLO	LO-Net	LO-Net-M
00	0.78	3.410	1.347	0.791	**0.775**	1.47	0.78
01	1.43	6.542	27.43	1.960	1.811	**1.36**	1.42
02	**0.92**	5.663	2.087	4.564	1.050	1.52	1.01
03	0.86	1.635	1.237	0.945	**0.702**	1.03	0.73
04	0.71	1.094	1.245	0.765	0.705	**0.51**	0.56
05	0.57	1.318	0.887	**0.504**	0.596	1.04	0.62
06	0.65	1.007	0.87	0.615	**0.483**	0.71	0.55
07	0.63	1.256	0.77	**0.453**	0.515	1.70	0.56
08	1.12	2.159	1.337	1.107	**1.059**	2.12	1.08
09	0.77	1.437	1.282	0.739	**0.733**	1.37	0.77
10	**0.79**	1.912	1.675	1.006	0.977	1.80	0.92

A. *Experimental Evaluation on KITTI Dataset*

KITTI training dataset contains 11 point cloud sequences with ground truth [33], where the point cloud is collected by Velodyne HDL64. These point cloud sequences seq00–seq10 cover a wide variety of environments of urban city, highway, and country, which provides a benchmark platform for comparing different approaches. The usage process of KITTI dataset is described as follows. First, we download the raw LiDAR data of sequences 00–10 from the KITTI website [34], where there are timestamp, ground truth pose, and 3-D coordinates of point cloud for each frame in each sequence. Due to the fact that the 3-D coordinates of point cloud are stored in ".bin" form, which is not easily read, the pykitti [35] is utilized to transform the point cloud from ".bin" to ".txt", where each line in the txt file corresponds to a 3-D coordinate. Afterward, we write codes to integrate corresponding timestamp and 3-D coordinates of point cloud into sensor_msgs::PointCloud2 message [36]. Besides, corresponding timestamp and pose are also integrated into the nav_msgs::Odometry message [36]. Then, they are stored in a rosbag file [37]. Finally, a rosbag file is obtained for each KITTI sequence, which includes a series of sensor_msgs::PointCloud2 and nav_msgs::Odometry messages, and can be played repeatedly. The sensor_msgs::PointCloud2 message of the rosbag file is inputted to the given method by a callback function. Meanwhile, the nav_msgs::Odometry message is visualized in rviz [38] to reflect the ground truth trajectory.

In this section, SGLO with $r_{\mathrm{MAP}} = 0.2$ m is compared with mainstream geometric map-based methods including LOAM [21], LOAM_Velodyne [39], LeGO-LOAM [6], and A-LOAM [22]. Additionally, two LiDAR odometry methods based on deep convolutional network are also involved in the comparison. LO-Net [40] is an end-to-end LiDAR odometry based on deep convolutional network. In order to further reduce the estimation error, the authors also extend LO-Net via adding a scan-to-map matching module using point-to-plane ICP, which is denoted as LO-Net-M [40]. LO-Net and LO-Net-M rely on the computing resource of graphics processing unit (GPU) while other methods run on central processing unit (CPU). Specially, LOAM [21] runs only at 10% of the real-time speed to achieve best results due to the heavy computational burden of scan-to-map matching. Actually, LOAM_Velodyne [39] is a version of LOAM with a real-time speed. Table II presents the comparison results in terms of accuracy on the KITTI dataset, where t_{rel} refers to the relative translation drift averaged over trajectories of 100 to 800 m length. The best result for each sequence is labeled in red. One can see that SGLO outperforms other methods including LOAM in five sequences. The accuracy advantage of the proposed method can also be seen from Fig. 2, which gives the comparison results of average relative translation drifts over 11 sequences, using segments of trajectories at 100, 200, …, 800 m lengths and 4, 6, …, 24 m/s speeds, respectively. Fig. 3 illustrates the trajectories estimated by different methods with four sequences on the KITTI dataset, and our SGLO is more coincident with the ground truth. It shall be noted that the estimation errors exist in almost all of the odometry methods. As shown in the ranking list of KITTI dataset benchmark [33], the estimation errors of LiDAR odometry methods are generally smaller than those of visual odometry methods. From the comparison results with existing methods, the estimation error of the proposed method SGLO is acceptable.

The motivation of our article is to reduce the high computation complexity of scan-to-map matching of LOAM [21], which is mainly caused by the discreteness and denseness of point feature-based geometric map. This drawback is solved by our sparse geometric map with line and plane features and line-to-line and plane-to-plane data associations. The averaged running time and skipped frame rate of scan-to-map matching module for different methods are given in Tables III and IV, respectively, where the skipped frame rate is calculated by a ratio of the numbers of unprocessed frames and total frames of scan-to-map matching.

It is seen that the average running time of SGLO is lower than 33% of LOAM_Velodyne [39] and 62% of A-LOAM [22], which demonstrates the high efficiency of SGLO. On the other hand, as mentioned in [21], the best accuracy can be reached without any skipped frame. From Table IV, the average skipped frame rate for our SGLO is 1.26%, which is far below those of the other methods. This paves the foundation of high accuracy. The aforementioned experimental results on the KITTI dataset reveal that our proposed SGLO performs well.

B. *Experimental Evaluation in an Outdoor Scenario*

In this section, we compare our approach with $r_{\mathrm{MAP}} = 0.5$ m and LOAM_Velodyne [39], LeGO-LOAM [6], and A-LOAM

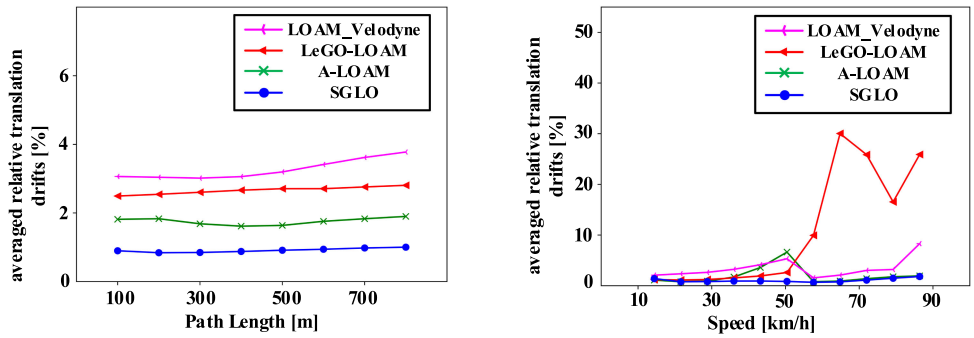

Fig. 2. Comparison of average relative translation drifts with respect to path length and speed over 11 sequences, respectively.

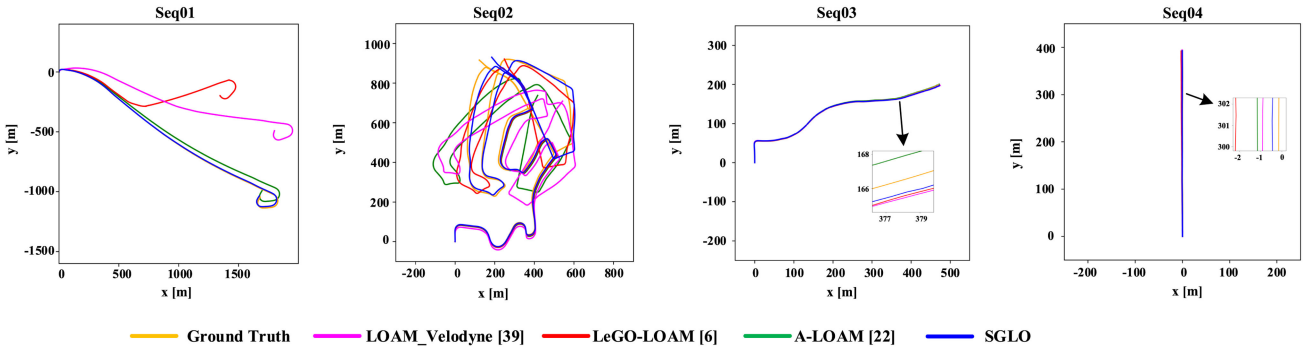

Fig. 3. Comparison of trajectories estimated by different methods on the KITTI dataset.

TABLE III
COMPARISON OF AVERAGE RUNNING TIMES (MS) OF SCAN-TO-MAP MATCHING ON THE KITTI DATASET

	LOAM_ Velodyne	LeGO-LOAM	A-LOAM	SGLO
00	266	149	140	94
01	260	245	144	64
02	258	149	137	84
03	256	204	140	86
04	261	241	135	71
05	278	159	155	94
06	343	254	174	87
07	219	130	122	94
08	304	161	154	93
09	223	152	124	84
10	195	123	100	87
average	260.27	178.82	138.64	**85.27**

TABLE IV
COMPARISON OF SKIPPED FRAME RATE (%) OF SCAN-TO-MAP MATCHING ON THE KITTI DATASET

Seq \ Methods (r_{skip}(%))	LOAM_ Velodyne	LeGO-LOAM	A-LOAM	SGLO
00	61	30	25	1.9
01	60	57	26	0.4
02	60	31	23	1.0
03	60	49	24	1.1
04	60	56	19	0
05	63	34	31	2.5
06	70	59	38	1.5
07	53	22	16	1.5
08	66	36	31	2.9
09	54	31	15	0.8
10	48	19	6	0.3
average	59.55	38.55	23.09	**1.26**

Fig. 4. Snapshot of global scene of our outdoor experiment acquired from AutoNavi Map.

[22] in an outdoor scenario. The experiment employs a Velodyne VLP-16 mounted on a car, which runs at a speed of 20–30 km/h. Compared with Velodyne HDL64, VLP-16 provides less information, and thus, there is no need to sample lines and planes in the process of scan-to-map matching. We drove the car along a road three times and the total length is about 3.7 km. Fig. 4 gives the snapshot of global scene of our outdoor experiment acquired from AutoNavi Map, and the trajectories and global maps estimated by different methods are presented in Fig. 5. One can see from Fig. 5(c) that the global map of A-LOAM [22] shows

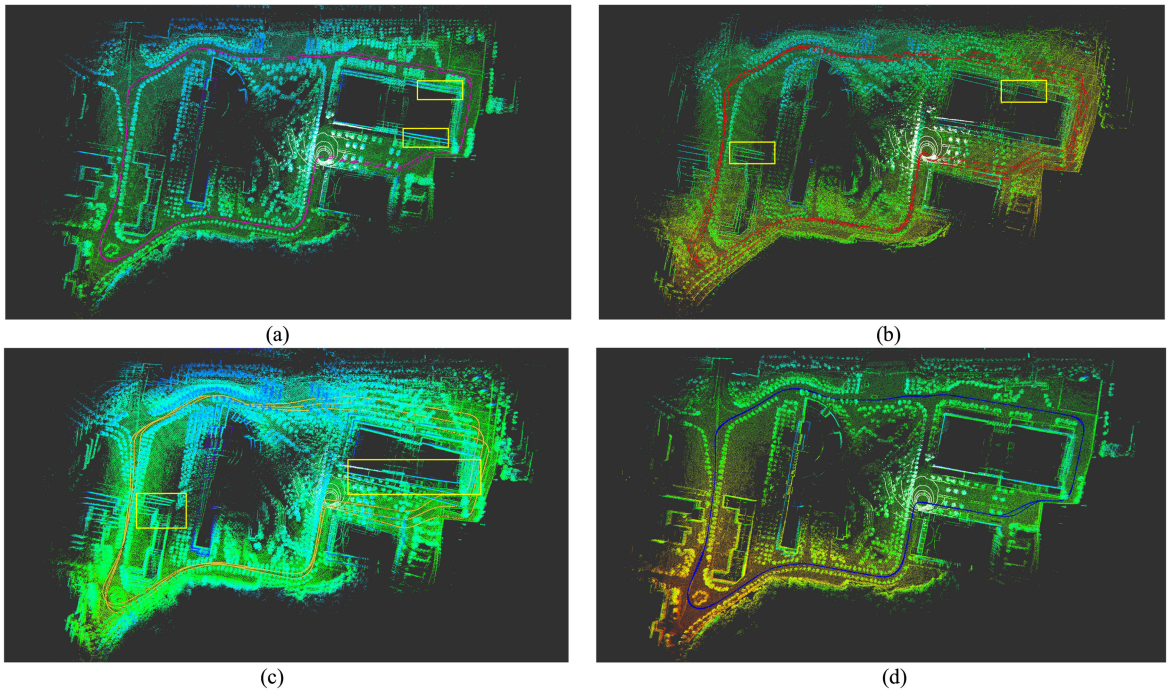

Fig. 5. Estimated trajectories and global maps of different methods in the outdoor scenario. (a) LOAM_Velodyne [39]. (b) LeGO-LOAM [6]. (c) A-LOAM [22]. (d) SGLO.

TABLE V
COMPARISON OF METHODS FOR SCAN-TO-MAP MATCHING IN THE OUTDOOR SCENARIO

	LOAM_ Velodyne	LeGO-LOAM	A-LOAM	SGLO
averaged running time (ms)	242	157	301	**70**
skipped frame rate (%)	59	38	66	**1.2**

obvious inconsistency with the ghosting. For LOAM_Velodyne [39] and LeGO-LOAM [6], ghosting appears especially in the right parts of Fig. 5(a) and (b). Fig. 5(d) depicts the result of the proposed SGLO where the global map is clear with a good consistency.

Table V presents the comparison of different methods for scan-to-map matching in terms of average running time and skipped frame rate. The average running time of LOAM_Velodyne, LeGO-LOAM, and A-LOAM is thrice, twice, and four times as much as that of SGLO. For SGLO, its average running time is 70 ms, and it skips much less frames than other methods, which guarantees the accuracy.

V. CONCLUSION

This article proposes an SGLO. The line and plane features are introduced to the scan-to-map matching module, which effectively reduces the storage memory and searching cost of the map. Moreover, line-to-line and plane-to-plane associations are employed, and the association amount is significantly decreased; with the pose optimization on the composite manifold, a fast and precise estimation result is achieved. The experimental results on the KITTI dataset and an outdoor scenario reveal the effectiveness of our proposed SGLO. The proposed method aims at reducing the computational cost of point feature-based scan-to-map matching by only retaining sparse geometric line and plane features with line-to-line and plane-to-plane data associations. A possible problem is that the pose estimation relies on an appropriate initialization, which requires that the motion of LiDAR is gentle. In the near future, inertial measurement unit (IMU) shall be combined to better initialize the pose estimation of LiDAR. Besides, we will consider the introduction of loop detection and pose graph optimization to further decrease the estimation error.

REFERENCES

[1] L. Chen, A. Yang, H. Hu, and W. Naeem, "RBPF-MSIS: Toward rao-blackwellized particle filter SLAM for autonomous underwater vehicle with slow mechanical scanning imaging sonar," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3301–3312, Sep. 2020.

[2] Y. Liu, R. Xiong, Y. Wang, X. Xie, X. Liu, and G. Zhang, "Stereo visual-inertial odometry with multiple Kalman filters ensemble," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6205–6216, Oct. 2016.

[3] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[4] S. Hwang and J. Song, "Monocular vision-based SLAM in indoor environment using corner, lamp, and door features from upward-looking camera," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4804–4812, Oct. 2011.

[5] T. Lee, C. Kim, and D. D. Cho, "A monocular vision sensor-based efficient SLAM method for indoor service robots," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 318–328, Jan. 2019.

[6] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[7] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 5000–5007.

[8] C. Park *et al.*, "Elastic LiDAR fusion: Dense map-centric continuous-time SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1206–1213.
[9] W. Hess *et al.*, "Real-time loop closure in 2D LiDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1271–1278.
[10] J. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2480–2485.
[11] S. Ceriani, C. Sanchez, P. Taddei, E. Wolfart, and V. Sequeira, "Pose interpolation SLAM for large maps using moving 3D sensors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 750–757.
[12] N. Gelfand *et al.*, "Geometrically stable sampling for the ICP algorithm," in *Proc. Int. Conf. 3D Digit. Imag. Model.*, 2003, pp. 260–267.
[13] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, 2013.
[14] M. Magnusson, A. Lilenthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *J. Field Robot.*, vol. 24, no. 10, pp. 803–827, 2007.
[15] T. Stoyanov and M. Magnusson, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1377–1393, 2012.
[16] L. Sun *et al.*, "DLO: Direct LiDAR odometry for 2.5D outdoor environment," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 774–778.
[17] J. Fossel, K. Tuyls, B. Schnieders, D. Claes, and D. Hennes, "NOctoSLAM: Fast octree surface normal mapping and registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6764–6769.
[18] A. Hornung, K. M. Wurm, and M. Bennewitz, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[19] A. Das, J. Servos, and S. L. Waslander, "3D scan registration using the normal distributions transform with ground segmentation and point cloud clustering," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 2207–2212.
[20] B. Douillard *et al.*, "Scan segments matching for pairwise 3D alignment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3033–3040.
[21] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," *Robot., Sci. Syst.*, 2014.
[22] T. Qin and S. Cao, *A-LOAM*. [Online]. Available: https://github.com/HKUST-Aerial-Robotics/A-LOAM
[23] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 3062–3067.
[24] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 4486–4491.
[25] K. Ji *et al.*, "CPFG-SLAM: A robust simultaneous localization and mapping based on LIDAR in off-road environment," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 650–655.
[26] D. Zermas, I. Izzat, and N. Papnikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5067–5073.
[27] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[28] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, 1992.
[29] J. Sola, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2019, *arXiv: 1812.01537v6*.
[30] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[31] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1–4.
[32] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," *Robot., Sci. Syst.*, 2018.
[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[34] [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_odometry.php
[35] [Online]. Available: https://github.com/utiasSTARS/pykitti
[36] [Online]. Available: http://docs.ros.org/kinetic/api/
[37] [Online]. Available: http://wiki.ros.org/rosbag
[38] [Online]. Available: https://wiki.ros.org/rviz/
[39] [Online]. Available: https://github.com/laboshinl/loam_velodyne
[40] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, and M. Cheng, "LO-Net: Deep real-time Lidar odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8473–8482.

Shuang Liang received the B.S. degree from the Dalian University of Technology, Dalian, China, in 2017. He is currently working toward the Ph.D. degree in control theory and control engineering with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His current research interests include intelligent robots and navigation and manipulation of service robots.

Zhiqiang Cao (Senior Member, IEEE) received the B.S. and M.S. degrees from the Shandong University of Technology, Jinan, China, in 1996 and 1999, respectively, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

He is currently a Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include environmental cognition, service robots and multirobot systems.

Peiyu Guan received the B.S. degree from Jilin University, Changchun, China, in 2017. She is currently working toward the Ph.D. degree in control theory and control engineering with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

Her current research interests include intelligent robots and visual localization of service robots.

Chengpeng Wang received the B.E. degree from the University of Science and Technology Beijing, Beijing, China, in 2019. He is currently working toward the Ph.D. degree in control theory and control engineering with the Institute of Automation, Chinese Academy of Sciences, Beijing.

His current research interests include navigation and manipulation of service robots.

Junzhi Yu (Senior Member, IEEE) received the B.E. degree in safety engineering and the M.E. degree in precision instruments and mechanology from the North University of China, Taiyuan, China, in 1998 and 2001, respectively, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2003.

He is currently a Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include biomimetic robots, intelligent control, and intelligent mechatronic systems.

Shuo Wang received the B.E. degree in electrical engineering from the Shenyang Architecture and Civil Engineering Institute, Shenyang, China, in 1995, the M.E. degree in industrial automation from Northeastern University, Shenyang, in 1998, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2001.

He is currently a Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include intelligent robots, underwater robots, and multirobot systems.