Efficient LiDAR Odometry for Autonomous Driving

Xin Zheng and Jianke Zhu, *Senior Member, IEEE*

***Abstract*—LiDAR odometry plays an important role in self-localization and mapping for autonomous navigation, which is usually treated as a scan registration problem. Although having achieved promising performance on the KITTI odometry benchmark, the conventional tree-based neighbor search still has the difficulty in dealing with the large-scale point cloud efficiently. The recent spherical range image-based method enjoys the merits of fast nearest neighbor search by spherical mapping. However, it is not very effective to deal with the ground points nearly parallel to LiDAR beams. To address these issues, we propose a novel efficient LiDAR odometry approach by taking advantage of both non-ground spherical range images and bird's-eye-view maps for ground points. Moreover, a range adaptive method is introduced to robustly estimate the local surface normal. Additionally, a very fast and memory-efficient model update scheme is proposed to fuse the points and their corresponding normals at different time-stamps. We have conducted extensive experiments on the KITTI odometry benchmark and UrbanLoco dataset, whose promising results demonstrate that our proposed approach is effective.**

***Index Terms*—LiDAR odometry, autonomous driving, normal estimation, scan registration.**

I. Introduction

AUTONOMOUS driving has attracted a large amount of research efforts due to the increasing needs in robotics, where odometry plays an important role in building the high-precision map for navigation. Moreover, odometry is essential to accurate self-localization for path planning and environment perception, which is the key to driving safety. An effective odometry method should be robust to various environments including urban, highway and country road [1]. In contrast to the approaches using cameras [2], [3], LiDAR odometry is able to deal with large lighting variations by taking advantage of its active sensor emitting the laser beams. In this paper, we focus our attention on LiDAR-based method.

Generally, LiDAR odometry can be treated as a registration problem between the current scan and the reference point cloud model, which is effectively solved by Iterative Closed Point (ICP) algorithm [4]. As a local optimization-based method, ICP requires good initialization for better convergence. More importantly, the accuracy of pose prediction is highly dependent on the quality of inlier correspondences built by nearest neighbor search, which is usually the computational bottleneck in LiDAR odometry.

To this end, SuMa [5] directly maps the scattered 3D point cloud onto a 2D spherical range image, where the correspondences can be very efficiently found within a small local patch. However, it is not as effective as KD-Tree [6], [7] on the KITTI odometry benchmark [8]. As depicted in Fig. 1, LiDAR scans are relatively dense on the surface perpendicular to laser beams while being sparse in the region parallel to laser beams. Therefore, range image representation may miss ground information during optimization. Two adjacent pixels in the range image are far away from each other on the ground, which is most likely belong to the different surfaces. The cross-product of range image gradients is not effective to compute normals, which inevitably leads to inferior performance.

To address the above limitations, we propose a novel Efficient LiDAR Odometry (ELO) approach in this paper. In addition to the spherical range projection, we suggest taking advantage of a bird's-eye-view (BEV) map that effectively retains the neighborhood relationship of ground surface points nearly parallel to laser beams. Thus, LiDAR odometry is formulated into a nonlinear least square minimization problem, where the non-ground cost and ground potential are adaptively fused according to their inlier ratio. Instead of using cross product, a range adaptive method is introduced to robustly estimate the normals, which is computed by eigen-decomposition with outlier rejection. A very fast and memory-efficient model update scheme is proposed to fuse the range image and their corresponding normal map at different time-stamps.

In summary, the main contributions of this paper are: 1) an efficient LiDAR odometry approach by taking advantage of both non-ground spherical range image and ground BEV map; 2) a robust range adaptive normal estimation method for LiDAR scan registration; 3) a very fast and memory efficient model update scheme that makes use of spherical range image and ground BEV map; 4) experiments on the KITTI odometry benchmark and UrbanLoco dataset [9] show that the proposed method not only achieves the promising results but also runs over 169 frames per second.

II. Related Work

In this paper, we investigate the problem of LiDAR odometry that is typically treated as a scan registration problem in literature. Given the input point clouds perceived by LiDAR at two consecutive timestamps, the most popular solution for point cloud alignment is Iterative Closed Point (ICP) algorithm [4], [10], which updates the transformation iteratively until convergence. Practically, the reliable correspondences between scans are essential to facilitate the effective ICP registration, where the nearest neighbor search is employed to obtain the matched

Manuscript received April 21, 2021; accepted August 17, 2021. Date of publication September 8, 2021; date of current version September 14, 2021. This letter was recommended for publication by Associate Editor M. Magnusson and Editor S. Behnke upon evaluation of the reviewers' comments. The work was supported by the National Natural Science Foundation of China under Grant 61831015. *(corresponding author: Jianke Zhu).*

The authors are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China, and also with the Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, Hangzhou, China (e-mail: xinzheng@zju.edu.cn; jkzhu@zju.edu.cn).

Digital Object Identifier 10.1109/LRA.2021.3110372

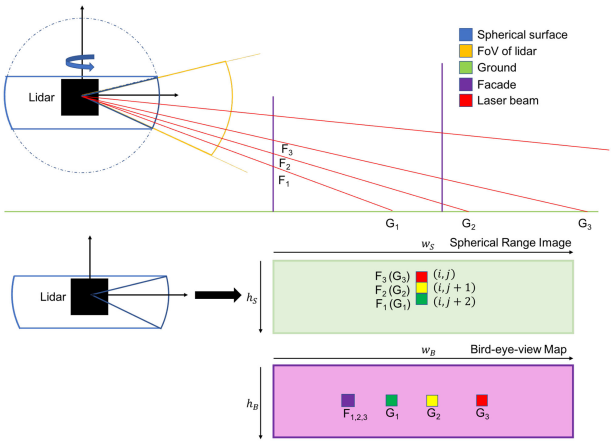


Fig. 1. Example on the spherical projection for the facade points F_1, F_2, F_3 and ground points G_1, G_2, G_3.

points. To this end, various data structures are presented to efficiently find the closed points.

Tree-based method [11] is widely used to retrieve the nearest neighbors with the space complexity of $O(n)$ and the time complexity of $O(\log n)$. The most successful LiDAR odometry approach LOAM [6] is based on KD-Tree, which has achieved low-drift with real-time performance in real-world applications. It selects the feature points by computing the roughness of a point on each scan line. Specifically, low roughness is denoted as planar feature while high roughness is denoted as edge feature. A variant of ICP algorithm combing both point-to-line [12] and point-to-plane [13] metrics makes it reliable in different scenarios. To obtain real-time performance, LOAM predicts poses at 10 Hz in frame-to-frame operation and adjusts drift at 1 Hz in frame-to-model operation. Despite promising performance, simply dividing points by roughness cannot reflect the property of real geometry. Additionally, storing the local map points in KD-Tree [11] is inconvenient for parallel computing. As in [14], Deschaud [7] employs Principal Component Analysis (PCA) to select the salient planar features, which represents the map as implicit moving least squared surface. Although having achieved encouraging results, it is very computational intensive.

In [15], [16], the 3D voxel grid-based approaches are able to find the nearest neighbors with the time complexity of $O(1)$, where each cell describes the geometry of local region. Similar to GICP [17], it takes into account of point distribution in scan registration. The main drawnback of 3D grid map representation is that it consumes huge storage especially in the outdoor environment with the fine resolution.

Behley and Stachniss [5] project 3D unordered LiDAR points onto the spherical range image, where the nearest neighbors can be very efficiently found within a small patch in 2D image space. Moreover, the surfel-based mapping [18] is introduced into the task of LiDAR odometry. The spherical range image of the local map in the frame-to-model registration is rendered by an active surfel map. It is worthy of noting that mapping 3D scatter points onto a 2D image plane preserves the spatial neighborhood relationship that can be easily implemented by the efficient parallel computing technique. However, this method and its improved version [19] are inferior to the KD-Tree based methods [6].

Nearly half of observations from the LiDAR belong to ground regardless of the various scenarios in driving. As for ground segmentation, Petrovskaya and Thrun [20] make use of neighboring beams and the angle between beams to segment points. Moosmann *et al.* [21] employ the convexity to find non-flat regions. An image-based segmentation method is proposed in [22]. Shan and Englot [23] present a lightweight ground-optimized LiDAR odometry method for the environment with the various terrain, which introduces the two-step optimization for pose prediction. It aims to reduce the computational cost by taking into account of the ground points. Instead, our proposed approach employs a ground BEV map to deal with the information loss issue in range image representation.

III. Efficient LiDAR Odometry

In this section, we present our proposed approach to efficient LiDAR odometry for autonomous driving. Firstly, we formulate the problem into a non-linear least squares minimization problem with both non-ground and ground costs. Secondly, we present the non-ground cost using spherical range image and a range adaptive normal estimation method. Thirdly, we give the details on ground cost with 2D bird's-eye-view map. Finally, we suggest an efficient map update scheme for both non-ground range image and ground map.

A. Efficient LiDAR Odometry for Autonomous Vehicles

1) Overview: As in [5], [6], [23], LiDAR odometry is formulated as the frame-to-model registration problem, which aims at finding an accurate transformation between the consecutive scans.

As depicted in Fig. 2, the raw 3D point cloud from LiDAR is firstly projected onto a spherical range image to facilitate the fast segmentation and non-ground cost (Section III-B). Secondly, the ground points are segmented from the resulting spherical range image, which are further projected onto the 2D bird's-eye-view map to form the ground cost function (Section III-C). Thirdly, the normal map of spherical projection image is computed by range adaptive method (Section III-B4), which is employed to estimate the pose increment by ICP. Finally, we update both the non-ground spherical range model and BEV ground map through the memory efficient update scheme (Section III-D).

2) Proposed Fusion Approach: To facilitate the effective registration, fast nearest neighbor search is the key to find the correspondences between the current scan and point could model. As the computational bottleneck for LiDAR odoemtry, we directly map the scattered 3D points onto a 2D spherical range image to efficiently find the nearest neighbors inspired by the projective data association [5].

Since the LiDAR points are unevenly distributed, the perceived ground points are usually sparser than non-ground ones. This incurs the issue that the adjacent pixels of ground points in spherical range image do not belong to the same local surface. We show an example in Fig. 1. Given three adjacent laser beams, the intersections of facade and ground are F_1, F_2, F_3 and G_1, G_2, G_3, respectively. Their corresponding pixels in the spherical range image are in the same column. Points lying on the facade perpendicular to laser beams are the exact nearest neighbors on one surface. However, the distance between adjacent

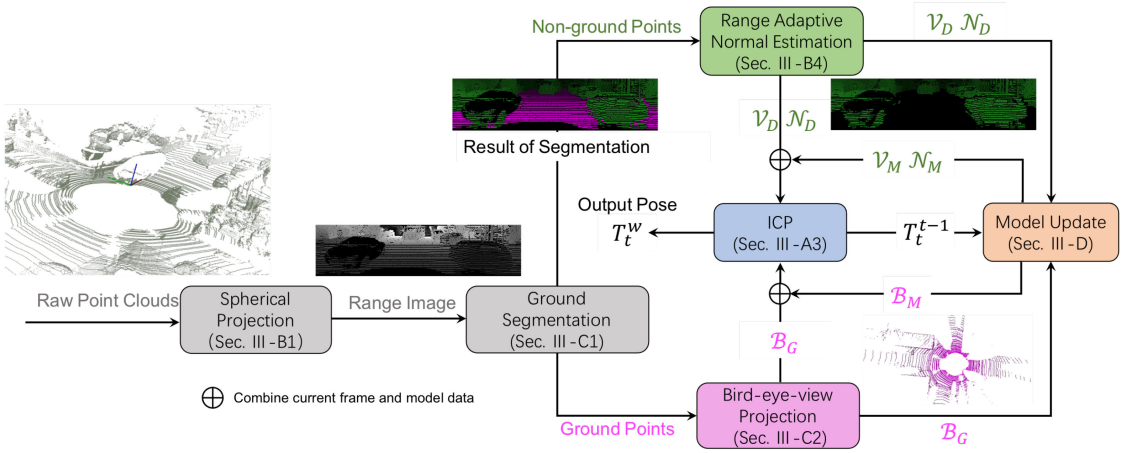


Fig. 2. Overview of our proposed efficient LiDAR odometry approach. The point cloud is projected onto a range image by spherical projection, which is used perform ground segmentation. The ground points are projected a birds-eye-view map. The normal map is estimated by range adaptive method, which is used to compute the pose increment by ICP.

pixels can be very large when the reflection surface is parallel to laser beams. Moreover, spherical range image divides the space by angle, which means that plenty of ground points between $\overline{G_1G_2}$ or $\overline{G_2G_3}$ would project into same pixel especially long range ground points. This will lead to the incorrect correspondence assignments and large errors in estimating the normal of ground region by the conventional spherical range image.

To deal with the above problem, we suggest to make use of bird's-eye-view (BEV) map in order to taken into careful consideration of the abundant ground information, which greatly reserves the neighborhood relationship among ground points. As illustrated in Fig 2, BEV map is a top-down projection capturing LiDAR points information. Let z-axis denote vertical direction. Since the space is divided on $x-y$ plane, the neighborhood information of ground surface is kept in BEV map. Unfortunately, the surface perpendicular to laser beams like those facade points F_1, F_2, F_3 are condensed into one pixel in BEV map. To this end, we propose a fusion approach by taking advantage of both non-ground spherical range image and ground BEV map in this paper.

3) Formulation: Assuming that the current scan is observed in LiDAR coordinate system L_t and the model is represented in the coordinate system L_{t-1}, where the subscript t is the timestamp. We aim at finding the transformation $T_{L_t}^{L_{t-1}}$ that efficiently aligns the current scan with previous model. To avoid gimbal lock affected by Euler angle representations, we parameterize the transformation by $T_{L_t}^{L_{t-1}} \in \mathrm{SE}(3)$, which is denoted as T_t^{t-1} for simplicity. Specifically, we employ an adaptive weighting strategy to fuse the non-ground cost E_S and ground cost E_G, which forms the following minimization:

$$\min_{T_t^{t-1}} wE_S + (1-w)E_G, \tag{1}$$

where $w = w_1 w_2$ is the product of two coefficients. w_1 is a user-defined weight to balance the number of non-ground and ground points, and w_2 is a ratio between non-ground and ground points.

Obviously, the above problem is a nonlinear least squares minimization that can be efficiently solved through an iterative Gauss-Newton algorithm. Considering the large pose changes, we employ a constant velocity assumption to initialize the transformation T_t^{t-1}. At each iteration, the pose update $\Delta T \in \mathfrak{se}(3)$ is estimated by $\Delta T = (J^T J)^{-1} J^T \mathbf{e}$, where $\mathbf{e}$ denotes the residual vector. $J \in \mathbb{R}^{1\times 6}$ is weighted Jacobian matrix:

$$J = wJ_S + (1-w)J_G, \tag{2}$$

where J_S is the Jacobian of non-ground term, and J_G is the Jacobian of ground term. The relative pose estimation T_t^{t-1} is iteratively updated by $T_t^{t-1} = \exp(\Delta T) T_t^{t-1}$ until the convergence, where $\exp(\cdot) : \mathfrak{se}(3) \mapsto SE(3)$ is the exponential map in Lie Algebra.

B. Non-Ground Cost E_S

We give the details on non-ground cost E_S that is computed by spherical range image.

1) Spherical Range Image (SRI): The input data acquired from LiDAR is usually unordered 3D point clouds, which can be mapped into an organized 2D range image by spherical projection as in [5]. Therefore, the projection data association can be employed to very efficiently perform millions of nearest neighborhood search tasks in parallel. Given a scanned point $p = (x, y, z)^T$, the mapping function from the Cartesian to its corresponding spherical coordinate is defined as follows:

$$\begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \arctan(y/x) \\ \arcsin\left(z/\sqrt{x^2+y^2+z^2}\right) \end{bmatrix}, \tag{3}$$

where r denotes the range. θ is the azimuth, and ϕ represents the elevation component. They are constrained by $r \geq 0$, $-\pi < \theta \leqslant \pi$, and $-\pi/2 < \phi \leqslant \pi/2$.

In this paper, the spherical range image is treated as a 2D search table $s(\theta, \phi)$, which stores the index of the Cartesian coordinates at azimuth θ and elevation ϕ. The final image coordinates (u, v) is converted by the spherical projection function

$\Pi_S : \mathbb{R}^3 \mapsto \mathbb{R}^2$,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \Pi_S(x, y, z) = \begin{bmatrix} \frac{1}{2}\left(1 - \frac{\theta}{\pi}\right) w_S \\ \left[1 - (\phi + f_{up}) f^{-1}\right] h_S \end{bmatrix}, \quad (4)$$

where $f = f_{up} + f_{down}$ is the vertical field-of-view of the LiDAR sensor. w_S and h_S are the width and height of SRI.

2) Non-Ground Cost E_S on Spherical Range Image: We deal with non-ground points using spherical range image by taking advantage of efficient projective data association. As in [5], the spherical range image is denoted as the vertex map $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$, and its corresponding normal map is $\mathcal{N}_D$. Drifting usually occurs in the frame-to-frame registration due to error accumulation. To alleviate this issue, we adopt a frame-to-model method, where the model is represented by the vertex map $\mathcal{V}_M$ and normal map $\mathcal{N}_M$. Thus, we minimize the point-to-plane error to estimate the relative pose change:

$$E_S(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u} \in \mathcal{V}_D} \left[\mathbf{n}_u^T \left(T_t^{t-1}\mathbf{u} - \mathbf{v}_u\right)\right]^2, \quad (5)$$

where each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to the model vertex $\mathbf{v}_u \in \mathcal{V}_M$ and its normal $\mathbf{n}_u \in \mathcal{N}_M$ via

$$\mathbf{v}_u = \mathcal{V}_M(\Pi_S(T_t^{t-1}\mathbf{u})), \quad (6)$$

$$\mathbf{n}_u = \mathcal{N}_M(\Pi_S(T_t^{t-1}\mathbf{u})). \quad (7)$$

The non-ground Jacobian J_S can be computed as follows:

$$J_S = \mathbf{n}_u^T \begin{bmatrix} I & [\mathbf{v}_u]_\times \end{bmatrix}, \quad (8)$$

where $[\mathbf{v}_u]_\times$ denotes the skew symmetric matrix of vector $\mathbf{v}_u$.

3) Normal Estimation by Eigen Decomposition: As shown in Eqn. (8), the accuracy of Jacobian is heavily dependent on normal estimation, which is also essential to building the correspondences and computing the residual. Therefore, surface normal is the key to precisely predict the pose update. The conventional methods [5], [18] simply compute the normal through the cross product of local image gradients that may not belong to the same local planar region. This incurs the large errors in normal estimation. To this end, we propose a novel robust range adaptive normal estimation with two outlier rejection criteria.

In general, a plane is defined by the equation $n_x x + n_y y + n_z z - d = 0$, where $(x, y, z)^T$ lies on the plane and (n_x, n_y, n_z, d) are the corresponding plane parameters. Given a subset of 3D points $\mathbf{p}_i$, $i = 1, 2, \ldots, k$ of the local surface, finding the optimal normal vector $\mathbf{n} = (n_x, n_y, n_z)$ is a least square minimization on error e as below:

$$e = \sum_{i=1}^{k} \left(\mathbf{p}_i^T \mathbf{n} - d\right)^2 \text{ subject to } |\mathbf{n}| = 1, \quad (9)$$

which has a closed-form solution by eigen-decomposition. The covariance matrix Σ explains the geometric information of the given local surface, which is defined as follows:

$$\Sigma = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T, \bar{\mathbf{p}} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{p}_i, \quad (10)$$

where $\bar{p}$ is the centroid of the point cloud subset. Using Eigen decomposition, $\Sigma \in R^{3\times3}$ can be decomposed into three eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, whose eigenvalues are in descending order $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The least eigenvalue λ_3 indicates the variations along the surface normal $\mathbf{n}$ that is equal to its eigenvector $\mathbf{v}_3$. Moreover, we employ the surface curvature $\sigma_{\mathbf{p}_i}$ at $\mathbf{p}_i$ to select the salience planar feature,

$$\sigma_{\mathbf{p}_i} = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}, \quad (11)$$

where we choose the point whose $\sigma_{\mathbf{p}_i}$ is below the curvature threshold δ_σ. We flip normal direction when the angle between normal vector and $\mathbf{p}_i$ is larger than 90°.

4) Range Adaptive Normal Estimation: With the spherical range image, we can visit a nearest neighbor with the time complexity of $O(1)$ for a small patch having the size of (l_x, l_y), whose center location $\mathbf{p}$ is determined by its (u, v) coordinates in range image. In the outdoor environment, the radius of LiDAR points have large variations within a scan, where the search window with the fixed size may not be effective. To account for the radius changes, we introduce a range adaptive searching window to select the nearest neighbors, in which the patch size is set according to the radius r, searching range δ_r and range image resolution (w_S, h_S):

$$\begin{pmatrix} l_x \\ l_y \end{pmatrix} = \begin{pmatrix} \max(\min(\frac{\delta_r}{r\pi} w_S, l_x^{\max}), l_x^{\min}) \\ \max(\min(\frac{\delta_r}{rf} h_S, l_y^{\max}), l_y^{\min}) \end{pmatrix}, \quad (12)$$

where $l_x^{\max}$, $l_y^{\max}$, $l_x^{\min}$, and $l_y^{\min}$ denote the maximum and minimum searching range, respectively.

Due to the discontinuities at boundaries and multiple reflectance, the decomposition of covariance matrix is very sensitive to outliers. To reject the large outliers, we suggest two effective criteria. A point $\mathbf{p}$ is considered as an outlier if the distances to half of points within its range adaptive window are larger than the threshold δ_d. Another criterion is based on the point-to-plane distance $d_{p2p} = (q - p)^T \cdot \mathbf{n}_p$. A point is marked as outlier when its point-to-plane distance d_{p2p} is large than the threshold δ_z, $d_{p2p} > \delta_z$.

C. Ground Cost E_G

We present the ground cost E_G by taking advantage of bird's-eye-view map.

1) Ground Segmentation: To efficiently obtain the ground segmentation, we employ a fast thresholding algorithm using the spherical range image. Since the height of LiDAR center with respect to road surface h_g can be estimated from calibration, the ground point candidates need satisfy the condition $h_p - h_g > \delta_{h_1}$ and $h_g - h_p > \delta_{h_2}$. h_p represents the height of candidate point. δ_{h_1} and δ_{h_2} are two vertical thresholds.

Pixels in the column v of spherical range image are regarded as lying in the same azimuth space. If two adjacent pixels are in ground, the angle change in height direction should be less than a small threshold δ_θ. A valid pixel at (u, v) in range image indexes a 3D point $\mathbf{p}$ in Cartesian coordinate. We find the first valid points up $\mathbf{p}_{up}$ and down $\mathbf{p}_{down}$ whose coordinates in range image are $(u, v + i)$ and $(u, v - j), i, j = 1, 2, 3, \ldots$. The angle change in two directions can be computed as below:

$$\theta_{up} = \arctan \frac{\lfloor \mathbf{p}_{up} - \mathbf{p} \rfloor_z}{\lfloor \mathbf{p}_{up} - \mathbf{p} \rfloor_{xy}}, \theta_{down} = \arctan \frac{\lfloor \mathbf{p}_{down} - \mathbf{p} \rfloor_z}{\lfloor \mathbf{p}_{down} - \mathbf{p} \rfloor_{xy}}, \quad (13)$$

where $\lfloor \cdot \rfloor_{(\cdot)}$ denotes the normal length in the specific direction. The ground points are selected by the angle changes below the

threshold $\theta_{up} < \delta_\theta$ and $\theta_{down} < \delta_\theta$. The remaining points are regarded as non-ground.

2) Bird's-Eye-View (BEV) Map: Similar to the spherical range image, bird's-eye-view projection also maps 3D Cartesian coordinate points into a 2D map. Specifically, each ground point $\mathbf{p} = (x, y, z)^T$ is converted into its corresponding coordinates (u, v) in BEV map by the projection function $\Pi_G : \mathbb{R}^3 \mapsto \mathbb{R}^2$ defined as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \Pi_G(x, y, z) = \begin{bmatrix} (x + w_B) \cdot s_x \\ (y + h_B) \cdot s_y \end{bmatrix}, \tag{14}$$

where w_B and h_B are the predefined scope in the two direction of BEV image. s_x and s_y represent the resolution of the BEV map in two directions.

In this paper, we suggest to make use of the 2D BEV map to effectively align the ground points of the current scan with the previous model map $\mathcal{B}_M$ through the projection data association. Current BEV vertex map $\mathcal{B}_G : \mathbb{R}^2 \mapsto \mathbb{R}^3$ can be treated as a 2D search table for the fast nearest neighbor search, where each pixel contains the 3D ground points. Let $\mathbf{n}_g$ denote the surface normal for ground point, the cost function E_G can be derived as below

$$E_G(\mathcal{B}_G, \mathcal{B}_M) = \sum_{\mathbf{g} \in \mathcal{B}_G} [\mathbf{n}_g^T (T_t^{t-1} \mathbf{g} - \mathbf{v}_g)]^2, \tag{15}$$

where each vertex $\mathbf{g} \in \mathcal{B}_G$ is projectively associated to a model vertex $\mathbf{v}_g \in \mathcal{B}_M$ via

$$\mathbf{v}_g = \mathcal{B}_M(\Pi_G(T_t^{t-1} \mathbf{g})), \tag{16}$$

The Jacobian J_G for the ground cost E_G can be computed by

$$J_G = \mathbf{n}_g^T \begin{bmatrix} I & [\mathbf{v}_g]_\times \end{bmatrix}, \tag{17}$$

where $[\mathbf{v}_g]_\times$ denotes the skew symmetric matrix of vector $\mathbf{v}_g$.

In contrast to the non-ground cost E_S, the surface normal $\mathbf{n}_g$ for ground point is not precomputed during the feature extraction stage. As the neighbors of ground points in a scan are usually far away from each other, the surface curvature of range image may not reflect the true geometry structure. Therefore, the normals of ground points are computed online while minimizing the ground cost $E_G(\mathcal{B}_G, \mathcal{B}_M)$. After projecting the current ground point $\mathbf{g}$ onto the previous local map $\mathbf{v}_g$, we retrieve the five closest points of $\mathbf{v}_g$ to calculate the surface normal in the BEV model map. By making use of the 2D BEV image, it is very efficient to find the candidate points in a predefined local patch on GPU.

D. Model Update Scheme

Differently from the conventional data structure like voxel grid and KD-Trees, we suggest a very fast and memory efficient model update scheme that takes advantage of 2D spherical range image and ground BEV map. More importantly, it can be easily implemented in parallel.

In general, it can be clearly observed that the sampled data of laser beam is hardly to be the same point in the successive scans. Moreover, the LiDAR points are very sparse so that the surface features are inadequate within one scan. Therefore, the correspondences between the previous frame and current scan are usually not the ideal matches, where the errors will be accumulated. This may lead to the drift of vehicle pose estimation inevitably. To deal with this issue, we employ the frame-to-model scheme to fuse the consecutive scans.

Once the pose change T_t^{t-1} is estimated, we suggest two different updating strategies for the non-ground spherical range image and ground BEV map, respectively. As for non-ground point cloud, we maintain the model vertex map $\mathcal{V}_M$ and normal map $\mathcal{N}_M$. Given a valid pixel with the 3D position $\mathbf{p}$ in the previous spherical range image, we transform it by T_t^{t-1}, which is further projected into current image coordinate as pixel (u, v). If the projected pixel is valid in the current vertex map $\mathcal{V}_D$, we compare the distance between the point $\mathbf{p}_s$ in current scan and the previous map point $\mathbf{p}_m$. The point close to LiDAR origin is chosen as $\min(\|\mathbf{p}_s\|_2, \|\mathbf{p}_m\|_2)$. Then, we update both vertex map $\mathcal{V}_M$ and normal map $\mathcal{N}_M$ with the selected point. If the location is not occupied, we simply cover it by the point in the previous map. For the ground points, we only maintain the BEV vertex model map $\mathcal{B}_M$ while the normal is computed with the five nearest points during ICP optimization. Similar to non-ground points, we just need to replace the spherical projection function Π_S with the BEV projection function Π_G in the updating process of ground model. We only need keep three 2D maps for our proposed model update scheme, which is highly memory efficient. The size of three model maps are same as frame data.

To reduce the influence of sensor noises, we record the timestamp of each point. Since the measurement noise of point cloud is highly related to its distance from the sensor, those long-range points have the high uncertainty. Once the vehicle travels forward, the early recorded points may introduce large errors in the pose estimation process. Therefore, we discard the old points whose timestamps exceed the time window τ_w, $t_c - t_o > \tau_w$. t_c denotes the current timestamp, and t_o is the observed timestamp in the model.

IV. Experiment

In this section, we present details of our experiments and discuss the results on LiDAR odometry. We test how effective the proposed approach is on the driving dataset and system sensitivity on different parameter settings. Additionally, we compare our presented method against the recent state-of-the-art approaches and evaluate on computational time.

A. Experimental Testbed

To investigate the efficacy of our proposed LiDAR odometry approach, we conduct the experiments on the KITTI odometry benchmark, where the 3D point scans are collected from the Velodyne HDL-64E S2. The whole dataset contains a wide variety of scenarios from urban city to highway traffic. There are 11 sequences (00-10) provided with GNSS-INS poses as the ground-truth for training, while another 11 sequences (11-21) without the ground truth for online evaluation on the leaderboard[1]. As in [7], a vertical angle of 0.195 ° is employed to correct the calibration errors in raw point clouds from the KITTI dataset. The average relative translation error t_{rel} (%) and rotation error r_{rel} (deg/100 m) are adopted as the performance metrics. In Table II, Table III and Table IV, the first and second row of each method are t_{rel} and r_{rel}. To examine the

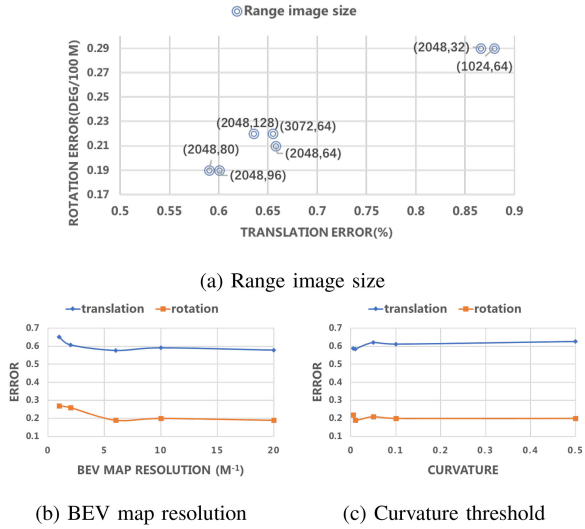
[1]http://www.cvlibs.net/datasets/kitti/eval_odometry.php

TABLE I
PARAMETER SETTINGS OF OUR METHOD

Parameter	value	Parameter	value
f_{up}, f_{down} Velodyne 64	$3^\circ, 25^\circ$	$\delta_{h_1}, \delta_{h_2}$	0.5 m, 0.1 m
f_{up}, f_{down} Velodyne 32	$10^\circ, 30^\circ$	δ_d	0.5 m
f_{up}, f_{down} Robosense 32	$15^\circ, 25^\circ$	δ_σ	0.01
$w_S \times h_S$ KITTI	2048×80	δ_r	0.3 m
$w_S \times h_S$ UrbanLoco	1024×40	δ_z	0.5
$w_B \times h_B$	$120\ m \times 60\ m$	w_1	0.7
s_x, s_y	$10\ m^{-1}$	δ_θ	0.5°
$l_x^{\max}, l_y^{\max}$	13, 7	τ_w	10 s
$l_x^{\min}, l_y^{\min}$	5, 3		

TABLE II
PERFORMANCE EVALUATION ON DIFFERENT PROJECTION METHODS

Approach	00	01	02	03	04	05	06	07	08	09	10
Spherical	0.64	0.73	0.59	0.93	0.45	0.48	0.33	0.56	0.87	0.56	1.42
(all points)	0.28	0.17	0.22	0.26	0.42	0.27	0.14	0.35	0.26	0.17	0.61
Spherical	0.71	0.80	0.61	0.87	0.52	0.52	0.33	0.54	0.93	0.59	0.85
(non-ground)	0.34	0.20	0.24	0.22	0.49	0.31	0.23	0.39	0.30	0.21	0.47
BEV map	0.76	0.66	43.02	64.46	0.27	5.07	12.60	118.90	1.03	127.73	165.00
(ground)	0.33	0.17	15.08	43.34	0.20	10.90	0.40	42.52	0.34	17.84	40.97
Fusion	**0.54**	**0.61**	**0.54**	**0.65**	**0.32**	**0.33**	**0.30**	**0.31**	**0.79**	**0.48**	**0.59**
(all points)	**0.20**	**0.13**	**0.18**	**0.27**	**0.15**	**0.17**	**0.13**	**0.16**	**0.21**	**0.14**	**0.19**



(a) Range image size

(b) BEV map resolution

(c) Curvature threshold

Fig. 3. Parameter sensitive analysis on sequence 10 of KITTI. The remaining parameters are kept the same as Table I.

generalization capability, we compare our proposed approach with the previous methods on the UrbanLoco dataset [9]. which is collected in Hong Kong by Velodyne HDL-32E and California by RoboSense RS-LiDAR-32 having drastically different landscapes, driving behaviors, architectures, and infrastructures.

Our method is implemented by C++ with CUDA. All the experiments are performed on a laptop computer having an Intel Core i7-9750H CPU@2.60 GHz with 16 GB RAM and an NVIDIA GeForce RTX 2060 GPU with 6 GB RAM. Additionally, we evaluate the computational time of our proposed approach on NVIDIA Jetson AGX that is a popular embedded device for autonomous vehicles. We set the parameters of our method empirically based on the KITTI training dataset, as shown in Table I.

B. Evaluation on Different Projection Methods

We firstly examine the performance on the different projection methods, including spherical range image, bird's-eye-view map, and our proposed fusion approach. To facilitate the fair comparison, all the methods employ our presented range adaptive approach to estimate the normals.

As shown in Table II, it can be clearly seen that our proposed fusion approach obtains the best results on the KITTI training dataset, which indicates that the ground information is essential to the challenging scenario for LiDAR odometry, such as highway. Moreover, the method like [5] directly using spherical range image performs similar to the one with non-ground points alone. This implies that the non-ground points are effective enough to achieve the reasonable odometry results while the ground information is not full utilized in the spherical projection. Additionally, it can be observed that drifting occurs at several sequences with BEV map for the ground points. This is because the normal of surfaces perpendicular to laser beams cannot be accurately estimated using BEV map. It leads to the large rotation errors so that large drifts exist in the sequences with lots of quarter turns. Surprisingly, the results of BEV map outperforms the spherical range image method in the challenging case of highway on Sequence 01, where the ground region occupies large portion of scans. Therefore, the reliability of LiDAR odometry can be greatly improved by imposing the ground constraints. It can be concluded that our proposed fusion approach can fully exploit most of the local surfaces in the driving scenario.

C. Evaluation on Normal Estimation

We study the odometry results with the different normal estimation methods, including cross product of two point pairs, eigen-decomposition, and our proposed range adaptive approach. As shown in Table IV, our presented range adaptive method consistently outperforms both cross product and plain eigen-decomposition at a very large margin. Moreover, the cross product method suffers from drifting issues in several sequences, since the adjacent points on the spherical range image have great potential lying on the different planes. Furthermore, the odometry results are improved a lot by the eigen-decomposition method.

Parameters in our methods are mostly related to distance, angular, and time threshold, which are commonly used in LiDAR odometry. As range image size, BEV map resolution and curvature threshold are three distinctive parameters in our proposed method, we investigate the sensitivity of these parameters on the KITTI sequence 10. Fig. 3 shows the experimental results. It can be seen that range image size is important to LiDAR odometry, which is an intrinsic parameter depend on the number of laser beams, vertical and horizon angular resolution. For Velodyne HDL-64 used in KITTI, the range image of 2048×80 obtains the best result. Although large BEV map resolution can improve the trajectory accuracy, the gain is limited with the resolution higher than $0.1m$. Curvature threshold plays an important role in choosing the salient planar features. The results are nearly same when the threshold changes. This indicates that our proposed outlier rejection criteria is effective, which makes the system robust in different settings.

D. Comparison With State-of-The-Art Methods

To further examine the performance of LiDAR odometry, we compare our proposed method with the state-of-the-art

TABLE III
RESULTS ON KITTI ODOMETRY BENCHMARK

Approach	00	01	02	03	04	05	06	07	08	09	10	Average	Online mean	Speed(s)
Frame-to-Frame	1.14	0.83	0.91	1.25	0.83	0.75	0.76	0.60	1.34	1.18	2.57	1.11	-	**0.005**
	0.52	0.28	0.36	0.59	0.51	0.43	0.51	0.48	0.52	0.43	0.84	0.50	-	
Frame-to-Model	0.54	**0.61**	0.54	**0.65**	**0.32**	0.33	**0.30**	**0.31**	**0.79**	0.48	0.59	**0.50**	**0.68**	0.006
	0.20	**0.13**	**0.18**	**0.27**	**0.15**	**0.17**	**0.13**	**0.16**	**0.21**	**0.14**	**0.19**	**0.18**	0.21	
Jetson	0.55	0.64	0.56	0.70	0.42	0.33	0.32	0.33	0.80	0.47	0.65	0.52	-	0.05
AGX	0.20	0.12	0.18	0.23	0.42	0.17	0.14	0.17	0.23	0.13	0.21	0.20	-	
SuMa [5]	2.10	4.00	2.30	1.40	11.90	1.50	1.00	1.80	2.50	1.90	1.80	2.93	-	-
Frame-to-Frame	0.90	1.20	0.80	0.70	1.10	0.80	0.60	1.20	1.00	0.80	1.00	0.92	-	
SuMa [5]	0.70	1.70	1.10	0.70	0.40	0.50	0.40	0.40	1.00	0.50	0.70	0.74	1.39	0.10
Frame-to-Model	0.30	0.30	0.40	0.50	0.30	0.20	0.20	0.30	0.40	0.30	0.30	0.34	0.34	
SuMa++	0.64	1.60	1.00	0.67	0.37	0.40	0.46	0.34	1.10	**0.47**	0.66	0.70	1.06	0.10
[19]	0.22	0.46	0.37	0.46	0.26	0.20	0.21	0.19	0.35	0.23	0.28	0.29	0.34	
LOAM	0.78	1.43	0.92	0.86	0.71	0.57	0.65	0.63	1.12	0.77	0.79	0.84	0.88	0.10
[6]	-	-	-	-	-	-	-	-	-	-	-	-	-	
IMLS-SLAM	**0.50**	0.82	**0.53**	0.68	0.33	**0.32**	0.33	0.33	0.80	0.55	**0.53**	0.52	0.69	1.25
[7]	-	-	-	-	-	-	-	-	-	-	-	-	**0.18**	
LeGO-LOAM	2.17	13.4	2.17	2.34	1.27	1.28	1.06	1.12	1.99	1.97	2.21	2.49	-	-
[23]	1.05	1.02	1.01	1.18	1.01	0.74	0.63	0.81	0.94	0.98	0.92	1.00	-	-

TABLE IV
PERFORMANCE EVALUATION ON NORMAL ESTIMATION

	00	01	02	03	04	05	06	07	08	09	10
Cross	8.23	1.32	0.77	0.78	1.74	0.72	0.32	0.57	0.91	0.78	2.74
Product	3.66	0.31	0.32	0.33	0.41	0.34	0.21	0.35	0.35	0.27	1.17
Eigen	0.85	0.98	0.97	1.62	1.33	0.62	0.59	0.80	1.42	1.03	1.39
Decomp.	0.27	0.13	0.25	0.32	0.17	0.21	0.19	0.31	0.28	0.25	0.42
Range	**0.54**	**0.61**	**0.54**	**0.65**	**0.32**	**0.33**	**0.30**	**0.31**	**0.79**	**0.48**	**0.59**
Adaptive	**0.20**	**0.13**	**0.18**	**0.27**	**0.15**	**0.17**	**0.13**	**0.16**	**0.21**	**0.14**	**0.19**

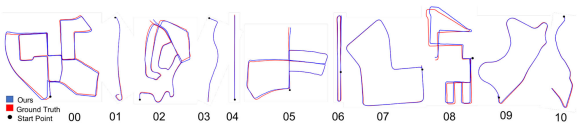


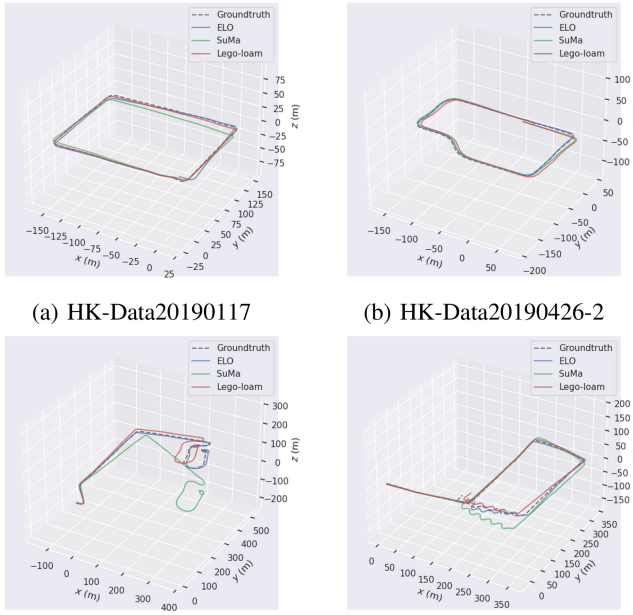
Fig. 4. Camera path for our proposed method on KITTI odoemtry dataset.

approaches. For LOAM [6], IMLS [7], SuMa [5], SuMa++ [19], we directly include their results reported in the papers. Since there are no published results for Lego-LOAM [23] on KITTI dataset, we conducted the experiments using their own implementation.

Table III shows the experimental results. It can be observed that our proposed approach achieves the best results with 0.50% drift in translation and 0.0018 deg/m error in rotation on the KITTI training dataset. Fig. 4 plots the trajectory of our proposed method. Moreover, we obtain 0.68% drift in translation and a 0.0021 deg/m error in rotation on the KITTI testing dataset, which outperforms IMLS with 0.69% error in translation. Note that IMLS is the state-of-the-art LiDAR odometry method, which performs the best among the published results. Comparing to the spherical range image-based methods [5], [19], the proposed approach significantly improves the trajectory accuracy on KITTI benchmark by taking advantage of our presented projection fusion method and range adaptive normal estimation scheme. In contrast to the conventional two-step ground optimization method like LeGO-LOAM [23], our proposed bird's-eye-view map is very effective to capture the ground information.

In addition, we compare our proposed approach against two methods on the UrbanLoco dataset using their own implementation, including SuMa[2] and LeGO-LOAM[3]. To facilitate fair comparison, all the parameter settings are kept the same as the KITTI dataset except for the size of range image. Since the UrbanLoco dataset was collect by two 32-line LiDARs, we reduce the range image size from 2048×80 to 1024×40. Meanwhile, parameter settings in SuMa and LeGO-LOAM are adjusted for corresponding LiDAR. Fig. 5 plots the trajectories aligned to GPS data, and Table V shows the average absolute trajectory errors. It can be clearly seen that our approach obtains the best results especially on two elevation-changeable CA sequences.

[2]https://github.com/jbehley/SuMa
[3]https://github.com/RobustFieldAutonomyLab/LeGO-LOAM



(a) HK-Data20190117 (b) HK-Data20190426-2

(c) CAColiTower (d) CALombardStreet

Fig. 5. Comparisons on the UrbanLoco dataset

TABLE V
ABSOLUTE TRAJECTORY ERROR ON URBANLOCO DATASET

	Ours	SuMa [5]	LeGO-LOAM [23]
HK-Data20190117	**4.78**	5.04	6.78
HK-Data20190426-2	**4.50**	4.74	14.17
CAColiTower	**6.67**	104.81	75.30
CALombardStreet	**14.78**	36.31	29.57

TABLE VI
EVALUATION ON COMPUTATIONAL TIME (MS)

Approach	Spherical Projection	Feature Extraction	ICP Optimization	Model Updating	Total
PC frame2frame	0.8	1.5	1.6	0.3	4.2
PC frame2model	0.9	1.5	2.1	1.3	5.9
AGX frame2model	2.9	5.3	11.2	7.4	26.8

E. Evaluation on Computational Efficiency

The complexity of LiDAR odometry is mainly dominated by the nearest neighbor search and normal computation. To demonstrate the efficiency of our method, we evaluate the computational cost on different steps including spherical projection, feature extraction, ICP optimization, and model updating. Feature extraction is made of ground segmentation and non-ground normal estimation. The spherical projection is implemented on CPU with OpenMP, and the remaining steps are computed on GPU. Additionally, we implement our proposed approach on Nvidia Jetson AGX, which is commonly used in autonomous vehicles.

As depicted in Table VI, the proposed approach runs about 169 frames per second on the commodity laptop computer while IMLS [7] requires 1.25 s to process single scan. Moreover, the conventional spherical projection method [5] runs around 15 Hz. This indicates that the presented scheme is not only very effective but also an order of magnitude faster than the conventional methods due to the efficient parallel implementation. It could be even faster with the less accurate frame-to-frame optimization, which runs at 238 frames per second. Finally, the presented method only requires 27 milliseconds on NVIDIA Jetson AGX, which could be potentially a key component for the autonomous vehicles.

V. CONCLUSION

This paper proposed a novel efficient LiDAR odometry approach, which takes into account of both non-ground spherical range image and ground bird's-eye-view map. Moreover, the range adaptive method was introduced to robustly estimate the local surface normal. Additionally, we suggested an memory-efficient model update scheme to fuse the points and their corresponding normals at different time-stamps. We have conducted extensive evaluations on KITTI odometry benchmark, whose promising results demonstrated that our proposed approach not only outperforms the state-of-the-art LiDAR odometry methods but also runs over 169 frames per second on a commodity laptop computer.

Despite these encouraging results, some limitations and future work should be addressed. Currently, our method only takes considerations of LiDAR data. Besides, we have yet included the loop closure detection. For future work, we will address these issues by incorporating dense image alignment and backend optimization.

REFERENCES

[1] 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.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[3] J. Zhu, "Image gradient-based joint direct visual odometry for stereo camera," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 4558–4564.

[4] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[5] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Robot.: Sci. Syst.*, vol. 2018, 2018.

[6] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robot.: Sci. Syst.*, vol. 2, no. 9, 2014.

[7] J.-E. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2480–2485.

[8] 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.

[9] W. Wen *et al.*, "Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2310–2316.

[10] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss, "Adaptive robust kernels for non-linear least squares problems," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2240–2247, Apr. 2021.

[11] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *Int. Conf. Comput. Vision Theory Applications*, vol. 1, pp. 331–340, 2009.

[12] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 19–25.

[13] K.-L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," Chapel Hill, University of North Carolina, vol. 4, no. 10, pp. 1–3, 2004.

[14] J. Demantké, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3D lidar point clouds," in *ISPRS-Int. Archives Photogrammetry, Remote Sensing Spatial Inf. Sciences*, vol. 3812, pp. 97–102, 2011.

[15] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst(Cat. No 03CH37453)*, vol. 3, 2003, pp. 2743–2748.

[16] C. Ulaş and H. Temeltaş, "3D multi-layered normal distribution transform for fast and long range scan matching," *J. Intell. Robotic Syst.*, vol. 71, no. 1, pp. 85–108, 2013.

[17] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robot.: Sci. Syst.*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[18] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," *Robot.: Sci. Syst.*, 2015.

[19] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.

[20] A. Petrovskaya and S. Thrun, "Model based vehicle tracking for autonomous driving in urban environments," *Proc. Robot.: Sci. Syst.*, Zurich, Switzerland, vol. 34, 2008.

[21] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 215–220.

[22] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 163–169.

[23] 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.