2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

LiTAMIN: LiDAR-based Tracking And MappINg by Stabilized ICP for Geometry Approximation with Normal Distributions

Masashi Yokozuka[1], Kenji Koide[1], Shuji Oishi[1] and Atsuhiko Banno[1]

Abstract— **This paper proposes a 3D LiDAR simultaneous localization and mapping (SLAM) method that improves accuracy, robustness, and computational efficiency for an iterative closest point (ICP) algorithm employing a locally approximated geometry with clusters of normal distributions. In comparison with previous normal distribution-based ICP methods, such as normal distribution transformation and generalized ICP, our ICP method is simply stabilized with normalization of the cost function by the Frobenius norm and a regularized covariance matrix. The previous methods are stabilized with principal component analysis, whose computational cost is higher than that of our method. Moreover, our SLAM method can reduce the effect of incorrect loop closure constraints. The experimental results show that our SLAM method has advantages over open source state-of-the-art methods, including LOAM, LeGO-LOAM, and hdl_graph_slam.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a basic technology for autonomous mobile robots. In particular, SLAM with light detection and ranging (LiDAR) is widely used because it exhibits stable performance in both indoor and outdoor environments. SLAM should be continuously improved for better accuracy, robustness, and computational efficiency.

SLAM methods with LiDAR are divided into two categories: matching-based and feature-based. Matching-based methods [1]–[4] employ geometric registration techniques, such as iterative closest point (ICP) and normal distribution transformation (NDT). These methods provide accurate position estimation by directly using scanned points. They are, however, not computationally efficient because they use a huge number of points for stable registration.

Feature-based methods [5]–[7] are actively studied for their computational efficiency; these methods extract features and use geometric primitives, such as line segments and planes. However, these methods become inaccurate and unstable when the geometric features in the environment are insufficient.

This paper describes a matching-based SLAM method that improves the computational efficiency and the robustness for ICP. We consider that improvement of the stability of the ICP algorithm is a critical issue for enhancing the overall performance of matching-based SLAM methods. Our proposed method, LiDAR-based Tracking And MappINg (LiTAMIN), approximates a geometric shape using normal distributions. This method, instead of using the point cloud obtained from LiDAR as it is, modifies the point cloud to reduce the number of points and make the density more uniform; this results in a faster and more stable SLAM. We propose a new cost function for ICP so that the minimization can be performed effectively and stably. Also, we introduce an index for reducing the effect of outliers in loop detection by LiDAR SLAM. Our experimental results show the improvements achieved by LiTAMIN in comparison with state-of-the-art methods. Figure 1 is an example mapping result by LiTAMIN.

[1]Authors are with the Robot Innovation Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Japan yokotsuka-masashi@aist.go.jp

This work was supported in part by the New Energy and Industrial Development Organization (NEDO).

II. Related work

ElasticFusion [3] and Surfel-based Mapping (SuMa) [4] are two SLAM methods based on point-to-plane ICP [8]. Point-to-plane ICP is an appropriate method when the point density is low because it deals with the distance between a point and a plane [8] instead of the distance between points. To apply point-to-plane ICP, SLAM systems require information on directions normal to the map geometry and computation of correspondences between points and planes. One of the characteristics of ElasticFusion and SuMa is the map representation using many surfels, which are small disks, instead of using complicated polygon meshes. Implementation of these methods is simple because the point correspondence computation can be performed with techniques similar to those of standard ICP. However, the method is not computationally efficient because processing for many surfels requires a GPU, and each surfel requires normal direction estimation by using principal component analysis (PCA), which has computational complexity.

Hdl_graph_slam [2] is a method using generalized ICP (GICP) [9]. GICP can deal with points, line segments, and planes flexibly by representing geometry with a set of normal distributions, while the point-to-plane ICP deals only with point-and-plane pairs. The merit of GICP is that it enables uniform and general processing for geometric registration by using covariance matrices without additional functions, such as line-segment or plane detection. However, GICP requires additional processing for stability because the representation of line segments and planes by a covariance matrix is a degenerate case.

LiDAR odometry and mapping (LOAM) [5] is an early feature-based method with LiDAR-based SLAM; it is similar to feature-based visual SLAM, such as ORB-SLAM [10]. This method detects geometric primitives by evaluating the smoothness of a local region. Areas with low smoothness are detected as edges, and areas with high smoothness

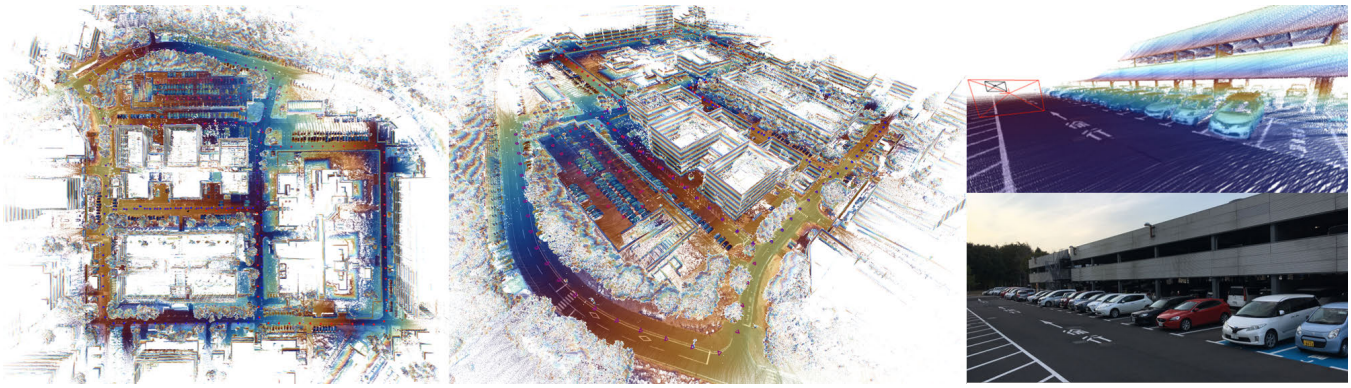

Fig. 1. Example mapping result by LiTAMIN. This map was built from the Segway dataset described in Section VI.

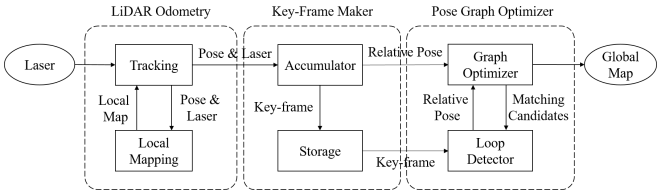

Fig. 2. System overview.

are detected as planes. However, the method cannot detect appropriate features when the scene contains too many small objects, such as vegetation and trees, because detection of an enormous number of features increases the computation cost. Likewise, when only a few simple objects, such as large planes, are dominant, the method cannot detect cues for correct registration.

Lightweight and ground-optimized LOAM (LeGO-LOAM) [6] improved the computational efficiency and the stability of LOAM by adding segmentation processing that carefully selects features and reduces the number of geometric primitives. LeGO-LOAM performs feature extraction with the assumption that a robot is always on a ground surface. The method extracts the ground surface from scan data, and performs segmentation for the remaining regions; appropriate features for the registration are extracted from these regions. Even if only the ground plane is detected, the changes in the roll, pitch, and z-value can be estimated. Other features can be used to estimate the x-y translation and the yaw angle of the robot. This processing can obtain a suitable number of features stably by using segmentation to remove regions that are too small. However, when the assumption does not hold, such as on uneven ground, and no suitable size segments can be identified, the method is not functional.

III. System overview

Figure 2 provides an overview of our proposed SLAM system, LiTAMIN. The LiDAR Odometry block contains two threads: a pose tracking thread by our ICP method and a local mapping thread using results of the tracking thread. This block continuously computes the self-position independently of the local mapping thread.

LiDAR odometry block does not use the global map to avoid scan drop-outs, but uses the local map which contains inconsistencies by the odometry. The map update-cycle is constant since the local mapping thread does not perform loop closure. Our system reduces the effect of inconsistency by removing the old points from the local map. The local map is built with accumlating every localized scans by the tracking thread for dealing with sparseness of one scan.

The Key-Frame Maker block outputs local maps and relative poses between the local maps, while accumulating the LiDAR odometry and setting the key-frames every 10 m. The memory usage is reduced by writing the key-frames to storage, such as a hard drive or solid-state drive.

We consider the trajectory computed by the tracking thread is accurate enough for a 10 m travel range, which is the distance of accumulation of scans for building a key-frame. Our system deformes the global map with the unit of the key-frame after loop closing on the pose-graph.

The Pose Graph Optimizer block corrects the recent relative poses between key-frames and detects loops in the pose graph. When it detects the loop candidates, the optimizer reads the necessary key-frames from storage. Loop detection lists all the key-frames within a 30-m radius from the current position as the loop candidates, and then applies the ICP-based loop closure processing.

Our system applyies ICP to every loop candidates. Sometimes the loop detector thread is delayed against the odometry block for the ICP processings. Since the odometry computation is independent of the global map, the delay does not affect the total computation results, although the global mapping is possibly delayed. After applying ICP, our system inserts the all relative poses, which are included errors, into the pose-graph; our system elminates the wrong poses on the pose-graph optimization.

IV. Fast and Stable ICP

In the SLAM system, which requires real-time processing, ICP methods have to balance accuracy and robustness to obtain computational efficiency. The standard ICP and other robust methods [11]–[13] directly employ point clouds. Fine initial solutions and uniform point density in the point clouds

TABLE I: Comparison of ICP variants for local approximation with a cluster of normal distributions.

Method	Map representation	Point association	Degeneracy avoidance	Cost function
Standard ICP	k-d tree	k-d tree	not required	$\sum_i (q_i - (Rp_i + t))^T (q_i - (Rp_i + t))$
NDT	voxel	voxel	PCA	$\sum_i (q_i - (Rp_i + t))^T C_i^{-1} (q_i - (Rp_i + t))$
Generalized ICP	k-d tree	k-d tree	PCA	$\sum_i (q_i - (Rp_i + t))^T (C_i^q + RC_i^p R^T)^{-1} (q_i - (Rp_i + t))$
LiTAMIN (proposed method)	voxel	k-d tree	not required	$\sum_i (q_i - (Rp_i + t))^T \frac{w_i (C_i + \lambda I)^{-1}}{\|(C_i + \lambda I)^{-1}\|_F} (q_i - (Rp_i + t))$

are desirable for these methods. Although some robust and accurate ICP methods [14]–[16] can ensure global optima without an initial solution, they have high computational costs. These methods are not practical for a SLAM that requires constant ICP processing for every frame. Reduction of the number of 3D points is one of the most effective solutions for improving the computational efficiency. Many ICP-based SLAM systems [9], [17]–[20] often use ICP methods with voxel grids and normal distributions because they can reduce the computational cost while still retaining enough geometric information. Among them, NDT [17] and GICP [9] are the most popular methods.

Our objective was to improve the accuracy and the robustness of these normal distribution-based methods while achieving a computational efficiency that is comparable to feature-based methods. Table I indicates the differences between our ICP method and the others.

A. Map representation and point association

Voxel grids or k-d trees are used for map representation and correspondence searching in SLAM systems. Voxel grid representation has an advantage in computational efficiency because the number of voxels is significantly lower than the number of points in the original point cloud. In contrast, k-d tree representation has an advantage in accuracy and robustness for registration. With respect to finding the corresponding points, k-d tree representation can find the association points with a nearest neighbor (NN) search, while voxel grid representation has no guarantee for the NN search. With regard to computational cost, voxel grid representation has an advantage in the correspondence search because the cost of computation with voxel grids is $O(N)$ and that with k-d trees is $O(Nlog(N))$. LiTAMIN combines the merits of the two representations. Our system represents LiDAR data as a reduced number of point sets by voxel filtering, where each voxel is represented by a single point, specifically the center of mass of the 3D points included in the voxel. The map is also represented by voxel grids for reducing the number of the points and making the density more uniform. The point associations are determined by the NN search with k-d tree representation. We implemented a voxel size of 1 m; therefore, the size of a local map corresponding to a keyframe is 200 m $\times$ 200 m $\times$ 40 m.

B. Cost function and degeneracy avoidance

LiTAMIN adopted an ICP with local geometry approximated by normal distributions in a manner similar to that in NDT [17] and GICP [9], which should cope with the degeneracy of covariance matrices. If the local geometry is a plane, the minimum eigenvalue of the covariance matrix is 0 or extremely small; thus, the cost functions of NDT and GICP in Table I diverge with the degenerate covariance matrices. Some NDT-based methods apply PCA and change the representation to point-to-plane distance metrics if the covariance matrix does not have an inverse. GICP uses covariance matrix C by applying the following transformation after PCA:

$$C := V \operatorname{diag}(1, 1, \varepsilon) V^T, \tag{1}$$

where V is a matrix with arranged eigenvectors given by PCA and $\operatorname{diag}(\cdots)$ is a diagonal matrix with arranged eigenvalues. GICP sets the eigenvalues to 1 except for the minimum eigenvalue, and replaces the minimum with a small value ε that does not create computational problems.

However, this stabilization technique by PCA is not suitable for fast computation because applying PCA to all voxels has a high computational cost. To reduce the cost, we propose the following covariance transformation:

$$C^{-1} := \frac{w(C + \lambda I)^{-1}}{\|(C + \lambda I)^{-1}\|_F}, \tag{2}$$

where w, I, and λ are the weight, identity matrix, and constant, respectively, and $\|\cdot\|_F$ indicates the Frobenius norm. w indicates the weight of point q; this weight is for an alternative because the magnitude of the transformed covariance matrix elements does not correspond to accuracy. We set w by occupancy probability [21] for each voxel. We set λ as 10^{-6} empirically, because this number corresponds to a normal distribution with a standard deviation of 1 mm; this number is not expected to affect the ICP results because LiDAR's range of measurement error is several centimeters. Replacing the eigenvalues with $\operatorname{diag}(1, 1, \varepsilon)$ in GICP indicates that the magnitude of eigenvalues does not affect the accuracy of ICP results; the degeneracy direction of covariance matrices plays an important role in the accuracy of GICP. From this consideration, we normalize the covariance matrix by the Frobenius norm because scaling a matrix with eigenvalues does not affect the geometric registration. The Frobenius norm, which indicates the scale of the matrix by the sum of squares of the eigenvalues, is defined as follows:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\sum_{i=1}^{n} \sigma_i^2}. \tag{3}$$

V. Robust loop closure

ICP-based loop closure should assume failures because ICP has no guarantee of global optima. Recent robust pose graph optimization methods can be categorized to two types: methods with initial guesses by odometry [22]–[27] and methods without initial guesses [28]–[30]. The methods with initial guesses detect outliers of loop constraints when a pose graph exceeds an assumed odometry error after adding constraints. These methods include approaches that use iterative reweighted least squares [22]–[24], approaches that use the χ^2 test [25], [26], and an approach that detects outliers by using a Gaussian mixture model [27]. The methods without initial guesses detect outliers by convex programming. Our method uses the approach that employs iterative reweighted least squares. Further, we propose a simple intuitive weighting method.

A. Cost function

The cost function in our method for pose graph optimization is the following:

$$E = \sum_{i,j} w_{ij} \left(\| R_i R_j^T \cdot \Delta R_{ij}^T - I \|_F^2 + \| (t_i - t_j) - \Delta t_{ij} \|_2^2 \right), \quad (4)$$

where i, j are key-frame indices, w_{ij} is weight, R_i is rotation, t_i is translation, ΔR_{ij} is relative rotation, Δt_{ij} is relative translation, and $\| \cdot \|_2$ indicates the L2 norm. Although ΔR_{ij} and Δt_{ij} are provided by the ICP results, the values have possible outliers because ICP has no guarantee of global optima. In consideration of the errors $e_R = R_i R_j^T \cdot \Delta R_{ij}^T - I$ and $e_t = (t_i - t_j) - \Delta t_{ij}$, our method addresses the problem with the following weight computation:

$$w_{ij} = \sqrt{\left(1 - \frac{\| e_t \|_2}{\| e_t \|_2 + \sigma_t} \right) \left(1 - \frac{\| e_R \|_F}{\| e_R \|_F + \sigma_R} \right)}, \quad (5)$$

where σ_t and σ_R indicate the tolerated error of translation and rotation, respectively; when e_t and e_R are large, w_{ij} becomes zero. Because the values of σ_t and σ_R depend on the accuracy of ICP, we determined the number by considering the error when ICP outputs optimum values. In the optimum situation, we considered the following small change of a rotation matrix by employing the angular velocity $\omega = (\omega_x, \omega_y, \omega_z)^T$:

$$\frac{\mathrm{d}R}{\mathrm{d}t} = I + \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \quad (6)$$

When the rotation matrix $R_i R_j^T \cdot \Delta R_{ij}^T$ is small enough, because the diagonal elements of e_R become zero, the error e_R has only non-diagonal elements, and $\| e_R \|_F$ can be approximated as

$$\| e_R \|_F \approx \sqrt{2} \| \omega \|_2. \quad (7)$$

Because we see $\| \omega \|_2$ as a rotation angle with an arbitrary axis, we concluded that the accuracy of the rotation angle for ICP results is within 3 degrees of the optimum value, and we set the value as $\sigma_R = 3\sqrt{2}$ degrees. Similarly, because we consider the accuracy of translation to be about 10 cm, we set the value as $\sigma_t = 0.1$ m. When e_R and e_t are larger than the tolerated error, w_{ij} approaches zero. Our method can remove the failure results of ICP as outliers because the corresponding constraint does not affect the cost function for a weight of zero.

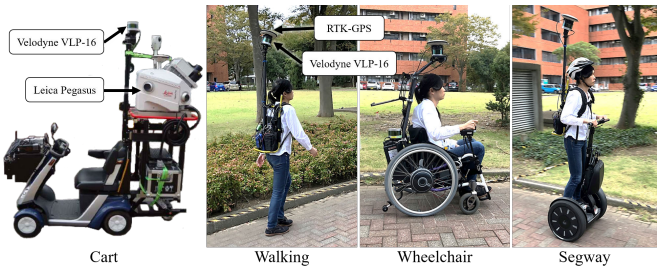

Fig. 3. Experimental devices and conditions.

B. Re-weighting schedule

We can reduce the wrong loop-closure constraints with weighting; however, our method may not close the loops, even if the ICP results are correct, because the initial error is potentially much larger than the tolerated error when closing loops. For this reason, our method used adaptive weights in the optimization iterations; we set $w_{ij} = 1$ at the first iteration, and the weight according to Eq. (5) at the second and subsequent iterations. Furthermore, our method incrementally optimizes the cost function by adding constraints one by one to avoid local optima. Our method tries to test the correctness of constraints by weighting of the first iteration. If the constraint is correct, that is, the errors e_R and e_t are within tolerated values, the constraint becomes active with a non-zero weight after the second iteration.

VI. Experiment

In this experiment, we evaluated the accuracy of trajectories generated by LiTAMIN and other state-of-the-art methods and compared them with the ground truth trajectory obtained from the experimental devices shown in Fig. 3. We obtained four evaluation datasets with changing mobility using a cart, walking, a wheelchair, and a Segway. Each SLAM method reconstructed the trajectories using a Velodyne VLP-16 LiDAR. We also used the precision survey instruments, the Leica Pegasus system and a real-time kinematic GPS (RTK-GPS), to obtain ground truth data. The accuracy of Leica Pegasus is 2 cm according to the manufacturer specifications. In this experiment, four other SLAM methods were compared with LiTAMIN: PCA method, hdl_graph_slam, LeGO-LOAM, and LOAM. The PCA method replaced Eq. (2) in LiTAMIN with Eq. (1). Hdl_graph_slam [2] is a GICP-based method, and LeGO-LOAM [6] and LOAM [5] are feature-based methods. For this experiment, the evaluation was performed using a laptop with an Intel Core i7-7820HQ processor. The error metric for the evaluation was the root mean square error (RMSE),

calculated as

$$\text{Overall RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} |\boldsymbol{g}_t - \boldsymbol{p}_t|_2^2}, \tag{8}$$

where t is the time index, $\boldsymbol{g}_t = (X_t, Y_t, Z_t)^T$ is the ground truth trajectory, and $\boldsymbol{p}_t = (x_t, y_t, z_t)^T$ indicates the trajectory estimated by each SLAM method. Additionally, we evaluated the height error,

$$\text{Height RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} |Z_t - z_t|_2^2}, \tag{9}$$

with respect to the ground plane,

$$\text{2D RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} |(X_t, Y_t)^T - (x_t, y_t)^T|_2^2}, \tag{10}$$

and errors in a segment cut from a trajectory,

$$\text{RMSE per } X \text{ m} = \sqrt{\frac{1}{(T_2 - T_1)}\sum_{t=T_1}^{T_2} |\boldsymbol{g}_t - \boldsymbol{x}_t|_2^2}, \tag{11}$$

where $\sum_{t=T_1}^{T_2-1} \sqrt{|\boldsymbol{g}_{(t+1)} - \boldsymbol{g}_t|_2^2} = X$. The RMSE per X (in meters) was computed by cutting a segment with a length X from the trajectory and sliding it along the trajectory. For this reason, the RMSE per X can be computed for more than one sample from one trajectory, and this metric can be used to obtain the average and standard deviation. We evaluated the height RMSE and 2D RMSE to confirm the direction in which the error is larger. We considered the RMSE per X with the short segment to indicate the accuracy of pose tracking; the long segment indicates the drift of localization results over a longer time span. Moreover, RMSE per X can evaluate the accuracy of each method stably even if the method fails position tracking locally because the error is evaluated by the segments. A similar error metric was proposed by Zhang et al. [31].

Table II indicates the results of error evaluation. In the table, red results indicate the best accuracy, blue results indicate the second-best accuracy, and bold results indicate the third-best accuracy for each error metric. Figure 4 shows the ground truth trajectories and the trajectories estimated by each method. Figure 5 shows the mapping results by LiTAMIN. Table III reports the computational time for each method, as well as the actual travel time. The results were obtained from the total computation time for building a map using all VLP-16 data-frames without frame drops and thread sleep. The font colors and bold rank the computation time in the same way as done in Table II. Table IV indicates the average computation time for each function in our SLAM system per execution function.

In order to evaluate the accuracy of LiTAMIN for a big loop case, we conducted an experiment in a larger environment with the same experimental device of Segway dataset. Figure 6 shows the trajectories by each method and the ground truth; Figure 7 shows the mapping result by LiTAMIN superimposed on an aerial photograph. Table V indicates the results of error evaluation. LeGO-LOAM could not close loops in this experiment for the large error.

VII. Discussion

Table II shows that LiTAMIN is more accurate than the other methods for most evaluation metrics, regardless of whether loop closure detection is performed. LiTAMIN applied with Eq. (2) is more accurate than the PCA method applied with the GICP stabilization method of Eq. (1), except for the wheelchair dataset. Especially, LiTAMIN indicates stable accuracy, while the PCA method showed significant accuracy reduction in the Segway dataset. We consider our stabilization method to be more accurate and robust than the GICP stabilization method because our method holds the shape of normal distribution, in contrast with the GICP method, which changes the shape by $\text{diag}(1,1,\varepsilon)$ in Eq. (1).

In comparison with the feature-based methods LeGO-LOAM and LOAM, ICP-based methods, including LiTAMIN, PCA, and hdl_graph_slam, were more accurate in short segments measuring 1 m and 3 m. This result shows that ICP-based methods have higher accuracy than feature-based methods for local pose tracking. We expect ICP-based methods to build clearer maps than feature-based methods because the accuracy of local map geometry depends on the local pose-tracking accuracy.

The reduced accuracy for the Segway dataset in Table II indicates that its trajectory was the most difficult to determine. The PCA method and LeGO-LOAM showed worse accuracy in long segments for the Segway dataset. However, these two methods did not have significantly worse results in short segments in comparison with other methods. These results stem from the fact that these two methods lost pose tracking during part of the trajectory. We could evaluate the accuracy of each method with fairness by employing not only overall trajectory error but also short- and long-segment errors.

LiTAMIN with loop closure can be seen to be improved in comparison with LiTAMIN without loop closure for long segments over 100 m; however, the improvement is not as pronounced for short trajectories. This indicates that loop closure detection can improve the consistency of a trajectory, but it cannot improve the accuracy of local geometry. While this trend can be seen in the PCA method and hdl_graph_slam similarly, it indicates that local tracking accuracy should be improved to improve the accuracy of local map geometry. For LeGO-LOAM, trends can be seen in which the accuracy becomes worse after loop closure for the cart and walking datasets. Especially, LeGO-LOAM significantly improved the 2D RMSE for the cart dataset; however, the height RMSE was worse. We considered that LeGO-LOAM with loop closure distorted the trajectory shape along the height direction due to overly strong smoothing because constraints along the height direction in outdoor scenes were few in comparison with 2D constraints.

Table III shows that LiTAMIN, despite being an ICP-based method, had comparable computational efficiency against feature-based LeGO-LOAM. Moreover, LiTAMIN could

TABLE II: Accuracy evaluation for each SLAM method. The marks ✓and − mean with (✓) and without (−) loop closure, respectively, for each method.

Cart Method	Loop closure	Overall RMSE	Height RMSE	2D RMSE	RMSE per 1 m	RMSE per 3 m	RMSE per 10 m	RMSE per 30 m	RMSE per 100 m	RMSE per 300 m	RMSE per 1 km
LiTAMIN	−	**1.462**	**1.154**	0.897	**0.016±0.004**	**0.018±0.007**	**0.021±0.015**	**0.031±0.033**	**0.120±0.043**	**0.331±0.072**	**1.303±0.180**
LiTAMIN	✓	**1.238**	**1.189**	**0.344**	**0.016±0.006**	**0.018±0.011**	**0.022±0.018**	**0.033±0.030**	**0.118±0.034**	**0.315±0.066**	**1.144±0.157**
PCA method	−	1.697	1.467	0.854	**0.017±0.005**	**0.019±0.010**	**0.023±0.019**	**0.034±0.033**	0.129±0.041	0.352±0.071	1.492±0.202
PCA method	✓	**1.442**	**1.384**	**0.405**	0.017±0.007	0.020±0.013	0.024±0.021	0.035±0.034	**0.128±0.040**	**0.352±0.062**	**1.350±0.178**
hdl_graph_slam	−	5.304	4.306	3.096	0.020±0.049	0.027±0.083	0.042±0.012	0.078±0.274	0.320±0.483	1.277±2.434	2.998±0.844
hdl_graph_slam	✓	2.613	1.595	2.071	0.020±0.049	0.027±0.083	0.042±0.012	0.078±0.274	0.317±0.482	1.139±2.465	2.275±0.560
LeGO-LOAM	−	2.691	2.402	1.213	0.041±0.052	0.052±0.052	0.061±0.051	0.076±0.049	0.180±0.050	0.506±0.142	2.325±0.356
LeGO-LOAM	✓	2.933	2.920	**0.278**	0.041±0.052	0.052±0.052	0.061±0.051	0.076±0.049	0.173±0.051	0.471±0.142	2.737±0.464
LOAM	−	2.893	2.732	0.953	0.034±0.012	0.049±0.016	0.061±0.023	0.076±0.044	0.181±0.076	0.426±0.129	2.158±0.443

Walking Method	Loop closure	Overall RMSE	Height RMSE	2D RMSE	RMSE per 1 m	RMSE per 3 m	RMSE per 10 m	RMSE per 30 m	RMSE per 100 m	RMSE per 300 m	RMSE per 1 km
LiTAMIN	−	**0.411**	**0.319**	**0.238**	**0.055±0.173**	0.067±0.223	**0.075±0.015**	**0.087±0.337**	**0.125±0.325**	**0.288±0.198**	**0.350±0.053**
LiTAMIN	✓	**0.395**	**0.302**	**0.238**	0.055±0.176	**0.066±0.226**	**0.074±0.018**	**0.085±0.337**	**0.123±0.326**	0.299±0.200	**0.338±0.053**
PCA method	−	0.442	0.345	0.248	0.056±0.175	0.069±0.226	0.079±0.019	0.091±0.339	0.129±0.324	**0.284±0.204**	**0.364±0.069**
PCA method	✓	0.474	0.392	**0.239**	0.056±0.175	0.068±0.227	**0.076±0.021**	**0.088±0.339**	**0.124±0.324**	**0.296±0.204**	0.410±0.066
hdl_graph_slam	−	3.609	3.310	1.434	**0.047±0.200**	**0.065±0.257**	0.084±0.012	0.122±0.472	0.394±0.525	0.976±0.600	2.956±0.471
hdl_graph_slam	✓	1.815	1.263	1.263	**0.046±0.202**	**0.062±0.254**	0.077±0.012	0.108±0.452	0.280±0.497	1.016±0.418	1.496±0.396
LeGO-LOAM	−	0.595	0.493	0.285	0.073±0.178	0.097±0.222	0.110±0.051	0.126±0.336	0.151±0.321	0.342±0.210	0.479±0.089
LeGO-LOAM	✓	0.576	0.418	0.363	0.073±0.178	0.097±0.222	0.110±0.051	0.126±0.336	0.107±0.279	0.313±0.193	0.446±0.075
LOAM	−	**0.431**	**0.316**	0.272	0.059±0.173	0.090±0.222	0.113±0.023	0.136±0.341	0.172±0.333	0.341±0.193	0.385±0.058

Wheelchair Method	Loop closure	Overall RMSE	Height RMSE	2D RMSE	RMSE per 1 m	RMSE per 3 m	RMSE per 10 m	RMSE per 30 m	RMSE per 100 m	RMSE per 300 m	RMSE per 1 km
LiTAMIN	−	0.980	0.958	**0.207**	0.024±0.013	0.030±0.016	**0.038±0.022**	**0.053±0.030**	0.106±0.053	0.258±0.115	0.738±0.194
LiTAMIN	✓	**0.701**	**0.661**	0.235	0.025±0.013	0.030±0.016	0.039±0.022	0.054±0.030	**0.102±0.043**	**0.215±0.106**	**0.519±0.111**
PCA method	−	**0.802**	**0.770**	**0.224**	**0.023±0.011**	**0.029±0.014**	**0.038±0.020**	**0.051±0.030**	**0.104±0.053**	**0.250±0.114**	**0.593±0.158**
PCA method	✓	**0.678**	**0.626**	0.260	0.023±0.011	0.029±0.014	**0.038±0.020**	**0.053±0.028**	**0.104±0.046**	**0.215±0.110**	**0.481±0.132**
hdl_graph_slam	−	7.318	4.380	5.863	**0.021±0.037**	**0.028±0.057**	0.052±0.102	0.113±0.232	0.756±0.265	1.429±0.276	5.819±0.456
hdl_graph_slam	✓	1.012	0.762	0.642	**0.020±0.012**	**0.026±0.016**	0.038±0.024	0.069±0.045	0.169±0.117	0.449±0.173	0.758±0.214
LeGO-LOAM	−	1.150	1.129	**0.215**	0.042±0.025	0.054±0.025	0.063±0.026	0.078±0.034	0.123±0.055	0.282±0.119	0.757±0.258
LeGO-LOAM	✓	1.237	1.210	0.254	0.042±0.025	0.054±0.025	0.063±0.026	0.078±0.034	0.116±0.061	0.380±0.176	0.863±0.271
LOAM	−	1.294	1.262	0.277	0.034±0.015	0.053±0.019	0.071±0.024	0.088±0.029	0.139±0.047	0.276±0.114	0.746±0.300

Segway Method	Loop closure	Overall RMSE	Height RMSE	2D RMSE	RMSE per 1 m	RMSE per 3 m	RMSE per 10 m	RMSE per 30 m	RMSE per 100 m	RMSE per 300 m	RMSE per 1 km
LiTAMIN	−	**0.934**	**0.327**	**0.860**	**0.029±0.180**	**0.039±0.016**	**0.050±0.285**	**0.075±0.325**	**0.114±0.267**	**0.321±0.131**	**0.494±0.100**
LiTAMIN	✓	**0.522**	**0.406**	**0.314**	**0.030±0.179**	**0.040±0.016**	**0.053±0.283**	**0.077±0.325**	**0.127±0.262**	**0.302±0.126**	**0.449±0.048**
PCA method	−	91.41	2.052	91.37	**0.031±1.040**	**0.043±1.449**	0.058±2.162	**0.084±3.293**	0.140±4.490	1.026±9.148	35.86±19.68
PCA method	✓	91.12	2.123	91.08	0.031±1.042	0.043±1.449	**0.058±2.160**	0.085±3.291	**0.138±4.488**	1.023±9.149	35.87±19.67
hdl_graph_slam	−	14.11	12.34	6.848	0.038±0.196	0.059±0.057	0.107±0.491	0.396±0.776	2.210±0.865	3.444±0.768	6.180±1.953
hdl_graph_slam	✓	2.608	2.223	1.323	0.035±0.178	0.049±0.016	0.069±0.325	0.137±0.369	0.474±0.293	1.036±0.283	1.589±0.333
LeGO-LOAM	−	50.92	28.65	33.14	0.089±0.186	0.115±0.025	0.142±0.351	0.261±0.612	0.984±2.385	3.857±10.42	18.24±18.56
LeGO-LOAM	✓	65.47	25.48	58.68	0.089±0.186	0.115±0.025	0.142±0.351	0.261±0.612	0.615±3.125	4.230±11.48	26.80±18.22
LOAM	−	**0.614**	**0.377**	**0.464**	0.048±0.191	0.080±0.019	0.119±0.309	0.152±0.353	0.226±0.318	**0.303±0.238**	**0.614±0.111**

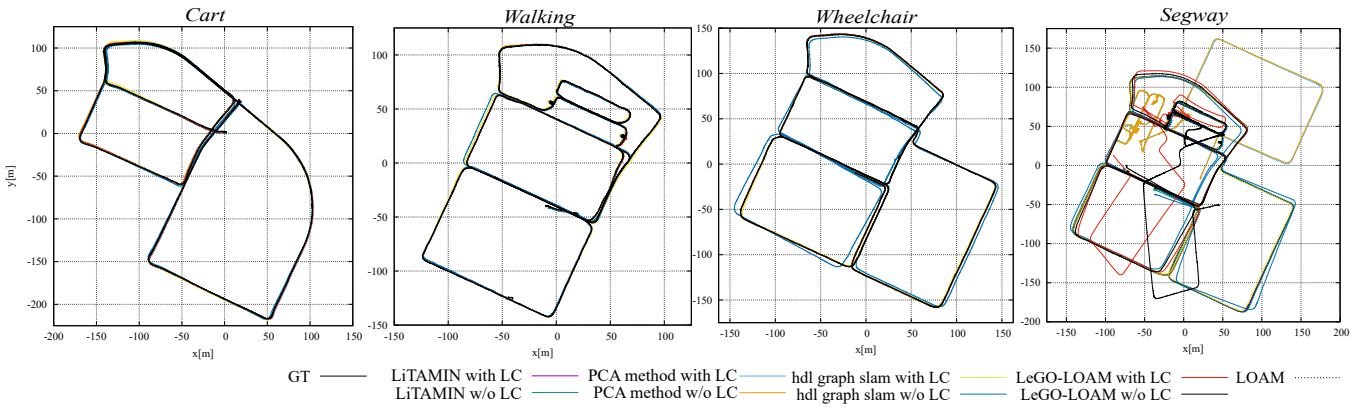

Fig. 4. Comparison of trajectories between ground truth data and each method (LC = loop closure).

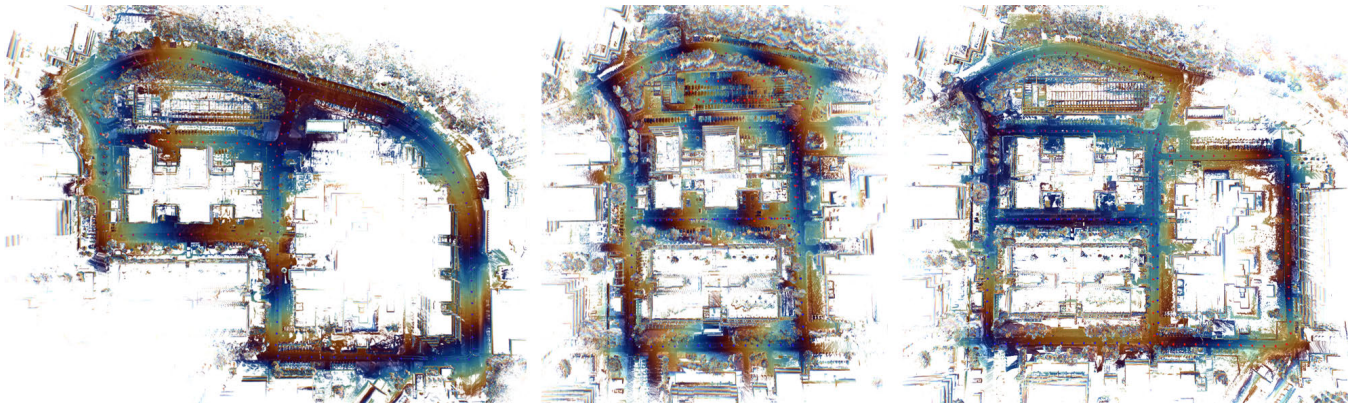

Fig. 5. Mapping results by LiTAMIN. The dataset used for each map from the left is cart, walking, and wheelchair. The Segway map is shown in Fig. 1.

TABLE III: Total computation time (sec) to build a map using all VLP-16 data frames without frame drops and thread sleep.

Method	Loop closure	Cart	Walking	Wheel-chair	Segway
LiTAMIN	✓	**634**	**584**	**572**	**481**
PCA method	✓	1,181	1,081	1,072	883
hdl_graph_slam	✓	4,800	4,353	4,306	3,567
LeGO-LOAM	✓	**302**	**468**	**328**	**315**
LOAM	–	**993**	**880**	**866**	**861**
Actual travel time		1,440	1,306	1,292	1,070

TABLE IV: Average computation time for each function of LiTAMIN.

Function	Average time (ms)	Execution unit	Implementation
Tracking	21±10	Per scan	Single thread
Local mapping	5±5	Per scan	Single thread
Accumulation	2±3	Per scan	Single thread
Storage	98±25	Per key-frame	Single thread
Graph optimization	190±176	Per key-frame insertion	Single thread
Loop detection	34±25	Per key-frame	Single thread

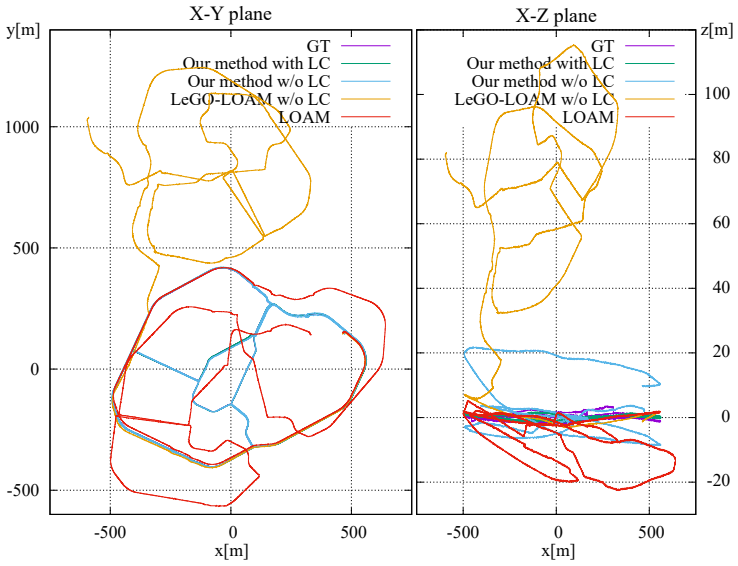

Fig. 6. Comparison of trajectories between GT data and each method for the big loop case.

Fig. 7. Mapping result by LiTAMIN for the big loop case. The background aerial photograph is referred to Google Map.

TABLE V: Accuracy evalutation for the big loop case. The mark ✓and – mean with(✓) or without(–) loop closure for each method.

big loop Method	Loop closure	Overall RMSE	Height RMSE	2D RMSE	RMSE per 1 m	RMSE per 3 m	RMSE per 10 m	RMSE per 30 m	RMSE per 100 m	RMSE per 300 m	RMSE per 1 km
LiTAMIN	–	**9.084**	**8.451**	**3.331**	**0.020±0.032**	**0.029±0.056**	**0.042±0.106**	**0.072±0.184**	**0.202±0.290**	**0.525±0.363**	**1.210±0.358**
LiTAMIN	✓	**1.347**	**1.100**	**0.769**	**0.020±0.033**	**0.026±0.049**	**0.034±0.073**	**0.049±0.102**	**0.097±0.164**	**0.215±0.182**	**0.599±0.243**
LeGO-LOAM	–	875.2	64.04	872.8	0.060±0.063	0.096±0.114	0.143±0.282	0.254±0.681	0.820±1.478	2.344±4.220	5.751±20.157
LOAM	–	**168.8**	**9.027**	**168.5**	**0.036±0.053**	**0.062±0.094**	**0.097±0.271**	**0.175±0.532**	**0.768±0.879**	**1.977±1.344**	**4.301±7.920**

build maps in half the time compared with the PCA method employing Eq. (1). We consider the computational efficiency of LiTAMIN to come from our stabilization method using Eq. (2) and local map building with 1-m voxels. Although the point correspondence computation of LiTAMIN was based on a k-d tree, we consider that the computational efficiency of LiTAMIN comes from building a k-d tree for each unit of a local map. Moreover, we consider that the voxel size of 1 m improves the accuracy and computational efficiency at the same time because this size is sufficient for accuracy and it reduces the size of the tree.

Table V shows that LiTAMIN could close the big loops while LeGO-LOAM and LOAM could not close the loops for the large error. This result is come from that odometry computation of LiTAMIN by our ICP method was accurate enough to close the big loops. The short segment errors in LeGO-LOAM and LOAM are not significantly worse results than that in LiTAMIN. This indicates that LeGO-LOAM and LOAM failed in pose tracking at some specific places. We consider the fine results by LiTAMIN were come from not only the accuracy but also the stability of our ICP method.

VIII. Conclusions

This paper describes a SLAM system with improved accuracy, robustness, and computational efficiency. The main contributions of LiTAMIN are a more stable ICP due to approximation of local geometry by normal distributions and robust loop closure detection with simple and intuitive weighting. The experimental results indicated that our method is more accurate than other state-of-the-art SLAM methods and was stable for some datasets, while other methods experiencing pose tracking failures. Moreover, our method, despite being an ICP-based system, has computational efficiency comparable to that of LeGO-LOAM, which is the fastest feature-based method. Future work will be the introduction of the tight coupling method with inertial measurement unit sensing, such as visual inertial odometry, to our ICP-based SLAM method.

References

[1] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2016.

[2] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, 2019.

[3] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic LiDAR Fusion: Dense Map-Centric Continuous-Time SLAM," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2017.

[4] J. Behley and C. Stachniss, "Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments," in *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[5] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[6] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[7] H. Ye, Y. Chen, and M. Liu, "Tightly Coupled 3D Lidar Inertial Odometry and Mapping," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2019.

[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *Proc. of International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2001.

[9] A. Segal, D. Hähnel, and S. Thrun, "Generalized-ICP," in *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[10] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[11] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The Trimmed Iterative Closest Point algorithm," in *Proc. of International Conference on Pattern Recognition (ICPR)*, 2002.

[12] S. Kaneko, T. Kondo, and A. Miyamoto, "Robust matching of 3D contours using iterative closest point algorithm improved by M-estimation," *Pattern Recognition*, 2003.

[13] T. Zinsser, J. Schmidt, and H. Niemann, "A refined ICP algorithm for robust 3-D correspondence estimation," in *Proc. of International Conference on Image Processing (ICIP)*, 2003.

[14] S. Granger and X. Pennec, "Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration," in *Proc. of European Conference on Computer Vision (ECCV)*, 2002.

[15] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse Iterative Closest Point," in *Proc. of Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, 2013.

[16] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

[17] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[18] E. Takeuchi and T. Tsubouchi, "A 3-D Scan Matching using Improved 3-D Normal Distributions Transform for Mobile Robotic Mapping," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[19] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT," *Journal of Field Robotics*, 2007.

[20] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3d registration reliability and speed - a comparison of icp and ndt," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2009.

[21] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[22] N. Sünderhauf and P. Protzel, "Switchable Constraints for Robust Pose Graph SLAM," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[23] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2013.

[24] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[25] M. C. Graham, J. P. How, and D. E. Gustafson, "Robust incremental SLAM with consistency-checking," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[26] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph SLAM," *The International Journal of Robotics Research*, 2013.

[27] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *International Journal of Robotics Research (IJRR)*, 2013.

[28] J. J. Casafranca, L. M. Paz, and P. Piniés, "A back-end L1 norm based solution for factor graph SLAM," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[29] L. Carlone and G. Calafiore, "Convex Relaxations for Pose Graph Optimization with Outliers," *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[30] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models," *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[31] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.