Point Wise or Feature Wise? A Benchmark Comparison of Publicly Available Lidar Odometry Algorithms in Urban Canyons

Feng Huang, Weisong Wen, Jiachen Zhang, and Li-Ta Hsu
Are with the Department of Aeronautical and Aviation Engineering, Hong Kong Polytechnic University, Hong Kong, QR828, China. E-mail: darren-f.huang@connect.polyu.hk; welson.wen@polyu.edu.hk; lt.hsu@polyu.edu.hk; jiachen.zhang@polyu.edu.hk

Digital Object Identifier 10.1109/MITS.2021.3092731
Date of current version: 1 April 2022

Abstract—Robust and precise localization is essential for an autonomous system with navigation requirements. Lidar odometry (LO) has been extensively studied in the past decades to realize this goal. Satisfactory accuracy can be achieved in scenarios with abundant environmental features using existing LO algorithms. Unfortunately, the performance of the LO is significantly degraded in urban canyons with numerous dynamic objects and complex environmental structures. Meanwhile, it is still not clear from the existing literature which LO algorithms perform well in such challenging environments. To fill this gap, this article evaluates an array of popular and extensively studied LO pipelines using the data sets collected in urban canyons of Hong Kong. We present the results in terms of their positioning accuracy and computational efficiency. The three major factors dominating the performance of LO in urban canyons are concluded, including the ego-vehicle dynamic, moving objects, and the degree of urbanization. According to our experiment results, point wise accomplishes better accuracy in urban canyons while feature-wise achieves cost-efficiency and satisfactory positioning accuracy.

Introduction

Positioning is one of the major challenges preventing the arrival of fully mobile autonomous systems, such as autonomous driving vehicles [1], autonomous mobile service robots [2], and autonomous aerial robots [3], [4]. The global navigation satellite system (GNSS) is popular for providing globally referenced positioning services, unfortunately, its performance relies highly on satellite visibility. Satisfactory accuracy can be obtained in open areas with centimeter-level accuracy using GNSS real-time kinematic (RTK). However, the positioning error can significantly deteriorate in urban canyons with high-rise buildings due to nonline of sight (NLOS) and multipath effects [5]. To mitigate the impacts of the errors caused by the multipath and NLOS receptions, much research has been proposed to exclude [6], [7] and correct [8], [9] the affected GNSS raw measurements to improve the GNSS solution in urban canyons. However, the derived positioning accuracy is still far from the navigation requirements of fully mobile autonomous systems.

3D lidar, which provides dense 3D point clouds of the surroundings, has recently been widely studied to provide accurate and high-frequency lidar odometry (LO) [10] for autonomous systems. The LO method estimates the pose of the autonomous system by accumulating the transformation among consecutive frames of 3D point clouds. Therefore, the accuracy of LO relies heavily on the performance of point cloud registration, which derives the motion between among frames of 3D point clouds. Numerous methods have been studied to perform point cloud registration, which can be mainly divided into feature-wise methods, e.g., lidar odometry and mapping (LOAM) [11] and point-wise methods, such as the normal distribution transform (NDT) [12]. Satisfactory accuracy can be achieved in the widely evaluated KITTI data set [1], with abundant environmental features and limited moving objects. However, according to our previous findings in [13] and [14], an NDT-based LO's performance is significantly degraded in challenging urban canyons with numerous moving objects. This is because the existing LO's pipelines rely on the assumption that the environmental features are static. In this article, we aim to present a benchmark of various LO algorithms in the highly urbanized environment for the research community as well as answering a question: What dominates the performance of LO in dynamic urban canyons? The contributions of this article are summarized as follows:

- This article presents a comprehensive performance evaluation and analysis of the publicly available LO pipelines listed in Table 1 using the challenging data sets collected in typical urban canyons of Hong Kong.
- This article concludes with three dominant factors, which degrade the performance of LO algorithms in urban canyons, including the motion difference, dynamic objects, and the degree of urbanization.

Related Works

Based on how the raw point clouds are modeled, the available LO methods can be separated into two groups: the point- and feature-wise methods. The point-wise method estimates the relative transformation based on the raw points. Differently, the feature-wise methods [11] extract representative features, such as the edge and planar features from raw point clouds. In other words, the classification of the point- and feature-wise methods are based on the assumption that all the raw points are directly utilized in further data association. The publicly available LO methods and their major properties are listed in Table 1. The following sections review these LO algorithms in detail.

Point-Wise-Based LO

The most well-known point-wise LO method is iterative closest point (ICP) [15]. ICP is a modular and straightforward algorithm that directly matches two frames of the point cloud by finding the correspondences at a point-wise level. One of the major drawbacks of ICP is the high computational load arising from the point cloud registration when coping with dense clouds. The performance of ICP relies on the initial guess due to the nonconvexity of ICP optimization [25]. Moreover, the unexpected outlier

The global navigation satellite system is popular for providing globally referenced positioning services, unfortunately, its performance relies highly on satellite visibility.

measurements occurring from moving objects can even introduce additional nonconvexity. Many variants of ICP have been proposed to improve its efficiency and accuracy, such as trimmed [26] and normal ICP [27]. Among these ICP variants, generalized-ICP (G-ICP) [16] is the most popular variant, with distinct accuracy. Instead of formulating the cost function directly by finding point-wise correspondence, the G-ICP, which benefits from standard ICP and the point-to-plane method introduced by Chen and Medioni [28], optimizes the transformation via a distribution-to-distribution fashion. Compared with conventional ICP, G-ICP explores the geometry correlation with points via distribution modeling. As a result, G-ICP is less sensitive to the initial guess compared with typical ICP. However, although G-ICP is known for its accuracy, the optimization of G-ICP shares the same drawback with ICP and its variants of relying on the time-consuming nearest neighbor search (NNS). According to the work in [17], the NNS dominates the computation efficiency of the G-ICP method.

Rather than relying on the time-consuming NNS process, the NDT [29], which is another major point-wise registration method, employs a voxel-based correspondence association to estimate the transformation. Given a set of raw and discrete point clouds, the NDT divides the point clouds into multiple voxels. Then, each voxel is modeled using a Gaussian distribution. The cost function for estimating the transformation is established by mapping the other point clouds to the voxelized one. Overall, the NDT is much faster than the typical ICP, and according to an interesting performance comparison work in [30], the NDT outperforms ICP in terms of the accuracy in the evaluated data set. However, the performance of the NDT is sensitive to the selection of the resolution of voxels. In other words, carefully tuning the voxel size is needed based on the sensor and environment conditions. Meanwhile, although the NDT does not rely on the time-consuming NNS, the real-time performance still cannot be guaranteed when coping with large amounts of dense point clouds.

Fortunately, the team from the Toyohashi University of Technology conducted continuous studies on accelerating the computational efficiency of G-ICP [17] and the NDT [18]. The work in [18] proposed, and multiple threads accelerated the NDT, namely, the OpenMP-boosted NDT method (NDT-OMP), for efficient point cloud registration, leading to real-time performance. Meanwhile, the continuous work in [17] proposed a voxelized G-ICP (VGICP) algorithm, which was developed on top of the conventional G-ICP, leading to real-time performance. A novel voxelization approach of VGICP enables parallel implementation by aggregating the distribution of all the points in a voxel. According to the evaluation of [17], similar or

Table 1. An overview of the Selected Publicly available LO methods evaluated in this article.

Style	Method	Modeling*	Optimization	Year
Point wise	ICP [15]	Minimize the point-to-point distance	s2s, SVD (the PCL version)	1992
	G-ICP [16]	Surface-based distribution to distribution	s2s, BFGS, and FLANN (the PCL version)	2009
	FastG-ICP [17]	Surface-based distribution to distribution	s2s, Gauss–Newton, and multithreaded	2020
	VGICP [17]	Voxel-based distribution to multidistribution	s2s, Gauss–Newton, and multithreaded	2020
	FastVGICPCuda [17]	Voxel-based distribution to multidistribution	s2s, Gauss–Newton, and CUDA optimized	2020
	NDT [12]	Voxel-based point to distribution	s2s, Newton method	2003
	NDT-OMP [18]	Voxel-based point to distribution	s2s, Newton method	2019
Feature wise	LOAM [11]	Minimize the distance of the feature points	s2s and s2m, LM	2014
	A-LOAM [19]	Minimize the distance of the feature points	s2s and s2m, Ceres Solver [20]	2018
	LeGO-LOAM [21]	Minimize the distance of the feature points, ground optimization	s2s and s2m, LM	2018
	LIO-mapping [22]	Minimize the distance of the feature points	s2s and s2m, LM	2019
	Fast LOAM [23]	Minimize the distance of the feature points	s2m, Ceres Solver [20]	2020

**Modeling* indicates how to model the transformation function. s2s: scan to scan; s2m: scan to map (also known as *scan to model*); PCL: Point Cloud Library; SVD: singular value decomposition; BFGS: Broyden–Fletcher–Goldfarb–Shanno; FLANN: Fast Library for Approximate Nearest Neighbor; CUDA: Compute Unified Device Architecture; LM: Levenberg–Marquardt [24].

The recent progress in neural networks is another promising solution that improves the performance of LO methods by deeply exploring the features within point clouds.

even better performance is obtained using the VGICP, compared with the G-ICP.

In short, G-ICP is one of the most popular and accurate variants. The VGICP relaxed the drawback of the high computation load of G-ICP arising from the NNS. The NDT is another popular point-wise method, and the NDT-OMP enables the real-time implementation of conventional NDT.

Feature-Based LO

Instead of directly estimating the transformation using all of the raw points, the feature-based LO methods [11], [21], [31], [32] extract a set of representative features from the raw points. The fast point feature histogram (FPFH) was proposed in [31] to extract and describe the features inside the raw points. The FPFH enables the exploration of the local geometry, and the transformation is optimized by matching the one-by-one FPFH-based correspondence. A similar descriptor, the Signature of Histograms of OrienTations (SHOT), is studied in [32] to extract the features. The other works in [33]–[35] optimize the ground-surface features, using multiple roadside lidars. As the feature-based LO methods extract only a limited set of features, which is significantly fewer than the inputted raw point clouds, the computational load for the matching is significantly lower than the optimization for G-ICP and the NDT. Meanwhile, the feature-based method is less sensitive to the initial guess, compared with the point-wise methods; however, it relies on the accuracy of the feature detection, and misdetection can lead to incorrect feature association. Theoretically, the feature-based method makes use of only a part of the raw points, the convergence accuracy [17] is worse than that of the point-wise LO methods.

Another well-known feature-based LO method is LOAM [11], which was first introduced by a team from Carnegie Mellon University in 2014. Theoretically, LOAM involves the properties of both the point- and feature-wise method, but in this article, we classify LOAM as a part of the feature-wise group. On the one hand, to decrease the computation load of a typical ICP, LOAM proposes extracting two kinds of features, the edge and the planar. The extraction of the feature is simply based on the smoothness of a small region near a given feature point. Different from the FPFH or SHOT, which could provide multiple categories of features based on their descriptors, LOAM involves only two feature groups. Even so, LOAM shares the advantage of feature-based LO, which does not use all of the raw points, leading to efficient registration. On the other hand, LOAM matches the extracted edge features to the previously maintained dense-edge feature map. Similarly, the extracted planar features are matched with the previously maintained dense planar feature map, leading to a scan-to-map matching, which is different from the typical ICP (scan-to-scan matching). Interestingly, the matching is evaluated by the Euclidean distance between features, instead of the descriptor's smoothness, which is the same as the point-wise registration. This can significantly decrease the mis-association of features and increase the robustness of the matching. Therefore, LOAM algorithms share the advantage of low-computation load from the feature-wise method and the robustness of the point-wise method. Due to the superiority of LOAM, it is ranked 2nd according to the evaluation from the KITTI benchmark among the evaluated methods (as of Nov 2020), the average translation error and rotation error is 0.57% and 0.0013 (°/m), respectively, for all the possible subsequences of (100, 200, ..., 800) in the 3D coordinates [36].

Although the performance evaluation among the KITTI data sets is prominent, the accuracy is not guaranteed in the diverse-operation scenarios [19], [21], [23]. The work in [21] argues that LOAM can drift significantly in altitude direction due to its limited features. Lightweight and ground optimized (LeGO)-LOAM is proposed to optimize the altitude direction of the state based on the detected ground points, and the drift is slightly decreased in the evaluated data set. An advanced implementation of LOAM (A-LOAM) is proposed in [19] to accelerate the computational efficiency of LOAM by replacing the complicated derivation of Jacobian with the state-of-the-art nonlinear optimization solver, Ceres Solver [20].

Different from LOAM, Fast LOAM is presented in [23], which is a fast implementation of LOAM. Interestingly, the odometry process is removed in Fast LOAM and only the mapping process is employed to estimate the transformation. The work in [22] presents an implementation of LOAM, i.e., a rotation-constrained refinement algorithm (LIO-mapping), where the parameters for feature selection and extraction are carefully tuned. However, although the variants of LOAM are studied gradually, the variants did not essentially improve LOAM. As mentioned previously, LOAM still relies on the availability of the features (edge and planar). Meanwhile, only a portion of the raw point clouds is used. As argued by the authors in [17], the combination of the feature-wise LO (e.g., LOAM) with the point-wise LO (e.g., VGICP) is a promising solution to guarantee both the accuracy and robustness, where the point-wise fine registration is performed after a coarse feature-wise registration. In short, the LOAM algorithms have dominated the existing LO methods since 2015 in the KITTI data set due to the following two reasons:

1) The edge and planner feature description and their matching guarantees both real-time performance and accuracy.
2) The scan-to-map matching among the features from the key frame and the historical dense feature map make it easier to find the correspondence accurately.

The recent progress [37], [38] in neural networks (NNs) is another promising solution that improves the performance of LO methods by deeply exploring the features within point clouds. A study was proposed in [39] to encode the sparse 3D point cloud to dense images and then perform the convolutional NN to LO odometry estimation. To tackle dynamic objects in scenes, LO-Net [40] was proposed with mask-weighted geometric constraint loss, which achieved results similar to LOAM. For generality of the learning-based method, an unsupervised LO method [41] based on the geometric consistency of point clouds was proposed by the team from the Korea Advanced Institute of Science and Technology in 2020. However, as these methods are not available in open source and additional training is required, we do not compare them in this article.

Brief Summary

Both the point- and feature-wise registration methods have their pros and cons. The LO listed in Table 1 will be theoretically compared and experimentally evaluated in the following sections.

Theoretical Comparisons of Different LO Pipelines

In the following sections, we present the key theory of the listed LO methods. To make a clear comparison, the commonly used notations are summarized in Table 2. The matrices are denoted as uppercase bold letters, vectors are denoted as lowercase bold letters, variable scalars are denoted as lowercase italic letters, and constant scalars are denoted as lowercase letters. In the "Summary" section, we summarize their theoretical differences in Table 3.

ICP

The major principle of ICP is to calculate the transformation between the source and target point clouds by minimizing the error of the point-to-point distances [15]. Figure 1 shows an example of the registration between two consecutive frames of point clouds. The general cost function of ICP is

$$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N} \left\| \mathbf{x}_{(k,i)} - (\mathbf{R}_{k+1}^{k} \mathbf{x}_{(k+1,i)} + \mathbf{t}_{k+1}^{k}) \right\|^2. \quad (1)$$

First, the ICP calculates the centroid of the point cloud $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{x}}_{k+1}$.

$$\bar{\mathbf{x}}_{(k,i)} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_{(k,i)}. \quad (2)$$

Then, subtract the center of mass from the consecutive point sets to obtain the decentering point set $\mathbf{x}'_{(k,i)} \in \mathbf{X}'_k$ and $\mathbf{x}'_{(k+1,i)} \in \mathbf{X}'_{k+1}$, corresponding to $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$, respectively [15],

$$\mathbf{x}'_{(k,i)} = \mathbf{x}_{(k,i)} - \bar{\mathbf{x}}_k. \quad (3)$$

Modifying (1) based on (2) and (3), we can have

$$\begin{aligned} &\frac{1}{2} \sum_{i=1}^{N} \left\| \mathbf{x}_{(k,i)} - (\mathbf{R}_{k+1}^{k} \mathbf{x}_{(k+1,i)} + \mathbf{t}_{k+1}^{k}) \right\|^2 \\ &\quad = \frac{1}{2} \sum_{i=1}^{N} \left\| \mathbf{x}_{(k,i)} - (\mathbf{R}_{k+1}^{k} \mathbf{x}_{(k+1,i)} + \mathbf{t}_{k+1}^{k}) - \bar{\mathbf{x}}_k \right. \\ &\qquad \left. + \mathbf{R}_{k+1}^{k} \bar{\mathbf{x}}_{k+1} + \bar{\mathbf{x}}_k - \mathbf{R}_{k+1}^{k} \bar{\mathbf{x}}_{k+1} \right\|^2. \end{aligned} \quad (4)$$

Then the right side of (4) can be simplified to

$$\begin{aligned} &\frac{1}{2} \sum_{i=1}^{N} \left\| (\mathbf{x}_{(k,i)} - \bar{\mathbf{x}}_k) - (\mathbf{R}_{k+1}^{k} (\mathbf{x}_{(k+1,i)} - \bar{\mathbf{x}}_{k+1})) \right\|^2 \\ &\quad + \left\| \bar{\mathbf{x}}_k - (\mathbf{R}_{k+1}^{k} \bar{\mathbf{x}}_{k+1} + \mathbf{t}_{k+1}^{k}) \right\|^2. \end{aligned} \quad (5)$$

Table 2. The symbols and their descriptions used in this article.

Symbol	Description
N	The total number of points in a point set
Bold uppercase $\mathbf{X}$	A set of points, $\mathbf{X} = \{\mathbf{x}_0, \ldots, \mathbf{x}_N\}$
$\mathbf{X}_k$	The point cloud $\mathbf{X}$ received at time stamp k
x, y, z	The 3D observation of a point in Cartesian coordinates
Bold lowercase $\mathbf{x}_{(k,i)}$	The ith point $\mathbf{x}_i$, $\mathbf{x}_i \in \mathbf{X}_k$
$\hat{\mathbf{x}}_{(k,i)}$	The observed value of $\mathbf{x}_{(k,i)}$
$\tilde{\mathbf{x}}_{(k,i)}$, $\tilde{\mathbf{X}}_k$	The ith point or the point cloud reconstructed with the initial guess by the previous estimation
$\bar{\mathbf{x}}_k$, $\bar{\mathbf{x}}_k^{\text{voxel}}$	The average of the points in $\mathbf{X}_k$ and the voxel, respectively
$\mathbf{T}_{k+1}^{k}$	The estimated transformation $\mathbf{T}$ from the source point cloud $\mathbf{X}_{k+1}$ to the target point cloud $\mathbf{X}_k$ $\mathbf{T}_{k+1}^{k}$ can be divided into the translational vector $\mathbf{t}_{k+1}^{k}$ and rotational motion $\mathbf{R}_{k+1}^{k}$, $[\mathbf{t}_{k+1}^{k}, \mathbf{R}_{k+1}^{k}] = [t_x, t_y, t_z, R_x, R_y, R_z]^T$
$\mathbf{d}_i$	The residual value between corresponding $\mathbf{x}_{(k,i)}$ and $\mathbf{x}_{(k+1,i)}$
N_i	The number of surrounding points around $\mathbf{x}_i$ with a specific distance r
n	The number of points in the cell, used in the voxel-based method
$\mathbf{C}_{(k,i)}$	The covariance matrix associated with point $\mathbf{x}_{(k,i)}$.

Derive the revised cost function by recalling (4) and (5):

$$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N} \| \mathbf{x}'_{(k,i)} - \mathbf{R}\mathbf{x}'_{(k+1,i)} \|^2 + \| (\bar{\mathbf{x}}_k - \mathbf{R}\,\bar{\mathbf{x}}_{k+1} - \mathbf{t} \|^2. \tag{6}$$

Finally, the commonly used singular value decomposition [42] is used to compute the estimated translation and rotation iteratively.

G-ICP

G-ICP [16] extends the classical ICP by combining ICP and "point-to-plane ICP" into distribution-to-distribution matching. G-ICP assumes that each measured point of $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ can be sampled as a Gaussian distribution $\mathbf{x}_{(k,i)} \sim N(\hat{\mathbf{x}}_{(k,i)}, \mathbf{C}_{(k,i)})$ and $\mathbf{x}_{(k+1,i)} \sim N(\hat{\mathbf{x}}_{(k+1,i)}, \mathbf{C}_{(k+1,i)})$, respectively. $\mathbf{C}_{(k,\,i)}$ can be calculated from

$$\mathbf{C}_{(k,i)} = \begin{bmatrix} \operatorname{cov}(x,x) & \operatorname{cov}(x,y) & \operatorname{cov}(x,z) \\ \operatorname{cov}(y,x) & \operatorname{cov}(y,y) & \operatorname{cov}(y,z) \\ \operatorname{cov}(z,x) & \operatorname{cov}(z,y) & \operatorname{cov}(z,z) \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} E(x^2) - \mu_x^2 & E(x,y) - \mu_x\mu_y & E(x,z) - \mu_x\mu_z \\ E(x,y) - \mu_x\mu_y & E(y^2) - \mu_y^2 & E(y,z) - \mu_y\mu_z \\ E(x,z) - \mu_x\mu_z & E(y,z) - \mu_y\mu_z & E(z^2) - \mu_z^2 \end{bmatrix}. \tag{8}$$

μ_x, μ_y, and μ_z are the expected values of $\mathbf{x}_{(k,i)}$, which are approximately equal to the average of its k-surrounding points [e.g., k = 20 by default in the Point Cloud Library (PCL) used in the k-dimensional (k-d) tree search] (https://github.com/PointCloudLibrary/pcl/blob/master/registration/include/pcl/registration/gicp.h#L107). The operator $E(*)$ is used to calculate the expectation of a given component.

Similar to ICP, the residual of G-ICP can be defined [16] as follows:

$$\hat{\mathbf{d}}_i = \hat{\mathbf{x}}_{(k,\,i)} - \mathbf{T}_{k+1}\hat{\mathbf{x}}_{(k+1,i)}. \tag{9}$$

The residual is assumed to be subjected to a Gaussian distribution, which describes the geometry correlation between the given point and the neighboring points. Assume that $\mathbf{x}_{(k,i)}$ and $\mathbf{x}_{(k+1,i)}$ are independently Gaussian distributed, the distribution of $\mathbf{d}_i$ can be represented as

$$\mathbf{d}_i \sim N(\hat{\mathbf{x}}_{(k,i)} - \mathbf{T}_{k+1}\hat{\mathbf{x}}_{(k+1,i)}, \mathbf{C}_{(k,i)} + \mathbf{T}_{k+1}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}{}^{T}) \tag{10}$$

$$= N(0, \mathbf{C}_{(k,\,i)} + \mathbf{T}_{k+1}\mathbf{C}_{(k+1,\,i)}\mathbf{T}_{k+1}{}^{T}). \tag{11}$$

Then we use a maximum-likelihood estimation (MLE) to compute the transformation $\mathbf{T}_{k+1}^{k}$ iteratively:

$$\mathbf{T}_{k+1}^{k} = \operatorname{argmax} \sum_{i=1}^{N} \log(p(\mathbf{d}_i)) \tag{12}$$

$$= \operatorname{argmin} \sum_{i=1}^{N} \mathbf{d}_i^{T} (\mathbf{C}_{(k,i)} + \mathbf{T}_{k+1}^{k}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}^{k}{}^{T})^{-1}\mathbf{d}_i. \tag{13}$$

VGICP

VGICP [17] extends G-ICP with voxelization to significantly reduce the processing time and retain its accuracy as well. First, (15) is extended to compute residual $\mathbf{d}'_i$ between $\mathbf{x}_{(k+1,i)}$ and its neighbor points within distance r in $\mathbf{X}_k$,

Table 3. A summary of the theoretical differences between the various LO methods in terms of point modeling, data association, and cost function.

Method	Point Modeling	Data Association	Cost Function
ICP [15]	N/A	k-d tree	$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N} \| \mathbf{x}'_{(k,i)} - \mathbf{R}\mathbf{x}'_{(k+1,i)} \|^2 + \| (\bar{\mathbf{x}}_k - \mathbf{R}\,\bar{\mathbf{x}}_{k+1} - \mathbf{t} \|^2$
G-ICP [16]	k-d tree	k-d tree	$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \sum_{i=1}^{N} \mathbf{d}_i^{T} (\mathbf{C}_{(k,i)} + \mathbf{T}_{k+1}^{k}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}^{k}{}^{T})^{-1}\mathbf{d}_i$
VGICP [17]	k-d tree	Voxel	$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \sum_{i=1}^{N} N_i \left(\frac{\sum \mathbf{x}_{(k,j)}}{N_i} - \mathbf{T}_{k+1}\mathbf{x}_{(k+1,i)} \right)^{T} \left(\frac{\sum \mathbf{C}_{(k,j)}}{N_i} + \mathbf{T}_{k+1}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}{}^{T} \right)^{-1} \left(\frac{\sum \mathbf{x}_{(k,j)}}{N_i} - \mathbf{T}_{k+1}\mathbf{x}_{(k+1,i)} \right)$
NDT [12]	Voxel	Voxel	$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \sum_{i=1}^{N} \exp\left(\frac{1}{2} (\mathbf{x}'_{(k+1,i)} - \bar{\mathbf{x}}_{(k,i)}^{\text{voxel}})^{T} (\mathbf{C}_{(k,i)}^{\text{ndt\_cell}})^{-1} (\mathbf{x}'_{(k+1,i)} - \bar{\mathbf{x}}_{(k,i)}^{\text{voxel}}) \right)$
LOAM [11]	Edge points and planar points	k-d tree	$\mathbf{T}_{k+1}^{k} = \operatorname{argmin} \frac{1}{2} \left\{ \sum_{i=1}^{N^{\text{edge}}} \| \omega_{(k+1,i)} f(\mathbf{x}_{(k+1,i)}^{e}, \mathbf{T}_{k+1}) \|^2 + \sum_{i=1}^{N^{\text{planar}}} \| \omega_{(k+1,i)} f(\mathbf{x}_{(k+1,i)}^{e}, \mathbf{T}_{k+1}) \|^2 \right\}$
LeGO-LOAM [21]	Clusters, edge points, and ground-planar points	k-d tree	$[t_z, R_x, R_y] = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N^{\text{planarground}}} \| \omega_{(k+1,i)} f(\mathbf{x}_{(k+1,i)}^{e}, \mathbf{T}_{k+1}) \|^2$ $[t_x, t_y, R_z] = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N^{\text{edge\_segmented}}} \| \omega_{(k+1,i)} f(\mathbf{x}_{(k+1i)}^{p}, \mathbf{T}_{k+1}) \|^2$

k-d tree: k-dimensional tree; N/A: not applicable.

$$\{\mathbf{x}_{(k,j)} \,|\, \|\mathbf{x}_{(k,j)} - \mathbf{x}_{(k+1,i)}\| < r\} \quad (14)$$

$$\hat{\mathbf{d}}'_i = \sum_{j=1}^{N_i} (\hat{\mathbf{x}}_{(k,j)} - \mathbf{T}_{k+1}\hat{\mathbf{x}}_{(k+1,i)}). \quad (15)$$

Similar to (12) and (13), the distribution of $\mathbf{d}'_i$ can be represented as [17]

$$\mathbf{d}'_i \sim N\left(0, \ \sum_{j=1}^{N_i} (\mathbf{C}_{(k,j)} + \mathbf{T}_{k+1}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}{}^T)\right), \quad (16)$$

where the calculation of the covariance is similar to G-ICP, and the cost function can be written as

$$\mathbf{T}_{k+1}^k = \operatorname{argmin} \sum_{i=1}^{N} N_i \left(\frac{\sum \mathbf{x}_{(k,j)}}{N_i} - \mathbf{T}_{k+1}\mathbf{x}_{(k+1,i)}\right)^T \left(\frac{\sum \mathbf{C}_{(k,j)}}{N_i} + \mathbf{T}_{k+1}\mathbf{C}_{(k+1,i)}\mathbf{T}_{k+1}{}^T\right)^{-1} \left(\frac{\sum \mathbf{x}_{(k,j)}}{N_i} - \mathbf{T}_{k+1}\mathbf{x}_{(k+1,i)}\right). \quad (17)$$

Equation (17) can be further converted into a voxel-based calculation by substituting $\bar{\mathbf{x}}_{(k,i)}^{\text{voxel}} = (\Sigma \mathbf{x}_{(k,j)}/N_i)$ and $\mathbf{C}_{(k,i)}^{\text{voxel}} = (\Sigma \mathbf{C}_{(k,j)}/N_i)$, respectively, in each voxel. Finally, the MLE is used to compute the transformation $\mathbf{T}_{k+1}^k$ iteratively as (12) and (13).

Compared with the point-correspondence models in ICP or G-ICP, the data association in VGICP is optimized by exploring the voxel correspondences, which are significantly faster than the k-d-tree-based neighboring search in ICP. More specifically, if there are 10 points in a voxel, the computation load for voxel correspondences could be 10-times smaller than ICP or G-ICP. As a result, the efficiency of VGICP is significantly faster than that of the existing point-wise method [16]. However, VGICP needs additional time on the k-d-tree-based neighboring search for the covariance matrix estimation for each point [17].

NDT

Different from the point registration with ICP, the NDT [43] first divided the points cloud into voxels. Then a

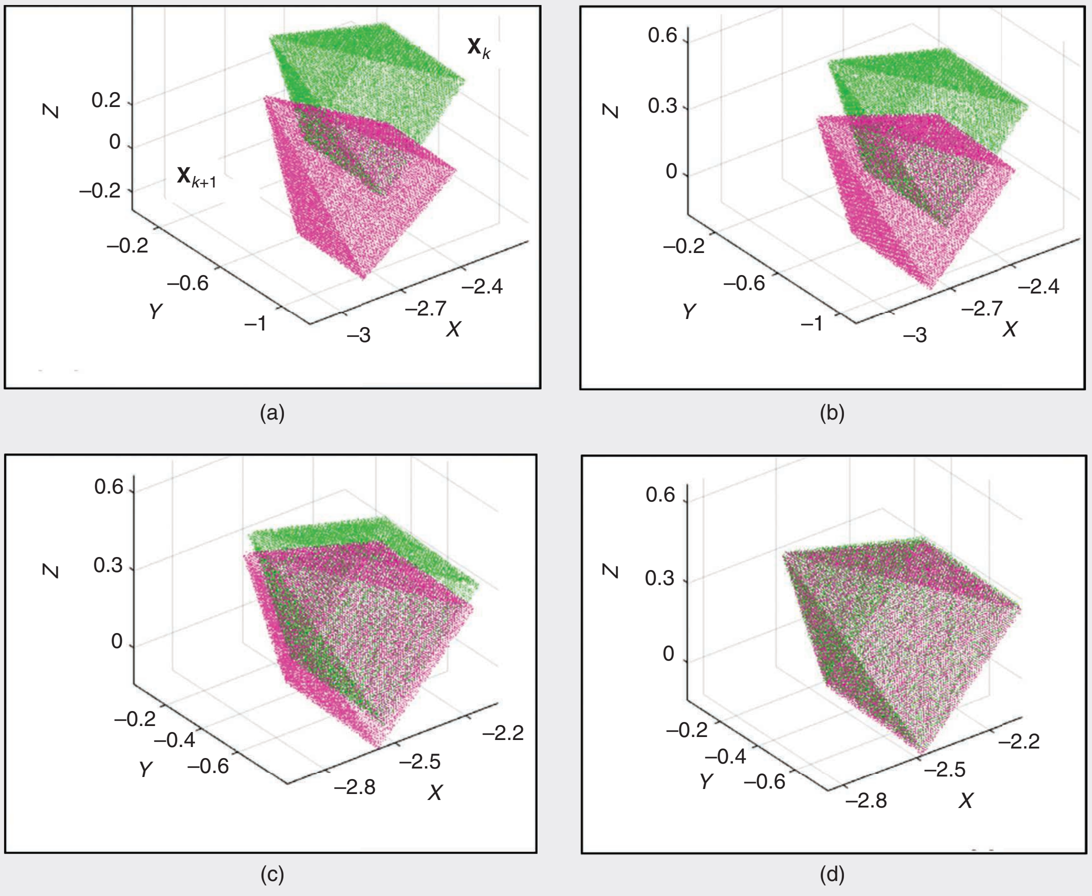


FIG 1 Point clouds $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ are from two consecutive scans. (a) The original scan. (b) and (c) The transformation after several iterations. (d) The results of the final point set registration.

VGICP exploits single-to-multiple distribution by using the average covariaame voxel.

probability density function (PDF) is computed for each voxel to represent the probability of a point $\mathbf{x}_{(k,i)}$ enclosed by the cell in $\mathbf{X}_k$. The covariance matrix of the points inside a cell of $\mathbf{X}_k$ is

$$\mathbf{C}^{\text{ndt\_cell}}_{(k,\,i)} = \frac{1}{n}\sum_{i=1}^{n}((\mathbf{x}_{(k,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)})(\mathbf{x}_{(k,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)})^T). \tag{18}$$

The calculation of the covariance $\mathbf{C}^{\text{ndt\_cell}}_{(k,\,i)}$ is different from that of the $\mathbf{C}^{\text{voxel}}_{(k,\,i)}$ in VGICP. The $\mathbf{C}^{\text{ndt\_cell}}_{(k,\,i)}$ in the NDT considers the points in the cell only, whereas at least four or more points in a cell are needed. In other words, a given point is associated with the distribution constructed in the target frame, leading to the point-to-distribution correspondence listed in Table 1. VGICP exploits single-to-multiple distribution by using the average covariaame voxel.

The PDF can be formulated as

$$p(\mathbf{x}_{(k,i)}) = \frac{1}{c}\exp\left(-\frac{(\mathbf{x}_{(k,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)})^T(C^{\text{ndt\_cell}}_{(k,\,i)})^{-1}(\mathbf{x}_{(k,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)})}{2}\right), \tag{19}$$

where c is a constant and can be set to one [43]. An example of a 2D NDT is presented in Figure 2.

The NDT of the scan $\mathbf{X}_k$ is built. Then the maximum matching score function of all the points in $\tilde{\mathbf{X}}_k$ can be converted to minimize the negative sum of the score

$$\mathbf{T}^k_{k+1} = \operatorname{argmin}\frac{1}{2}\sum_{i=1}^{N}(\tilde{\boldsymbol{x}}_{(k+1,\,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)})^T(\mathbf{C}^{\text{ndt\_cell}}_{(k,\,i)})^{-1} \times (\tilde{\boldsymbol{x}}_{(k+1,\,i)} - \bar{\mathbf{x}}^{\text{voxel}}_{(k,\,i)}). \tag{20}$$

$\mathbf{T}^k_{k+1}$ can be updated using the Newton method to minimize the score iteratively. Different from ICP, the NDT employs multiple Gaussian components (each Gaussian component corresponding to a voxel) to describe the geometry of the 3D point clouds. However, the performance of the NDT relies on the selection of the size of the voxel.

NDT-OMP is a modified version of the NDT implementation in the PCL, which is an NDT-OMP algorithm that is Streaming SIMD Extensions (SSE) friendly and multithreaded [18]. The processing speed of NDT-OMP is up to 10-times faster compared to the original version in the PCL.

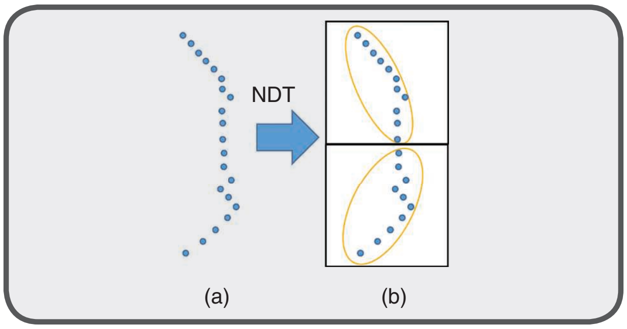


FIG 2 The points are divided into (a) multiple cells and (b) their distribution in a voxel in 2D NDT.

LOAM

Feature Extraction

LOAM [11] mainly involves three steps: feature extraction, LO, and the lidar-odometry process. Let m indicate the ring number of point cloud $\mathbf{X}_k$ and $\mathbf{S}^m_{(k,i)}$ be a set of continuous neighboring points of $\mathbf{x}_{(k,i)}$ in scan ring m. Normally, the points in a ring are in a clockwise or counterclockwise order according to the receiving time within a scan period (normally 0.1 s). The feature points are extracted according to the curvature c_i of point $\mathbf{x}_{(k,i)}$ and its successive points [11]:

$$c_i = \frac{1}{N_s * \left\|\mathbf{x}_{(k,i)}\right\|}\left\|\sum_{j\in \mathbf{S}^m_{(k,i)},\, j\neq i}(\mathbf{x}_{(k,i)} - \mathbf{x}_{(k,j)})\right\|. \tag{21}$$

$\mathbf{x}_{(k,j)}$ denotes the consecutive point of $\mathbf{x}_{(k,i)}$ within subset $\mathbf{S}^m_{(k,i)}$. N_s represents the number of points in $\mathbf{S}^m_{(k,i)}$, including $\mathbf{x}_{(k,i)}$ and 10 consecutive points. The operator $\|\cdot\|$ indicates the L2 vector norm. A point is selected as the edge point if its curvature value is larger than a predetermined threshold c_{e-th}, or classified as a planar point by a smaller curvature in $\mathbf{X}_k$. For points $\mathbf{x}^m_{(k,i)}$ within ring $\mathbf{X}^m_{(k,i)}$ in each scan, we normally divide it into four to eight subregions $\mathbf{X}^{s,\,m}_{(k,i)}$, and each subregion selects two edge points $\mathbf{x}^e_{(k,i)}$ from $\mathbf{x}^{e,m}_{(k,i)}$ and four planar points $\mathbf{x}^p_{(k,i)}$ within $\mathbf{x}^{p,m}_{(k,i)}$ for further processing [11], as shown in (22). Ten-times edge points and enormous planar features are utilized in the mapping process to achieve better accuracy but bring more computation costs. An example of the feature-extraction results is displayed in Figure 3.

$$\mathbf{x}^{s,\,m}_{(k,i)} = \begin{cases}\mathbf{x}^e_{(k,i)} \in \mathbf{x}^{e,m}_{(k,i)},\ c_i > c_{e-th} \text{ and } N^e_{(k,i)} \le 2\\ \mathbf{x}^p_{(k,i)} \in \mathbf{x}^{p,m}_{(k,i)},\ c_i < c_{e-th} \text{ and } N^p_{(k,i)} \le 4\end{cases}. \tag{22}$$

Lidar Odometry

In general, the odometry is estimated by accumulating the transformation among consecutive frames of point clouds. The role of lidar odometry in LOAM is to estimate the motion between two successive sweeps. The estimated

$\mathbf{T}^{L}_{(k,\,k+1)}$ is used to correct the distortion of points in $\mathbf{X}_{k+1}$ and provide the initial guess to project $\mathbf{X}_k$ as $\tilde{\mathbf{X}}_k$ During the next frame, the corresponding features can be found between $\tilde{\mathbf{X}}_k$ and $\mathbf{X}_{k+1}$

The role of lidar odometry in LOAM is to estimate the motion between two successive sweeps.

For each edge point $\mathbf{x}^{e}_{(k+1,i)}$ searching its nearest neighbors in $\tilde{\mathbf{X}}_k$ to fit a line by $\tilde{\mathbf{x}}^{e}_{(k,j)}, \tilde{\mathbf{x}}^{e}_{(k,l)} \in \tilde{\mathbf{X}}_k$ as the corresponding edge. The distance $d^{e}_{(k+1,i)}$ between the edge point $\mathbf{x}^{e}_{(k+1,i)}$ and the fitted line represents the residual of the edge feature to be minimized, which can be described as

$$d^{e}_{(k+1,i)} = \frac{|(\mathbf{x}^{e}_{(k+1,i)} - \tilde{\mathbf{x}}^{e}_{(k,j)}) \times (\mathbf{x}^{e}_{(k+1,i)} - \tilde{\mathbf{x}}^{e}_{(k,l)})|}{|\tilde{\mathbf{x}}^{e}_{(k,j)} - \tilde{\mathbf{x}}^{e}_{(k,l)}|}. \quad (23)$$

Similarly, for each plane point $\mathbf{x}^{p}_{(k+1,i)}$ in $\mathbf{X}_{k+1}$, the distance $d^{p}_{(k+1,i)}$ between the point and the fitted plane in $\mathbf{X}'_k$, is the residual of the plane feature to be minimized, which can be represented as

$$d^{p}_{(k+1,i)} = \frac{\left|\begin{matrix}(\mathbf{x}^{p}_{(k+1,i)} - \tilde{\mathbf{x}}^{p}_{(k,j)})\cdot \\ ((\tilde{\mathbf{x}}^{p}_{(k,j)} - \tilde{\mathbf{x}}^{p}_{(k,l)}) \times (\tilde{\mathbf{x}}^{p}_{(k,j)} - \tilde{\mathbf{x}}^{p}_{(k,m)})\end{matrix}\right|}{|(\tilde{\mathbf{x}}^{p}_{(k,j)} - \tilde{\mathbf{x}}^{p}_{(k,l)}) \times (\tilde{\mathbf{x}}^{p}_{(k,j)} - \tilde{\mathbf{x}}^{p}_{(k,m)})|}, \quad (24)$$

where $\tilde{\mathbf{x}}^{p}_{(k,j)}, \tilde{\mathbf{x}}^{p}_{(k,l)}, \tilde{\mathbf{x}}^{p}_{(k,m)}$ are the three nearest points of $x^{p}_{(k+1,i)}$ among the planar points in $\mathbf{X}'_k$ using the k-d tree search.

A geometric relationship between the edge and plane feature point in $\mathbf{X}_{k+1}$ and the corresponding in $\mathbf{X}'_k$ can be estimated as

$$f(\mathbf{x}^{e}_{(k+1,i)}, \mathbf{T}_{k+1}) = d^{e}_{(k+1,i)} \quad (25)$$

$$f(\mathbf{x}^{p}_{(k+1,i)}, \mathbf{T}_{k+1}) = d^{p}_{(k+1,i)}. \quad (26)$$

Therefore, transformation $\mathbf{T}^{k}_{k+1}$ can be calculated by minimizing the cost function based on (25) and (26):

$$\mathbf{T}^{k}_{k+1} = \operatorname{argmin} \frac{1}{2}\left\{\sum_{i=1}^{N^{\text{edge}}} \left\|\omega_{(k+1,i)} f(\mathbf{x}^{e}_{(k+1,i)}, \mathbf{T}_{k+1})\right\|^2 + \sum_{i=1}^{N^{\text{planar}}} \left\|\omega_{(k+1,i)} f(\mathbf{x}^{p}_{(k+1,i)}, \mathbf{T}_{k+1})\right\|^2\right\}. \quad (27)$$

N^{edge} and N^{planar} are the number of edges and planar points obtained in $\mathbf{X}_{k+1}$. The smaller-weight $\omega_{(k+1,i)}$ is assigned to feature points with a large residual. The optimization can be solved using the Levenberg–Marquardt [24] method by minimizing the distance of feature.

Lidar Mapping

The mapping algorithm matches the $\mathbf{X}_{k+1}$ and the point cloud map $\mathbf{M}_k$ to mitigate the error estimation arising from LO. If $\mathbf{T}^{L}_{(k,\,k+1)}$ is the transformation of LO between t_k and t_{k+1}, the initial guess $\mathbf{T}^{W}_{k+1}$ can be represented as

$$\mathbf{T}^{W}_{k+1} = \mathbf{T}^{W}_{k}\mathbf{T}^{L}_{(k,\,k+1)}. \quad (28)$$

Then, $\mathbf{T}^{W}_{k+1}$ can be used to transform $\mathbf{X}_{k+1}$ into a world coordinate, donated as $\bar{\mathbf{X}}_{k+1}$. Similar to the process of LO in the "Lidar Odometry" section, the feature-to-feature correspondence can be extracted by applying a k-d tree search between $\bar{\mathbf{X}}_{k+1}$ and $\mathbf{M}_k$. Then the $\mathbf{T}^{W}_{k+1}$ can be optimized by minimizing the residuals following (23), (24), and (27). Later, the $\bar{\mathbf{X}}_{k+1}$ is added to the $\mathbf{M}_k$ to generate the $\mathbf{M}_{k+1}$ for scan-to-map processing next.

Compared with the point-wise method theoretically, the curvature is exploited by LOAM for the edge- and planar-point classification, based on (21) and (22). Then the feature points are utilized to compute the interframe motion by minimizing the residuals in (23) and (24). Finally, precious-state estimation can be obtained with the scan-to-mapping strategy, based on the approximate initial guess using (28). Compared with the conventional point-wise LO pipelines, LOAM extracts the features, which can significantly decrease the probability of data misassociation. Moreover, the number of correspondences involved in the optimization is significantly smaller than the one for the point-wise pipeline, leading to improved efficiency.

LeGO-LOAM

LeGO-LOAM [21] is a lightweight and ground-optimized LOAM. First, LeGO-LOAM proposed projecting a point cloud onto a range image with a certain resolution base on the horizontal and vertical angular resolution of the lidar scanner. Each valid point in the range image has a different pixel value based on the Euclidean distance from the point to the sensor. Then, segmentation is applied to the range images to classify the ground and the

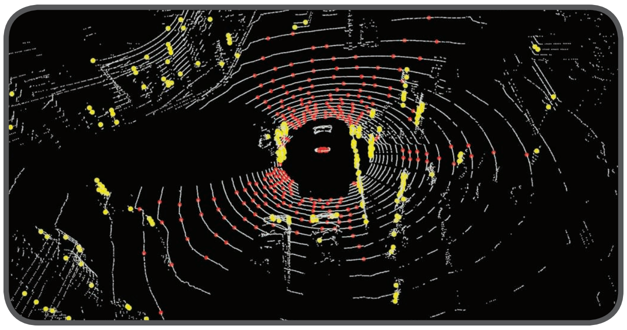


FIG 3 An illustration of extracted edge (yellow) and planar points (red) for scan to map, concerning a frame of lidar point clouds (grey) (Velodyne HDL-32E) in a crossing road using LOAM.

We use the relative pose error to investigate the local consistency of the SLAM trajectory with standard practice.

large objects. In the odometry module, $[t_z, \theta_{\text{roll}}, \theta_{\text{pitch}}]$ is obtained by finding the correspondence of planar features from the ground points. Meanwhile, $[t_x, t_y, \theta_{\text{yaw}}]$ is estimated by matching the edge features from the segmented clusters:

$$[t_z, R_x, R_y] = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N^{\text{p\_ground}}} \left\| \omega_{(k+1,i)} f(\boldsymbol{x}^p_{(k+1,i)}, \boldsymbol{T}_{k+1}) \right\|^2 \tag{29}$$

$$[t_x, t_y, R_z] = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N^{\text{e\_seg}}} \left\| \omega_{(k+1,i)} f(\boldsymbol{x}^e_{(k+1,i)}, \boldsymbol{T}_{k+1}) \right\|^2, \tag{30}$$

where $N^{\text{p\_ground}}$ and $N^{\text{e\_seg}}$ are the number of planars pointed within the ground area and the edge points from the large objects on the range image in $\mathbf{X}_{k+1}$. According to the evaluation in [21], the drift in the vertical direction of LeGO-LOAM is significantly smaller than conventional LOAM because $[t_z, \theta_{\text{roll}}, \theta_{\text{pitch}}]$ is innovatively estimated by the ground points. We believe that this is one of the major contributions of LeGO-LOAM. Moreover, compared with conventional LOAM, LeGO-LOAM also performs loop-closure detection to eliminate drift in the lidar-mapping process [44].

Summary
The models of the point cloud are different among the diverse methods mentioned previously. An overview of the fundamental differences of the LO methods selected is shown in Table 3. K-d trees and voxelization are used for point modeling and the corresponding search in the aforementioned LO methods. G-ICP adopted a Gaussian-based point representation, hence it has advantages in terms of accuracy and robustness compared to ICP, but with heavy computation costs on point modeling. The NDT is a computationally efficient LO method because voxelization is applied for both modeling and corresponding searching; however, its performance is sensitive to cell size in a diverse environment. VGICP combines the strength of G-ICP and NDT and achieves both fast and accurate point registration.

Feature-wise LO algorithms, such as LOAM and its variants, employ feature extraction before searching the correspondence, which reduces the computational load, thus it can be run in real time for most of the cases. But the feature-wise method uses a subset of the point cloud and can cause degeneration in scenes with limited environmental features. We now focus on an evaluation of both point- and feature-based LO methods that challenge LO methods in urban canyons.

Experiments

Experiment Setup
This article does not use KITTI as it has been widely evaluated in numerous literature, and the results are summarized in [45]. The performances of the LO methods are evaluated using our recently published UrbanNav data set [46], which contains the data collected from various degrees of urban areas in Hong Kong and Tokyo. The data set contains measurements from GNSS, inertial measurement units (IMUs), cameras, and lidar. Besides, the ground-truth data is recorded by NovAtel SPAN-CPT, which integrates GNSS RTK with the fiber-optic gyroscope level of IMU. This article focuses on two competitive data sets: HK-Data20200314 (Data1) and HK-Data20190428 (Data2). Data2 is a more challenging data set compared to Data1. The typical scenes of both data sets are depicted in Figure 4. The complete data sets are publicly available for further evaluation and algorithm development at https://www.polyu-ipn-lab.com/download.

The data sets are processed offline with a laptop, and the specifications of the computing device used to evaluate the LO algorithms are as follows:

- An Intel i7-9750H CPU at 2.60 GHz × 12 threads
- 2 × 16-GB DDR4 2666-MHz random-access memory
- A GeForce GTX 1660 Ti graphic card.

We set up an Ubuntu 18.04/ Robotic Operating System (ROS) melodic as the platform baseline for the majority of the LO pipelines. The original publicly available LOAM implementation can be operated only in Ubuntu 16.04/ROS Kinetic due to the software incompatibilities in ROS melodic.

The recommended parameters we provide have been retained across all the experiments. We substituted the package configuration for Velodyne HDL32. The detailed configuration and rosbag playback rate are presented in Table 4. For FastG-ICP and Fast LOAM, the published rate was set to 0.5 as the processing time for some frames is slower than 10 Hz. All of the experiments are conducted with published points data only. The estimated trajectories of all the methods are shown in Figure 5.

The performance of the LO methods was evaluated via the popular EVO tools (https://github.com/MichaelGrupp/evo), a Python package widely used for evaluating and comparing odometry or simultaneous localization and mapping (SLAM) algorithms. We use the relative pose error (RPE) to investigate the local consistency of the SLAM trajectory with standard practice [47]. RPE compares the estimated relative pose with the reference pose in a fixed time interval: Given a sequence of poses from the estimated trajectory $\mathbf{T}_{\mathrm{est}} = \{\mathbf{T}_{(\mathrm{est},1)}, \mathbf{T}_{(\mathrm{est},2)}, \ldots, \mathbf{T}_{(\mathrm{est},N_{\mathrm{epochs}})}\}$ and ground truth $\mathbf{T}_{\mathrm{gt}} = \{\mathbf{T}_{(\mathrm{gt},1)}, \mathbf{T}_{(\mathrm{gt},2)}, \ldots, \mathbf{T}_{(\mathrm{gt},N_{\mathrm{epochs}})}\}$, N_{epochs} represents the number of states for evaluation. The $RPE_{i,j}$ between time stamp t_i and t_j can be defined as [48]

$$\mathrm{RPE}_{i,j} = (\mathbf{T}^{-1}_{(\mathrm{gt},i)}\mathbf{T}_{(\mathrm{gt},j)})^{-1}(\mathbf{T}^{-1}_{(\mathrm{est},i)}\mathbf{T}_{(\mathrm{est},j)}). \tag{31}$$

$\mathrm{RPE}_{i,j}$ belongs to the Special Euclidean Group $SE(3)$.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_{\mathrm{epochs}}}\sum_{i=1}^{N_{\mathrm{epochs}}} \mathrm{RPE}_i^2}. \tag{32}$$

N_{epochs} represents the number of epochs to be evaluated. The root-mean-square error (RMSE) can be calculated based on the overall RPE. The evaluation results of both data sets are displayed in Table 5.

Computation Cost

The processing time per frame (PTPF) is used for the evaluation of the computation cost of LO, as shown in Table 6. The PTPF is measured from receiving a new point cloud to the result of LO odometry. For the point registration-based method, only the processing time of odometry was recorded as the PTPF. For LOAM-related methods, the processing time was collected at the end of the odometry and the mapping process. LOAM-related methods evaluate the PTPF of odometry and mapping separately, among which LOAM ranked best in the odometry process, with

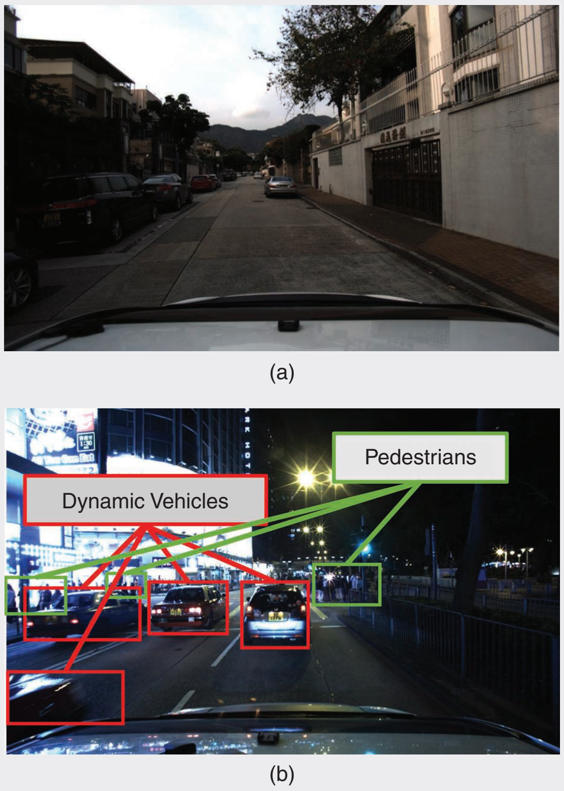


FIG 4 A demonstration of the scenarios in the two urban data sets. (a) Data1: Low-rise buildings and multiple right-turn areas in HK-Data20200314. (b) Data2: A variety of dynamic vehicles and numerous high-rise buildings in HK-Data20190428.

Table 4. The code implementation and configuration of the publicly available LO methods.

Method	Code Repository	Recommended Parameters	Rosbag Playback Rate
ICP [15]	github.com/koide3/hdl_graph_slam	The default parameters in hdl_graph_slam.launch.	0.1
G-ICP [16]			0.1
FastG-ICP [17]	github.com/SMRT-AIST/fast_gicp	It is integrated into hdl_graph_slam. The default parameters are adopted in hdl_graph_slam.launch.	0.5
FastVGICP [17]			1
FastVGICPCuda [17]			1
NDT [12]	github.com/koide3/hdl_graph_slam	The ndt_resolution was set to 3 for the outdoor environment in hdl_graph_slam.launch.	0.1
NDT-OMP [18]	github.com/koide3/ndt_omp/commit/c799f459b4838dd9c65968573370b16c4e7ce7d9		1
LOAM [11]	github.com/laboshinl/loam_velodyne	VLP-16 was changed to HDL-32 in loam_velodyne.launch.	1
A-LOAM [19]	github.com/HKUST-Aerial-Robotics/A-LOAM	The default parameters are adopted in aloam_velodyne_HDL_32.launch.	1
LeGO-LOAM [21]	github.com/RobustFieldAutonomyLab/LeGO-LOAM	The VLP-16 configuration was changed to HDL-32E in utility.h.	1
LIO-mapping [22]	github.com/hyye/lio-mapping	"sensor_type = 32, no_deskew = false" was updated in 64_scans_test.launch as 32 scans_test.launch.	1
Fast LOAM [23]	github.com/wh200720041/floam	The scan_line = 64 was changed to 32 in floam.launch.	0.5

a mean value of 9.41 ms, while LIO-mapping performed fastest in the mapping process, with a mean value of 105.54 ms. Fast LOAM demonstrated a mean PTPF of 60.78 ms by merging the odometry and mapping process of the conventional LOAM.

FastVGICPCuda achieved the fastest processing time among the point-registration methods while LOAM performed rapidly in terms of the odometry process. LeGO-LOAM demonstrated the fastest time in the mapping process among the LOAM series methods. However, the processing time of FastG-ICP, FastVGICP, and FastVGICPCuda had a potential issue when dealing with large data points (https://github.com/SMRT-AIST/fast_gicp/issues/17) (marked in red in Table 6) because the max processing speed is much faster than the mean processing speed and published rate in Table 4. The processing delay might result in a large error in some frames.

Detailed Analysis of Accuracy

The translation error is used as the criteria of accuracy evaluation, as shown in Table 5 and Figure 7. For Data1, G-ICP obtained the most accurate results among all the evaluated point-registration-related methods, while LOAM achieved the best outcomes in all the methods in terms of the RMSE. For another data set, Data2, G-ICP retained the best performance, while Fast LOAM performed better than LOAM.

A-LOAM's performance in Table 5 and Figure 6(a) of Data1 is much worse than that of the other LOAM-related methods, up to approximately 0.8 meters in Data1. The reason is that A-LOAM lacks outlier removal in the feature-extraction

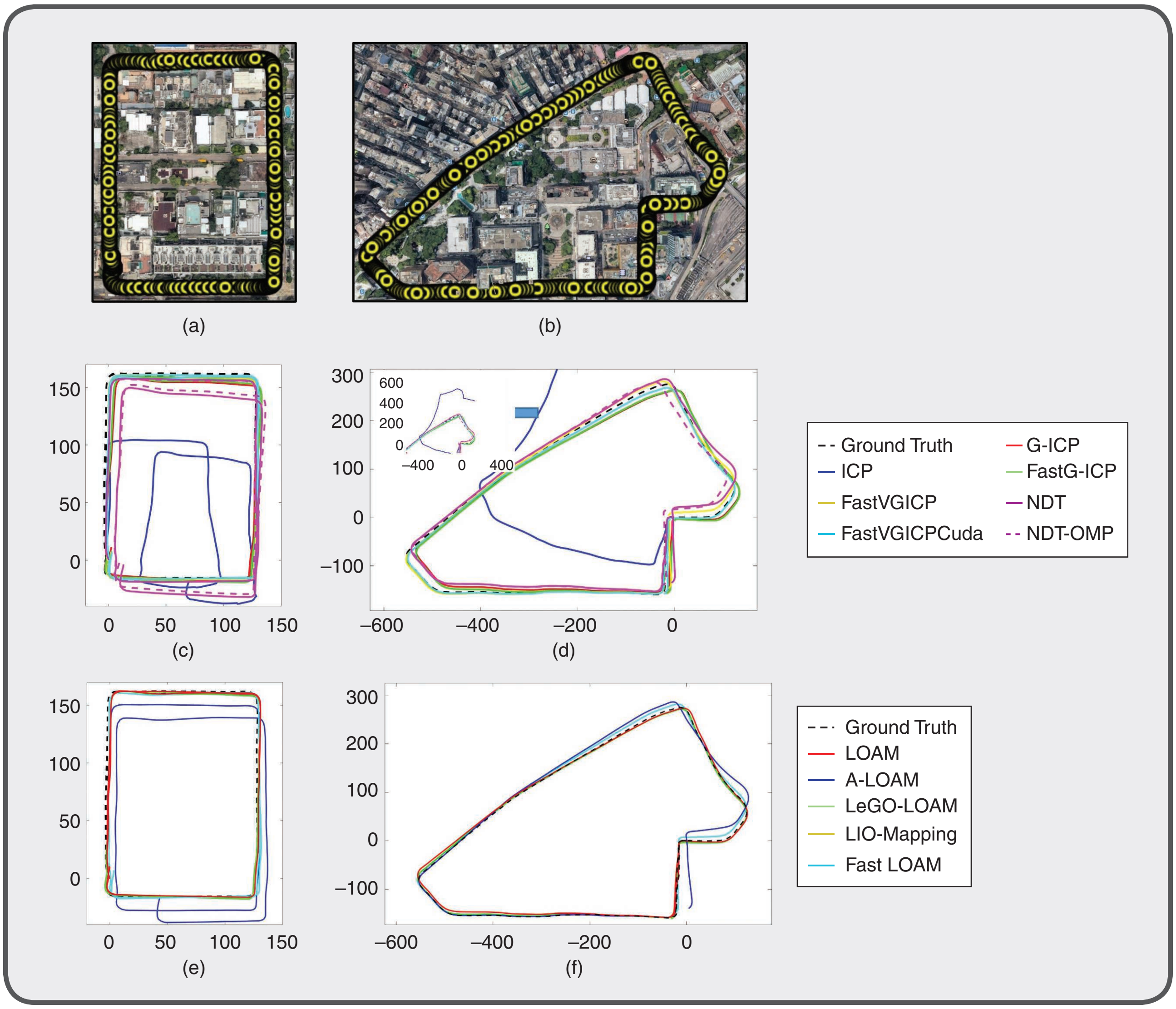


FIG 5 (a) The satellite-image view of the ground truth in Data1. (b) The satellite-image view of the ground truth in Data2. The trajectories in (c) and (d) are evaluated by the point-wise LO methods of Data1 and Data2, respectively. The trajectories in (e) and (f) are evaluated by the feature-wise LO methods of Data1 and Data2, respectively. The background image of Figure 5 (a) and (b) are captured from Google Earth software.

process, which is part of the original LOAM [11]. To verify this, we perform an experiment to apply the feature-extraction portion in “LIO-mapping” together with the “A-LOAM” odometry and mapping process. The accuracy of LO is significantly improved compared with A-LOAM itself, which is shown in Figure 6(b), as it can nearly overlay when returned to start.

In terms of the theoretical comparisons and the experimental results in the previous section, we argue that three major factors affect the performance of LOs, including 1) the motion difference between two consecutive epochs, which has an impact on the initial guess that project $\mathbf{X}_k$ to $\tilde{\mathbf{X}}_k$; 2) the density of dynamic objects, a static environment assumption, and the geometry corresponding is affected; and 3) the degree of urbanization of the evaluated environment, which explores the accuracy of LOs for those highly urbanized areas that GNSS positioning has degenerated [5].

The first factor we considered is the motion difference between the two frames. For calculating the motion difference from the time stamp t_i to t_{i+1}, we use the Euclidean norm of the vector ξ_{k+1} in Lie algebra $SE(3)$ to represent its corresponding rigid body motion $\mathbf{T}_{k+1}^{k} \in SE(3)$. The formula can be expressed as

$$\Delta d_{i+1} = \left\| \log(\mathbf{T}_{k+1}^{k})^{\vee} \right\|, \quad \mathbf{T}_{k+1}^{k} \in R^{4\times 4}, \tag{33}$$

where superscript $\vee$ indicates the de-antisymmetric operation. The operator $\|\cdot\|$ indicates the L2 vector norm.

The second factor is the number of dynamic objects in each scan. We labeled dynamic objects such as cars and buses in the two data sets with SUSTechPOINTS [50], a semiautomatic annotation tool. The density of the dynamic objects factor is defined as

$$c = \left(\frac{N_{\text{car}} + N_{\text{bus}}}{N_{\text{total}}} \right) \times 100\%, \tag{34}$$

where N_{car} and N_{bus} represent the number of lidar points of cars and buses, respectively, separately in the current scan. N_{total} indicates the total number of points in the current frame.

To evaluate the degree of urbanization of the evaluated scene, our previous work in [51] proposed adopting 3D building models to further estimate the sky mask (GNSS sky plot with building boundaries). The mean mask elevation angle μ_{MEA} is defined to quantitatively represent the degree of urbanization as follows:

$$\mu_{\text{MEA}} = \frac{\sum_{\alpha=1}^{N} \theta_\alpha}{N}, \tag{35}$$

where θ_α represents the elevation angle, related to the height of a building at an azimuth angle of α; and N denotes the number of equally spaced azimuth angles from the sky mask, i.e., 360° in the resolution of the azimuth angle, that is, 1°. A location surrounded by high-rise buildings results in a large μ_{MEA}, and vice versa, a small μ_{MEA} is obtained in rural areas.

To verify the impacts of the listed three factors in the performance of the LO pipelines, we look into the results,

Table 5. The evaluation results of the two urban data sets. The best performances of the registration- and feature-based methods in terms of accuracy are highlighted in **bold** and **blue**, respectively.

				Relative Translation Error (m)		Relative Rotation Error (°)	
Data Set	Description	Trajectory Length	Method	RMSE	Mean	RMSE	Mean
Data1	Low-urbanization, small loop	1.21 km	ICP	1.857	1.529	2.073	1.533
			G-ICP	**0.371**	0.326	1.912	1.268
			FastG-ICP	0.383	0.33	1.814	1.238
			FastVGICP	0.372	**0.321**	**1.67**	**1.113**
			FastVGICPCuda	0.627	0.389	1.773	1.184
			NDT	0.405	0.323	1.938	1.301
			NDT-OMP	0.51	0.397	1.84	1.197
			LOAM	**0.354**	**0.311**	2.113	1.379
			A-LOAM	0.803	0.476	1.87	1.232
			LeGO-LOAM	0.374	0.324	1.972	1.236
			LIO-mapping	0.479	0.337	**1.201**	**0.803**
			Fast LOAM	0.376	0.322	1.661	1.094
Data2	Heavy traffic, tall buildings	2.01 km	ICP	1.684	1.213	1.494	0.902
			G-ICP	**0.417**	**0.296**	1.129	0.662
			FastG-ICP	0.543	0.301	1.303	**0.472**
			FastVGICP	0.711	0.367	**1.093**	0.618
			FastVGICPCuda	0.874	0.41	1.734	0.732
			NDT	0.816	0.422	1.106	0.657
			NDT-OMP	0.788	0.388	1.149	0.661
			LOAM	0.45	0.321	1.383	0.799
			A-LOAM	0.478	0.331	1.234	0.695
			LeGO-LOAM	0.462	0.333	1.226	0.694
			LIO-mapping	0.664	0.379	**0.886**	**0.471**
			Fast LOAM	**0.423**	**0.294**	1.141	0.619

Table 6. The PTPF evaluation results of Data1 and Data2. The top performances of the registration and feature based are highlighted in **bold** and blue, respectively. The values in red denote the maximum PTPF that has obviously shifted, compared to the mean values.

		Odometry PTPF (ms)			Mapping PTPF (ms)		
Data Set	Method	Maximum	Minimum	Mean	Maximum	Minimum	Mean
Data1	ICP	1,156.76	29.99	91.47		N/A	
	G-ICP	1,476.64	92.79	232.89		N/A	
	FastG-ICP	375.21	24.5	75.02		N/A	
	FastVGICP	786.19	22.39	42.23		N/A	
	FastVGICPCuda	1,182.24	**12.92**	**29**		N/A	
	NDT	1,681.75	156.59	472.97		N/A	
	NDT-OMP	**301.57**	9.35	51.67		N/A	
	LOAM	49.04	3.98	9.41	345.32	16.32	138.81
	A-LOAM	40.74	10.73	15.93	612.79	61.27	198.51
	LeGO-LOAM	31.58	4.64	9.73	252.27	30.32	122.93
	LIO-mapping	79.88	7.88	25.22	274.58	28.94	105.54
	Fast LOAM	N/A			124.02	22.85	60.78
Data2	ICP	988.06	34.42	105.38		N/A	
	G-ICP	520.1	106.35	219.64		N/A	
	FastG-ICP	1,465.45	21.69	87.14		N/A	
	FastVGICP	339.66	25.77	47.16		N/A	
	FastVGICPCuda	1,626.14	**16.03**	**38.08**		N/A	
	NDT	1,442.36	195.41	488.43		N/A	
	NDT-OMP	**142.85**	13.87	39.18		N/A	
	LOAM	56.85	4.7	10.73	322.08	17.5	125.8
	A-LOAM	38.06	11.96	15.2	588.59	76.02	209.23
	LeGO-LOAM	33.7	3.35	11.16	248.47	26.49	115.19
	LIO-mapping	86.57	9.13	28.16	328.82	37.4	119.38
	Fast LOAM	N/A			164.99	27.23	90.52

which are presented in Figures 8 and 9. The interquartile range (IQR) [52] is useful to find the outliers with a large number of data. The RPE exceeding the Q3+1.5 IQR of the boxplot in Figure 7 are considered outliers and marked with red dots. For example, the threshold to label the outliner of LOs is 0.7 m and 1 m for Data1 and Data2, respectively.

Verification of the Listed Three Factors Using Data1

Data1—Factor 1—Motion Difference

As depicted in Figure 8(a), the motion difference increased from 0 to 5 m, and the translation error jumped to 0.7 m for both G-ICP and LOAM. It is a typical scene of motion changing when a vehicle starts from the roadside. Both methods demonstrated an accurate performance between Figure 8(a) and (b) as there was a minor change in motion. As shown in Figure 8(b), the ego vehicle slows down before passing the crossing and thus produces a significant motion offset. As shown in Figure 8(c), the error increases again as the car turns right, then returns to the starting point, before stopping at the end.

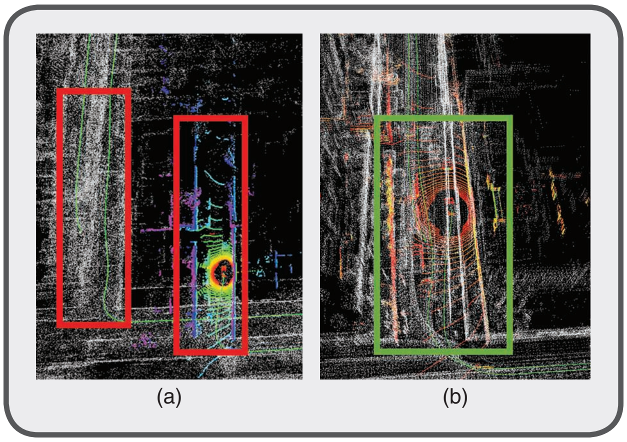


FIG 6 (a) The mapping process of A-LOAM. The repeating route does not have the overlap, which is (b) marked in red, replacing the feature-extraction process of A-LOAM with the one in the LIO-mapping. The pose nearly overlays the previous estimation, which is marked in green.

Data1—Factor 2—Dynamic Objects

Data1 is a low-traffic area without any vehicles, with the exception of in Figure 8(d). The results show that the LOAM is more susceptible to dynamic vehicles compared with G-ICP in the evaluated data set.

Data1—Factor 3—Degree of the Urbanization (Sky Mask)

Figure 8(e) depicts that the error of G-ICP reached a peak as the sky mask changed. Further, the LOAM error increased in Figure 8(f) when the density of the buildings drastically dropped.

Verification of the Listed Three Factors Using Data2

Data2—Factor 1—Motion Difference

As presented in Figure 9(a), the translation error jumped to more than 1 m for both G-ICP and Fast LOAM. The ego vehicle resumed as the traffic light turned back to green from red, and the motion difference increased from 0 to 10 m. Both methods achieved superior performance around frame 300 between Figure 9(a) and (b) as there was a stopping scenario and a motion equal to zero. In Figure 9(b), the ego car turned to the right through a busy intersection. The error was up to 1.5 meters due to the motion difference plus the number of dynamic vehicles.

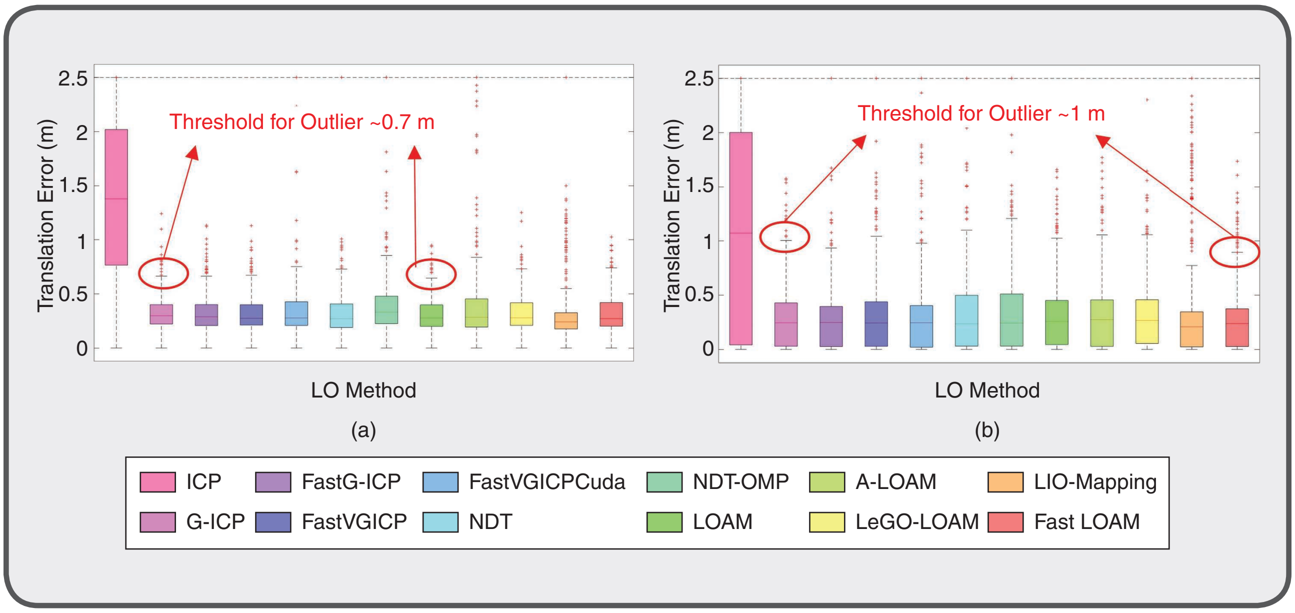


FIG 7 The boxplot evaluation of translation error statistics for the LO methods. The whiskers were set at 1.5 times the interquartile range (IQR) (Q3 + 1.5*IQR) to better classify the outliers [49]. (a) Data1 (light urban) and (b) Data2 (dense urban).

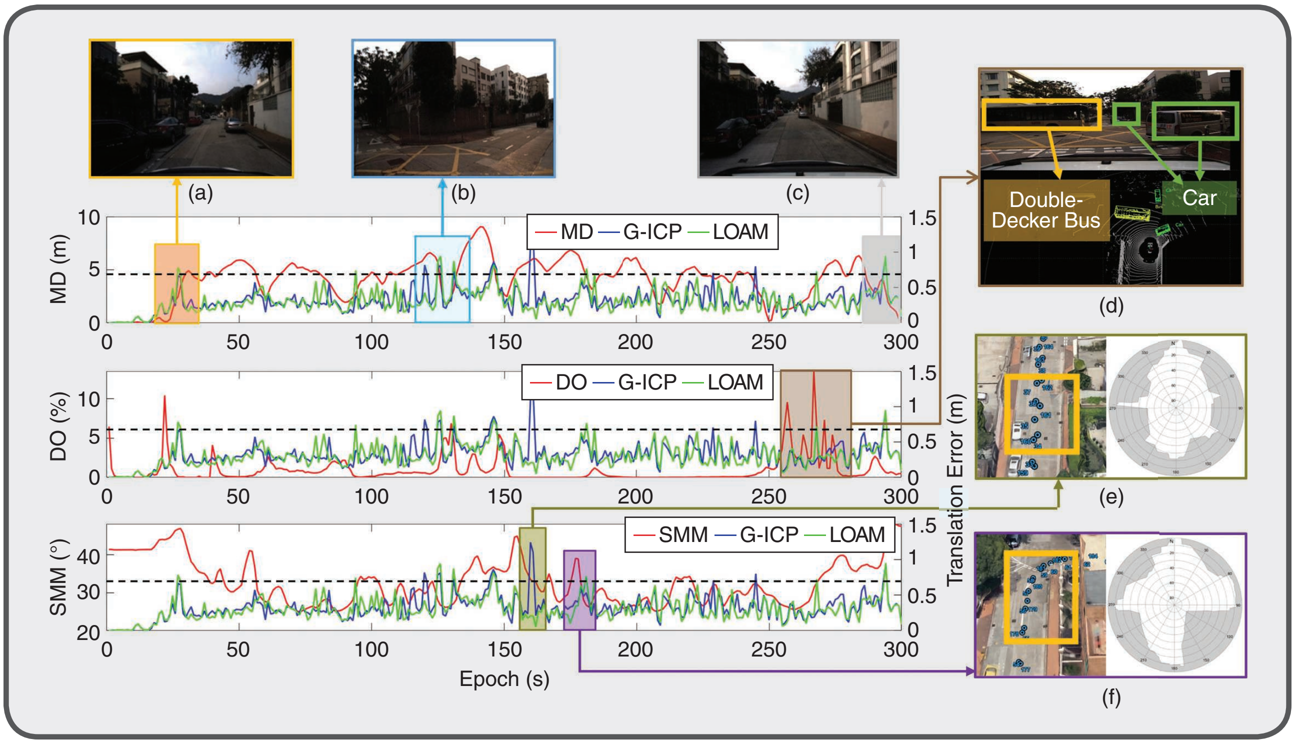


FIG 8 A comparison of the motion difference (MD), dynamic objects (DOs), and sky mask mean (SMM) versus the translation error in the G-ICP and LOAM using Data1. The dashed line indicates the threshold for outliers. (a)–(c) The labels of the typical challenging scenarios for the LO methods. (d) The scenario with dynamic vehicles for the LO methods. (e) and (f) The skymask of the challenging scenarios for the LO methods. The background image of Figure 8(e) and (f) is captured from Google Earth software.

Data2—Factor 2—Dynamic Objects

The scenes of heavy traffic and numerous cars are shown in Figure 9(c) and (d). It shows that the dynamic objects have a significant impact on the error.

Data2—Factor 3—Degree of the Urbanization (Sky Mask)

As displayed in Figure 9(e) and (f), the rapid changing of the sky mask resulted in accuracy loss of the LO methods.

Discussions

There are still many open issues for existing LO methods under highly dynamic urban canyons. Table 5 summarizes each LO method's performance, while Table 6 lists the computational efficiency. Given a real-time constrained approach, LOAM provided the most precise and robust performance across the two data sets. Regardless of the computation cost, G-ICP demonstrated the most competitive performance among the point-registration methods on challenging urban scenes.

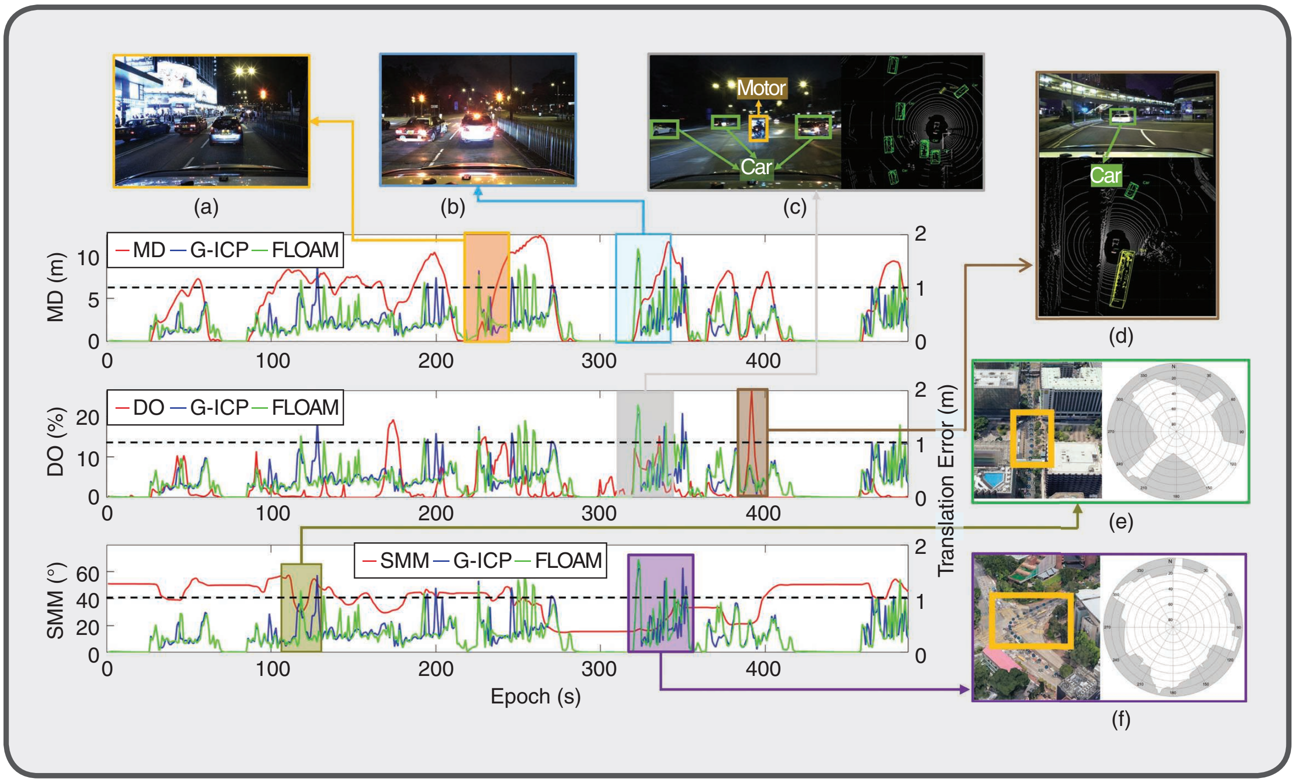


FIG 9 A comparison of the motion difference (MD), dynamic objects (DOs), and sky mask mean (SMM) versus the translation error in the G-ICP and fast LOAM (FLOAM) using Data2. The dashed line indicates the threshold for outliers. (a)–(d) The labels of the typical challenging scenarios for the LO methods. (e) and (f) The skymask of the challenging scenarios for the LO methods. The background image of Figure 9(e) and (f) is captured from Google Earth software.

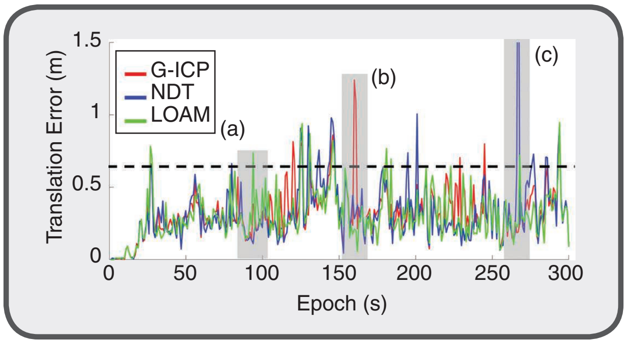


FIG 10 A performance comparison of the G-ICP (red), NDT (blue), and LOAM (green) in Data1. The dashed line indicates the threshold for outliers (~0.7 m). In (a)–(c), the rectangles represent the labels of the performances' divergence among the LO methods.

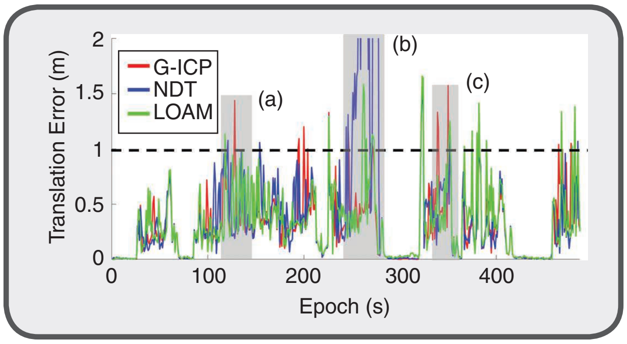


FIG 11 A performance comparison of G-ICP (red), NDT (blue), and LOAM (green) in Data2. The dashed line indicated the threshold for outliers (~1 m). In (a)–(c), the rectangles represent the labels of the performances' divergence among the LO methods.

Another performance comparison of the three representative and widely used LO methods, G-ICP, NDT, and LOAM, is shown in Figures 10 and 11. Considering the impact factors, LOAM is prone to be affected by large motion difference, as depicted in Figures 10(a) and 11(b) (the motion difference can be seen in Figures 8 and 9).

The interquartile range is useful to find the outliers with a large number of data.

In numerous dynamic object scenes in Figure 10(a) and (c), LOAM and the NDT did not perform as accurately as the G-ICP methods. The performance of G-ICP is more sensitive to sky-mask change in Figures 10(b) and 11(a) and (c). Therefore, the user can consider a suitable or mix LO approaches to tackle the challenges of different environments in deploying LO methods for state estimation in a large-scale application in urban canyons.

Furthermore, the evaluation results of Data2 indicate that the *z*-axis has a larger error compared to those in XY-directions, as shown in Figure 12. One possible reason for this is that the horizontal field of view of the Velodyne HDL-32E lidar is much wider than that which is in the vertical direction, e.g., 360° and 40° in the horizontal and vertical directions, respectively. It could be challenging to provide state constraints if the vehicle is suffering more up and down movements. As a result, the performance of LO is still challenged in the area with several layers of the viaduct.

Conclusions and Future Work

In this article, we presented a benchmark comparison and error analysis of publicly available and popular LO methods based on two challenging data sets collected in the urban canyons of Hong Kong. The results are presented in the “Experiments” section, and Figure 13 suggested that the point-wise LO methods can be improved upon with additional computation load, such as G-ICP. The voxelization from VGICP can reduce the computational load of neighboring point searching but might also lead to accuracy loss in state estimation. Feature-wise methods show both accuracy and cost-efficiency based on the extracted feature points. However, the performance of both methods is affected by the aforementioned three kinds of dominant factors. According to our experiments, we suggest that combining both the feature- and point-wise methods can be a promising solution. First, the feature-wise method can efficiently provide a coarse odometry estimation. Then, the coarse one can be applied as an initial guess for the point-wise point cloud registration, which relies heavily on the initial guess,

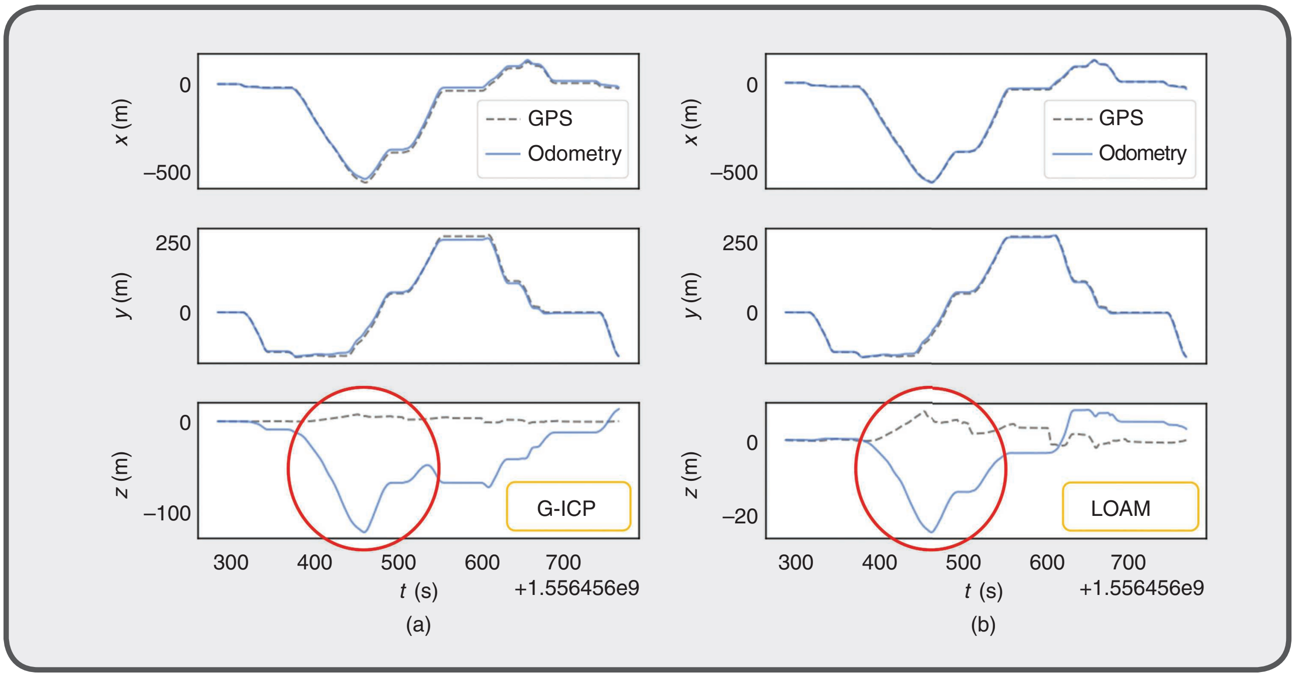


FIG 12 A trajectory comparison of the (a) G-ICP and (b) LOAM versus the ground truth in Data2 in terms of the XYZ direction. A large difference between the LO and the ground truth on the *z*-axis is marked in the red circle.

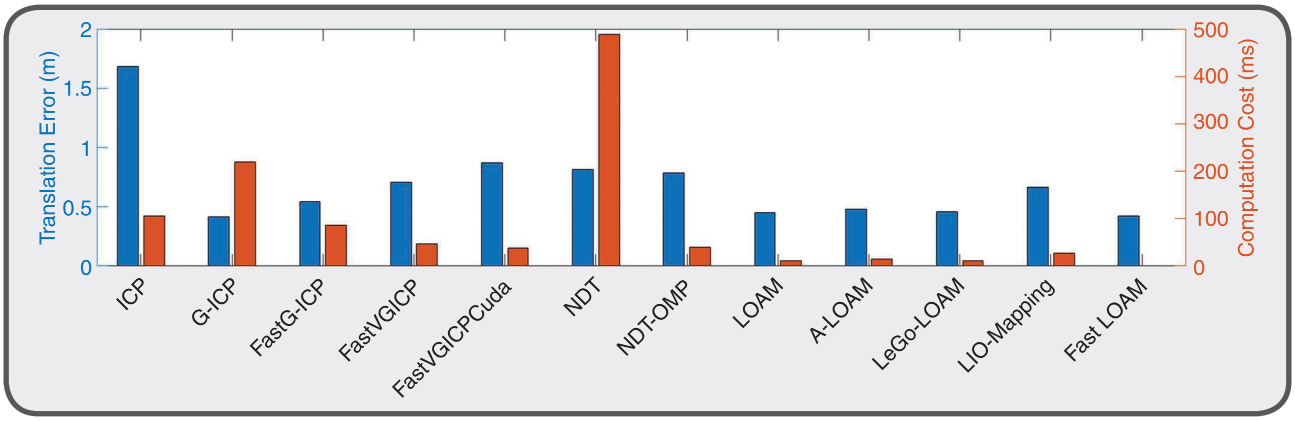


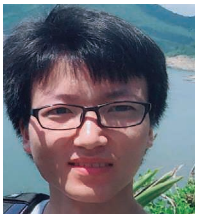
FIG 13 A comparison of the publicly available LO algorithms in Data2 in terms of the translation error and average computation cost (scan to scan).

leading to a coarse-to-fine LO pipeline. However, it is important to note that the feature-wise method might also provide an erroneous estimation. In this case, a fault detector such as the degeneration factor [53] should be used.

In the future, we will study the combination of the point and feature wise to improve the performance of existing LOs. Moreover, we will study the degeneration of LOs in feature-insufficient environments and its integration with other sensors, like GNSS and IMUs. We will also investigate the performance of NN-based feature-extraction LO as it has the potential for accurate feature extraction and scene segmentation. Finally, the performance of LO methods under adverse weather conditions [54], [55] in urban areas is a good direction for further study.

About the Authors

Feng Huang earned his M.Sc. degree in electronic engineering from the Hong Kong University of Science and Technology in 2016, where he is currently a Ph.D. student in the Department of Aeronautical and Aviation Engineering, Hong Kong, QR828, China. His research interests include localization and sensor fusion for autonomous driving. He is a Student Member of IEEE.



Weisong Wen. earned his Ph.D. degree in mechanical engineering from the Hong Kong Polytechnic University. He is currently a research assistant professor in the Department of Aeronautical and Aviation Engineering, Hong Kong Polytechnic University, Hong Kong, QR828, China. His research interests include multisensor integrated localization for autonomous vehicles, simultaneous localization and mapping, and global navigation satellite systems positioning in urban canyons.

Jiachen Zhang earned her bachelor's degree in information engineering from Tianjin University, China, in 2016, where she is a graduate student in optical engineering. She is currently a research assistant with the Intelligent Positioning and Navigation Laboratory in the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong, QR828, China. Her research interests include localization and sensor fusion for autonomous driving.

Li-Ta Hsu (lt.hsu@polyu.edu.hk) earned his Ph.D. degree in aeronautics and astronautics from National Cheng Kung University, Tainan, Taiwan, in 2013. He is currently an assistant professor in the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong, QR828, China. His research interests include global navigation satellite systems positioning in challenging environments and localization for pedestrian, autonomous driving, and unmanned aerial vehicles. He is a Member of IEEE.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. 2012 IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[2] C. Wang et al., "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 109–116. doi: 10.1109/IROS.2017.8202145.

[3] C. Liu and S. Shen, "An augmented reality interaction interface for autonomous drone," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 24 Oct.–24 Jan. 2021, pp. 11,419–11,424. doi: 10.1109/IROS45743.2020.9341037.

[4] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, 2016. doi: 10.1109/JIOT.2016.2612119.

[5] L.-T. Hsu, "Analysis and modeling GPS NLOS effect in highly urbanized area," *GPS Solutions*, vol. 22, no. 1, p. 7, 2018. doi: 10.1007/s10291-017-0667-9.

[6] W. Wen, G. Zhang, and L.-T. Hsu, "GNSS NLOS exclusion based on dynamic object detection using LiDAR point cloud," *IEEE Trans.*

Intell. Transp. Syst., vol. 22, no. 2, pp. 853–862, Feb. 2021. doi: 10.1109/TITS.2019.2961128.

[7] L.-T. Hsu, Y. Gu, and S. Kamijo, "NLOS correction/exclusion for GNSS measurement using RAIM and city building models," *Sensors*, vol. 15, no. 7, pp. 17,329–17,349, 2015. doi: 10.3390/s150717329.

[8] W. Wen, G. Zhang, and L. T. Hsu, "Correcting NLOS by 3D LiDAR and building height to improve GNSS single point positioning," *Navigation*, vol. 66, no. 4, pp. 705–718, 2019. doi: 10.1002/navi.335.

[9] X. Bai, W. Wen, and L.-T. Hsu, "Using Sky-pointing fish-eye camera and LiDAR to aid GNSS single-point positioning in urban canyons," *IET Intell. Transp. Syst.*, vol. 14, no. 8, pp. 908–914, 2020. doi: 10.1049/iet-its.2019.0587.

[10] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *J. Field Robot.*, vol. 24, no. 10, pp. 803–827, 2007. doi: 10.1002/rob.20204.

[11] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, 2017. doi: 10.1007/s10514-016-9548-2.

[12] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proc. 2003 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS 2003)(Cat. No. 03CH37453)*, 2003, vol. 3, pp. 2743–2748. doi: 10.1109/IROS.2003.1249285.

[13] W. Wen, L.-T. Hsu, and G. Zhang, "Performance analysis of NDT-based graph SLAM for autonomous vehicle in diverse typical driving scenarios of Hong Kong," *Sensors*, vol. 18, no. 11, p. 3928, 2018. doi: 10.3390/s18113928.

[14] Y.-C. Kan, L.-T. Hsu, and E. Chung, "Performance evaluation on map-based NDT scan matching localization using simulated occlusion datasets," *IEEE Sensors Lett.*, vol. 5, no. 3, pp. 1–4, 2021. doi: 10.1109/LSENS.2021.3060097.

[15] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992. doi: 10.1109/34.121791.

[16] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot., Sci. Syst.*, vol. 2, no. 4, pp. 435, 2009.

[17] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. 2021 IEEE Int. Conf. Robot. Automat, (ICRA)*, pp. 11054–11059, doi: 10.1109/ICRA48506.2021.9560835.

[18] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, 2019. doi: 10.1177/1729881419841532.

[19] T. Q and C. Shaozu, "Advanced implementation of LOAM," GitHub. Accessed: June 1, 2020. [Online.] https://github.com/HKUST-Aerial-Robotics/A-LOAM

[20] S. Agarwal and K. Mierle, "Ceres solver," 2012. [Online]. Available: http://ceres-solver. org

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

[22] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *Proc. 2019 Int. Conf. Robot. Automat. (ICRA)*, pp. 3144–3150. doi: 10.1109/icra.2019.8793511.

[23] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 2095–2101. doi: 10.1109/ICRA40945.2020.9196764.

[24] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical Analysis*, G. A. Watson, Eds. New York NY, USA: Springer-Verlag, 1978, pp. 105–116.

[25] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman, "Point registration via efficient convex relaxation," *ACM Trans. Graph. (TOG)*, vol. 35, no. 4, pp. 1–12, 2016. doi: 10.1145/2897824.2925913.

[26] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proc. Object Recognition Supported by User Interaction for Service Robots*, 2002, vol. 3, pp. 545–548. doi: 10.1109/ICPR.2002.1047997.

[27] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proc. 2015 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 742–749. doi: 10.1109/IROS.2015.7353455.

[28] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 9–11, 1991, vol. 3, pp. 2724–2729. doi: 10.1109/ROBOT.1991.132043.

[29] B. Huhle, M. Magnusson, W. Straßer, and A. J. Lilienthal, "Registration of colored 3D point clouds with a kernel-based extension to the normal distributions transform," in *Proc. 2008 IEEE Int. Conf. Robot. Automat.*, pp. 4025–4030. doi: 10.1109/ROBOT.2008.4543829.

[30] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3D scan-matching algorithms," in *Proc. 2015 IEEE Int. Conf. Robot. Automat. (ICRA)*, pp. 3631–3637. doi: 10.1109/ICRA.2015.7139703.

[31] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. 2009 IEEE Int. Conf. Robot. Automat.*, pp. 3212–3217. doi: 10.1109/ROBOT.2009.5152473.

[32] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Understand.*, vol. 125, pp. 251–264, Aug. 2014. doi: 10.1016/j.cviu.2014.04.011.

[33] B. Lv, H. Xu, J. Wu, Y. Tian, S. Tian, and S. Feng, "Revolution and rotation-based method for roadside LiDAR data integration," *Optics Laser Technol.*, vol. 119, p. 105,571, Nov. 2019. doi: 10.1016/j.optlastec.2019.105571.

[34] J. Wu, H. Xu, and W. Liu, "Points registration for roadside LiDAR sensors," *Transp. Res. Rec.*, vol. 2673, no. 9, pp. 627–639, 2019. doi: 10.1177/0361198119843855.

[35] Z. Zhang, J. Zheng, H. Xu, X. Wang, X. Fan, and R. Chen, "Automatic background construction and object detection based on roadside LiDAR," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 10, pp. 4086–4097, 2019. doi: 10.1109/TITS.2019.2936498.

[36] "The KITTI visual odometry/SLAM evaluation 2012," Karlsruhe Institute of Technology. Accessed: June 1, 2020. [Online.] http://www.cvlibs.net/datasets/kitti/eval_odometry.php

[37] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," 2020, arXiv:2006.12567.

[38] W. Wang et al., "PointLoc: Deep pose regressor for LiDAR point cloud localization," 2020, arXiv:2003.02392.

[39] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions (ICARSC)*, Apr. 25–27, 2018, pp. 71–77. doi: 10.1109/ICARSC.2018.8374163.

[40] Q. Li et al., "LO-Net: Deep real-time lidar odometry," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, June 15–20, 2019, pp. 8465–8474. doi: 10.1109/CVPR.2019.00867.

[41] Y. Cho, G. Kim, and A. Kim, "Unsupervised geometry-aware deep LiDAR odometry," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 31 May–31 Aug. 2020, pp. 2145–2152. doi: 10.1109/ICRA40945.2020.9197366.

[42] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*, F. L. Bauer, Eds. New York, NY, USA: Springer-Verlag, 1971, pp. 134–151.

[43] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," M.S. thesis, Örebro universitet, Örebro, Sweden, 2009.

[44] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. 2016 IEEE Int. Conf. Robot. Automat. (ICRA)*, pp. 1271–1278. doi: 10.1109/ICRA.2016.7487258.

[45] M. Elhousni and X. Huang, "A survey on 3D LiDAR localization for autonomous vehicles," in *Proc. 2020 IEEE Intell. Veh. Symp. (IV)*, pp. 1879–1884. doi: 10.1109/IV47402.2020.9304812.

[46] L.-T. Hsu et al., "UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *Proc. 34th Int. Tech. Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, St. Louis, MO, USA, Sep. 2021, pp. 226-256. [Online]. Available: https://doi.org/10.33012/2021.17895.

[47] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 573–580. doi: 10.1109/IROS.2012.6385773.

[48] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017, pp. 205–284.

[49] H. Hofmann, H. Wickham, and K. Kafada, "Letter-value plots: Boxplots for large data," *J. Comput. Graph. Stat.*, vol. 26, no. 3, pp. 469–477, 2017, doi: 10.1080/10618600.2017.1305277.

[50] E. Li, S. Wang, C. Li, D. Li, X. Wu, and Q. Hao, "SUSTech POINTS: A portable 3D point cloud interactive annotation platform system," in *Proc. 2020 IEEE Intell. Veh. Symp. (IV)*, pp. 1108–1115. doi: 10.1109/IV47402.2020.9304562.

[51] W. Wen et al., "UrbanLoco: A full sensor suite dataset for mapping and localization in urban scenes," presented at the ICRA 2020, Paris, France, 2019.

[52] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Wiley Interdisciplinary Rev., Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 73–79, 2011. doi: 10.1002/widm.2.

[53] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *Proc. 2016 IEEE Int. Conf. Robot. Automat. (ICRA)*, pp. 809–816. doi: 10.1109/ICRA.2016.7487211.

[54] A. Carballo et al., "LIBRE: The multiple 3D LiDAR dataset," in *Proc. 2020 IEEE Intell. Veh. Symp. (IV)*, pp. 1094–1101. doi: 10.1109/IV47402.2020.9304681.

[55] J. Wu, H. Xu, Y. Tian, R. Pi, and R. Yue, "Vehicle detection under adverse weather from roadside LiDAR data," *Sensors*, vol. 20, no. 12, p. 3433, 2020. doi: 10.3390/s20123433.