2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9560947

LiTAMIN2: Ultra Light LiDAR-based SLAM using Geometric Approximation applied with KL-Divergence

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

***Abstract*— In this paper, a three-dimensional light detection and ranging simultaneous localization and mapping (SLAM) method is proposed that is available for tracking and mapping with 500–1000 Hz processing. The proposed method significantly reduces the number of points used for point cloud registration using a novel ICP metric to speed up the registration process while maintaining accuracy. Point cloud registration with ICP is less accurate when the number of points is reduced because ICP basically minimizes the distance between points. To avoid this problem, symmetric KL-divergence is introduced to the ICP cost that reflects the difference between two probabilistic distributions. The cost includes not only the distance between points but also differences between distribution shapes. The experimental results on the KITTI dataset indicate that the proposed method has high computational efficiency, strongly outperforms other methods, and has similar accuracy to the state-of-the-art SLAM method.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a fundamental element of mobility technologies and services, such as autonomous mobile robots. In particular, light detection and ranging (LiDAR) and depth sensors have already been commercialized and are being applied because of their stable and accurate performance. In the near future, not only self-driving cars, but all types of mobile devices will be equipped with LiDAR or depth sensors. We anticipate a world in which point cloud data captured via SLAM will be aggregated in the cloud and shared to provide a variety of services.

There is a need to efficiently generate and update global maps from the huge amount of point cloud data aggregated in real time from devices around the world. Because the number of servers used in this process is much smaller than the number of devices, it is essential to use SLAM methods that go beyond real-time performance. The performance of current LiDAR-based SLAM is only slightly better than real-time performance.

In addition to the server, speedup is also necessary to operate SLAM on edge devices, which are severely constrained in terms of computational resources. The current LiDAR-based SLAM method is based on the premise that real-time performance is guaranteed using the CPU and GPU on a PC, and it is necessary to improve the computational efficiency of the SLAM method to ensure real-time performance on edge devices.

[1]The authors are with the Human-Centered Mobility Research Center (HCMRC), 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).

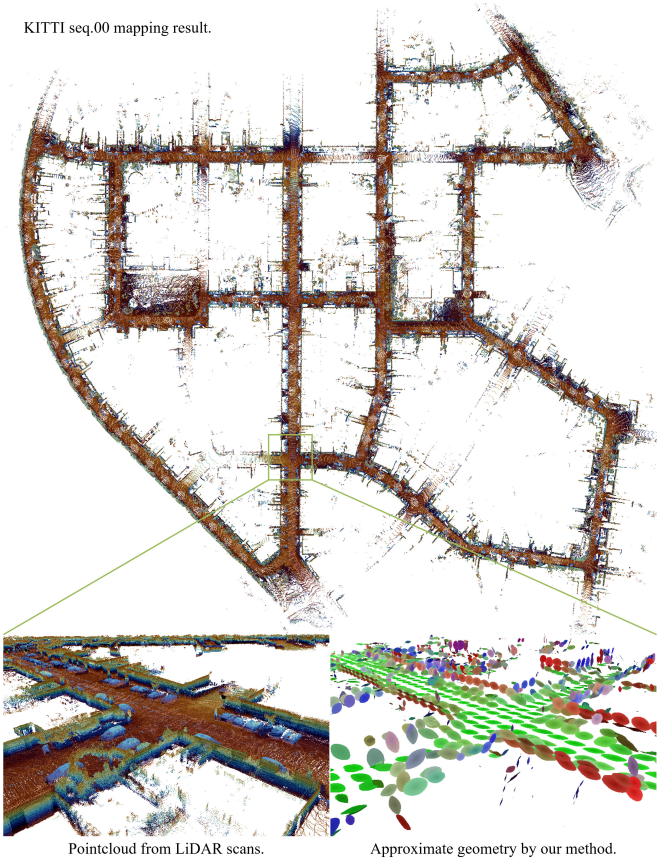

Fig. 1. Example mapping result for the KITTI sequence 00 data using LiTAMIN2. The color of the bottom right figure indicates the normal direction given from normal distributions decomposed using principal component analysis, i.e. the direction is the eigenvector of the minimum eigenvalue.

Although many studies have been conducted on SLAM benchmarks [1] and methods that emphasize accuracy [2]–[14], few studies have significantly improved the current computational efficiency. In the future, to process a large number of robots and devices intensively and efficiently, it is expected that SLAM will emphasize high speed.

The aim of this paper is to establish a method that is as accurate as the state-of-the-art method, while significantly exceeding real-time performance. In this paper, a three-dimensional (3D) LiDAR-based SLAM is discussed, which significantly improves the computational efficiency of LiDAR-based SLAMs, running at 500–1000 Hz and providing the same level of accuracy as state-of-the-art methods. The proposed method significantly reduces the number of points used for point cloud registration using a novel ICP metric while maintaining accuracy. Point cloud registration

with ICP is less accurate when the number of points is reduced. To avoid this problem, symmetric KL-divergence is introduced to the ICP cost. The experimental results (Fig. 1) on the KITTI dataset indicate that the proposed method has high computational efficiency, strongly outperforms other methods, and has similar accuracy to the state-of-the-art SLAM method.

II. Related work

LiDAR SLAM methods can be divided into two categories: ICP-based methods [2]–[8] and feature-based methods [9]–[14].

For ICP-based methods, voxelization is a simple but effective approach to speed it up. By dividing the point clouds into small groups and approximating each sub-point cloud with a normal distribution, the number of points can be significantly reduced while preserving the shape, to a certain extent. Normal distribution transformation (NDT) [15]–[17] and generalized-ICP (GICP) [18] are the most common ICP methods for performing voxel approximation, but there are some differences between them. NDT approximates only the target points with normal distributions and determines the voxel-wise correspondences, whereas GICP performs the normal distribution approximation on both the target and source point clouds and finds the correspondences using exact nearest neighbor search with a kd-tree. NDT tends to be more computationally efficient and GICP tends to be more accurate.

Feature-based methods extract geometric features, such as a line segment, plane, and point, from the input range data, and efficiently determine the correspondences. LOAM [9] was the first feature-based method to perform fast and accurate odometry calculations using LiDAR. It significantly reduces the number of points required in the localization phase using feature matching. LeGO-LOAM [11] further speeds up LOAM by relying only on good features to perform feature selection, and is one of the fastest methods currently available [2].

To achieve faster registration, some methods leverage GPU computation power, including SuMa [4], Elastic-Fusion [6], Elastic-LiDAR Fusion [5], and Droeschel et al.'s method [8]. They approximate the shape of the range data as a set of small disks called a Surfel [19]. A Surfel is a point-based rendering method [20], which is designed to render 3D shapes with a point cloud instead of a polygon mesh, and is suitable for GPU processing. It thus allows the performance of fast point-to-plane ICP [21] via the projective data association using hardware support.

Deep neural network-based approaches to LiDAR odometry [22]–[24] have also come into fashion. LO-Net [22] is an end-to-end LiDAR-odometry method. Although there is less variation in the data used for training and testing, it demonstrates accuracy similar to that of conventional approaches on a limited dataset. The main computation in LO-Net is the convolutional tensor operation, which makes it easy to parallelize point-by-point processing, and is also scalable for the future evolution of GPUs. However, whether end-to-end odometry estimation works on unlearned environments and motion has not been investigated sufficiently, and further research is needed.

Current LiDAR SLAM methods require roughly O(N) or O(Nlog(N)) for N points, and theoretically, different approaches should be introduced to improve the algorithm. The claim in the present study is simple and straightforward: The number of points required for ICP-based methods should be small. Generally, the accuracy of ICP-based methods degrades as the number of points is reduced; hence, an approach should be found to solve the trade-off problem between speed and accuracy.

III. Method

In this section, we describe the differences between the proposed method and the conventional method, LiTAMIN [2]; the differences are the method used to reduce the number of points and the cost function used for the ICP.

A. Reduction of the number of points

As shown in Figure 2, LiTAMIN voted a group of input points into the voxel grids, aligned them using the means of the voting points, and integrated the point clouds into the voxel map. The proposed method performs SLAM in a similar manner, but the difference is that it uses covariance, not just the mean, for the voting results of the input point groups. Whereas LiTAMIN was a point-to-normal distribution mapping, the proposed method extends it to normal distribution-to-normal distribution mapping. This is intended to improve accuracy by considering the spread of the distributions. The proposed method increases each voxel size and reduces the number of points, which significantly reduces the computational cost. Moreover, it avoids the loss of accuracy by considering the shape of the distribution instead of the points.

B. ICP cost function applied with symmetric KL-divergence

Table I shows the cost functions of ICP for existing methods and the proposed method. The difference between the proposed method and other methods is that the cost takes into account not only the distance between the points but also the shapes of the distributions. Although other methods, such as NDT [15], GICP [18], and LiTAMIN [2], that take covariance into account have been proposed, in practice, they only evaluate the distance by weighting the inverse of the covariance. The proposed method simultaneously evaluates the weighted distance in the first term and the difference in distribution shape in the second term. For example, if the distances between points are small but the shape of the distribution does not match, the cost is designed to be large. This cost was derived from the KL-divergence $D_{KL}(p\|q)$ of two Gaussian distributions p and q [25], [26]:

$$D_{KL}(p\|q) = \int p(x)\log\frac{p(x)}{q(x)}dx \propto (\mu_q-\mu_p)^T C_q^{-1}(\mu_q-\mu_p) + \mathrm{Tr}(C_q^{-1}C_p) - d + \log\frac{|C_q|}{|C_p|}. \quad (1)$$

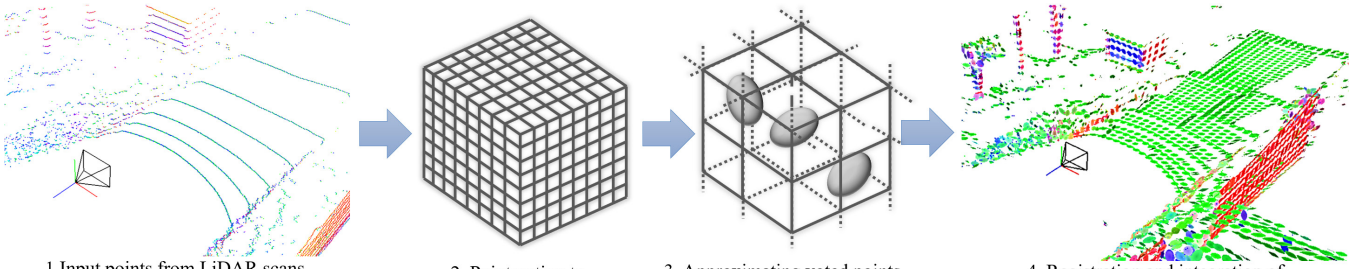

Fig. 2. Overview of our method. The color of the figure on the right shows the normal direction given from normal distributions decomposed using principal component analysis; the color is similar to that in Figure 1.

TABLE I: Comparison of the ICP cost functions for local approximation with a cluster of normal distributions.

Method	ICP cost function per point-assosiation
Standard ICP	$(q-(Rp+t))^T(q-(Rp+t))$
NDT	$(q-(Rp+t))^T C^{-1}(q-(Rp+t))$
Generalized ICP	$(q-(Rp+t))^T(C_q+RC_pR^T)^{-1}(q-(Rp+t))$
LiTAMIN	$(q-(Rp+t))^T \frac{w(C+\lambda I)^{-1}}{\|(C+\lambda I)^{-1}\|_F}(q-(Rp+t))$
LiTAMIN2 (proposed method)	$w_{ICP}\left[(q-(Rp+t))^T \frac{(C_q+RC_pR^T+\lambda I)^{-1}}{\|(C_q+RC_pR^T+\lambda I)^{-1}\|_F}(q-(Rp+t))\right]+w_{Cov}\left[\mathrm{Tr}(RC_p^{-1}R^TC_q)+\mathrm{Tr}(C_q^{-1}RC_pR^T)-6\right]$

where μ_p and μ_q are means, C_p and C_q are covariance matrices, $\mathrm{Tr}(\cdot)$ is the matrix trace, and d is the dimension of x. KL-divergence is a measure of the difference between distributions, which represents not only the difference between the mean values but also the difference between the shapes of the distributions. The aim of KL-divergence is to make a robust registration that takes into account the shape of the distribution.

KL-divergence is, however, not symmetric with $D_{KL}(p\|q) \neq D_{KL}(q\|p)$: it is generally considered not to be a distance. Because ICP is a distance-minimizing algorithm and requires a more appropriate metric, $D_{SymKL}(p\|q)$ is used, which introduces the following symmetry in this study:

$$D_{SymKL}(p\|q) = (\mu_q-\mu_p)^T(C_q+C_p)^{-1}(\mu_q-\mu_p)+\mathrm{Tr}(C_q^{-1}C_p)+\mathrm{Tr}(C_p^{-1}C_q)-2d. \quad (2)$$

Our cost function is derived by applying Frobenius normalization to $D_{SymKL}(p\|q)$ and introducing a rigid body transformation R and t. To further address the outliers, the ICP error E_{ICP} and distribution shape error E_{Cov} are set as follows:

$$E_{ICP} = (q-(Rp+t))^T C_{qp}(q-(Rp+t)) \quad (3)$$
$$E_{Cov} = (\mathrm{Tr}(RC_p^{-1}R^TC_q)+\mathrm{Tr}(C_q^{-1}RC_pR^T)-6)^2, \quad (4)$$

where

$$C_{qp} = \frac{(C_q+RC_pR^T+\lambda I)^{-1}}{\|(C_q+RC_pR^T+\lambda I)^{-1}\|_F}.$$

Additionally, the weights are as follows:

$$w_{ICP} = 1-\frac{E_{ICP}}{E_{ICP}+\sigma_{ICP}^2}, \quad (5)$$
$$w_{Cov} = 1-\frac{E_{Cov}}{E_{Cov}+\sigma_{Cov}^2}. \quad (6)$$

Note that $\|\cdot\|_F$ is the Frobenius norm, R and t are estimates of the rigid body transformation, and σ_{ICP} and σ_{Cov} are acceptable error values. w_{ICP} and w_{Cov} approach 1 if the error is less than or equal to an acceptable value, and 0 if it is greater than or equal to an acceptable value.

In Table I, the first term of the proposed method can be regarded as the ICP cost considering the covariance of the two distributions at LiTAMIN. The major difference from LiTAMIN is the second term that represents the difference in distribution shape. In this study, this term is introduced so that accuracy does not decrease, even if the number of points is greatly reduced because of the large voxel size. It is possible to calculate only the first term, that is, ICP cost. Hence, the difference between the case of the first term alone, and the case of the first and second terms combined in experiments is investigated.

C. Implementation and parameters

In this study, the Newtonian method was used to optimize the cost function of the proposed method. Because the second term of the cost function is not a squared error, it needs to be found up to the Hessian, and Newtonian optimization was used rather than the Gaussian–Newtonian method. The damping factor of the Levenberg–Marquardt method [27]

was not used in this study because the calculations were stable without it.

The acceptable values of σ_{ICP} and σ_{Cov} were empirically set to 0.5 and 3, respectively. $sigma_{ICP}$ corresponds to the Mahalanobis distance to C_{qp}, which means that correspondence points below 0.5 are trusted. σ_{Cov} corresponds to E_{Cov}; if C_q and RC_pR^T are identical, $E_{Cov} = \mathrm{Tr}(I) + \mathrm{Tr}(I) - 6$ should be 0. In $D_{SymKL}(p\|q)$, $-2d$ is a term to make the minimum value 0 of D_{SymKL}. $\sigma_{Cov} = 3$ was set to allow for about half the error that would be allowed in the absence of this term. The parameter *lambda* of Frobenius normalization was set to 10^{-6}, as in LiTAMIN.

LiTAMIN used the occupancy probability [28] for weight w, whereas the proposed method used w_{ICP} and w_{Cov} instead. For loop closure, the proposed ICP cost was used; the same parameters and implementations as in LiTAMIN were used for the other elements. The corresponding points for ICP were searched using a kd-tree in addition to LiTAMIN. Regarding the map representation, voxel maps were used in addition to LiTAMIN. Voxel maps were used to reduce the number of normal distributions that make up the map.

LiTAMIN implemented tracking and mapping in separate threads, but the proposed method combined them in a single thread. This is because it had sufficient computational efficiency, and to reduce the overhead of communication between threads. Loop closure and the graph optimizer were implemented similarly to LiTAMIN.

IV. Experiments

A. Comparison

Several state-of-the-art methods were selected as the competitors that used different speeding up algorithms, specifically, LiTAMIN, SuMa, LeGO-LOAM, LOAM, hdl-graph-slam [3], LO-Net and DeepLO. A detailed evaluation is provided on the SuMa project page, thus this was referred to in the study, whereas the computation time was acquired by the researchers running the open source. The open sources of LeGO-LOAM, LOAM, and hdl-graph-slam were also used to obtain the trajectories and to measure the computation speeds in the following experiments. Regarding LO-Net, the results in the original paper were used.

Each experiment was conducted using a desktop PC with an Intel Core i9-9900K with 32 GB RAM and an NVIDIA GeForce RTX 2080 Ti.

B. Evaluation benchmark and criteria

The KITTI Vision Benchmark was used in the experiments, which contains point clouds captured by an onboard Velodyne HDL-64E S2 in several environments. It thus allows the evaluation of the trajectories obtained by any SLAM methods. The provided point clouds were already deskewed, thus they were fed directly into the proposed method and the competitors.

The performance of each method was evaluated based on the following three criteria:

1) **KITTI stats:** The KITTI Vision Benchmark [1] statistics, that is, KITTI stats, were used in the accuracy evaluation. These criteria enable the evaluation of the quality of the estimated trajectory using the relative relations against the ground truth. In this study, the translation and rotation errors were calculated in that manner for different lengths, specifically, every 100 m up to 800 m, and the average of the errors was computed. Code provided by the benchmark was used to calculate the KITTI stats.
2) **Absolute Trajectory Error (ATE):** To evaluate the loop closing performance of each method, the ATE [29] was also calculated. ATE is an indicator of the absolute position and attitude error against the ground truth. The KITTI stats are calculated as an average of errors in sub-trajectories, which may underrate the effect of loop closing; however, the ATE allows a comparison of the entire shape of trajectories modified by loop closing based on the absolute error evaluation.
3) **Total time and frame rate:** As an index of the computational efficiency, the total time taken to process all sequences of the KITTI Vision Benchmark including the loop closing was calculated. The frame rate of the odometry was also presented to evaluate the speed of the position estimation, which may be important for some real-time applications.

C. Ablation study

The proposed method approximates the sub-point cloud voted in each voxel to a normal distribution. Because the voxel size significantly affects performance, the proposed method was thoroughly evaluated for different voxel sizes, as shown in Table II. Additionally, Table II shows the average reduction percentage from the original scan points using voxelization.

From the KITTI stats, it is not always the case that the finer the voxel, the better the accuracy. Note that the voxel size was fixed at 3 meters in the following experiments because the best performance was achieved with this value.

D. Comparison study

Table III shows a comparison of KITTI stats. For SuMa, the trajectory datasets, frame-to-frame, frame-to-model, and frame-to-model with loop closure, obtained from the authors' project page were compared. For LeGO-LOAM and hdl_graph_slam, loop closure was implemented, but because loop detection did not occur in our experiments, the results were not listed in Table III. For LOAM, the results of measurements using open sources in their paper and the results of the original paper were considered. The statistics of the individual sequences were listed in the original paper, but the final resultant KITTI stats were not listed in Table III because they were not listed in the original paper. LO-Net was not listed in Table III because statistics for individual sequences were provided in [22], but the average of all final error values was not provided.

Table IV represents a comparison of ATEs. The results for SuMa were evaluated for trajectories taken from the authors' project page, in addition to Table II. For LOAM,

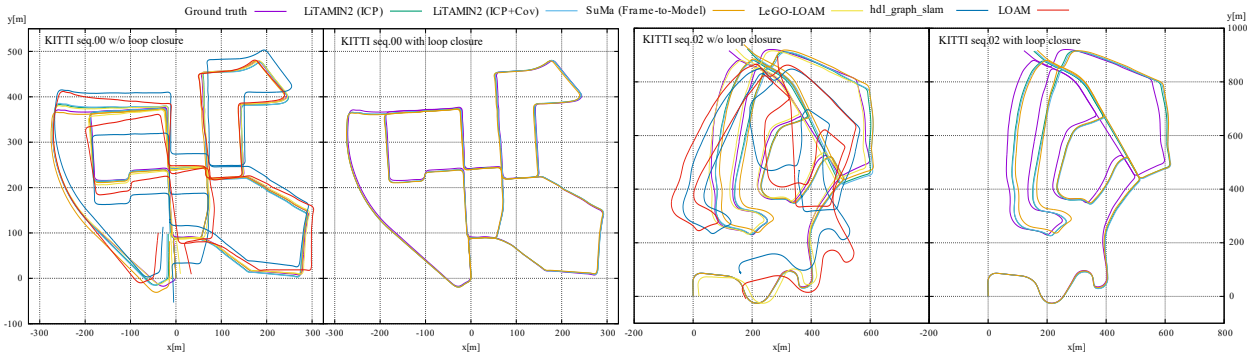

Fig. 3. Comparison of trajectories between GT data and each method.

the results for the open source software are shown. LO-Net was excluded in Table IV because the results of ATE were not included in the original paper.

Table V shows the processing time taken to create the map using each sequence of KITTI and the average frame rate of the odometry process. The proposed method, LiTAMIN, and SuMa show the processing time including loop closing because loop closing in these methods was successful for all sequences. The SuMa results were obtained on the computer used in this study using open source because the computer specification in SuMa's original paper was different.

Figure 3 shows the comparison of trajectories for each method. The second and forth figures from the left show the loop closure results for the proposed method and SuMa, but not other methods that could not detect loops.

V. Discussion

From the results in Table II, the proposed method was most accurate when the voxel size was 3 m and ICP+Cov was used as the cost function. The frame rate of odometry at that time was 510 fps for the ICP cost alone and 239 fps for the ICP+Cov cost, which was much faster than the conventional methods in Table V. This could simply be because the number of points used for registration was greatly reduced by voxel voting. Moreover, the proposed method was as accurate as the most accurate method, SuMa, as can be seen from Table III, despite the significant decrease in the number of points. We believe that this is the result of the original design intention, and the result of taking the shape of the distribution into account using symmetric KL-divergence.

According to the odometry frame rates in Table II, a comparison of the case with ICP costs alone and the case with ICP+Cov costs confirms that the accuracy of the rotation using Cov costs was slightly higher than the results using ICP alone. The reason could be that the shape of the normal distribution was more rotationally constrained. $\mathrm{Tr}(C_q^{-1}C_p)$ in Table I is a cost whose value decreased as the main axes of the normal distribution coincided, and we believe that this is also the result of our intention. However, as shown in the results of the odometry frame rates, the processing time was about twice as long in terms of calculation cost, so the ICP cost alone is sufficient if accuracy is not important. The choice of which cost to use depends on the application.

If the same accuracy as that of LiTAMIN by the conventional method is sufficient, then Table II confirms that the proposed method could achieve the same level of accuracy as LiTAMIN, even for voxels with a roughness of about 6 m. The frame rate of the proposed method exceeded 1000 fps for the ICP cost only, which is the fastest ever achieved. The results are applicable to the scale of the KITTI Vision Benchmark environment, which is similar to driving on a roadway outdoors. In indoor and more confined environments, the size of the voxels should be chosen appropriately.

Table IV is the result of evaluating the consistency of the entire trajectory. The proposed method achieved better accuracy than SuMa after loop closing, and the effect of the loop closing correction was significant. We can confirm that the proposed method had a relatively large decrease in the error in the results before and after loop closing for SuMa. This is because the loop closing method of LiTAMIN, the predecessor of the proposed method, worked properly, and the proposed ICP cost used for the detection of the loop constraint worked properly. This can be confirmed by the large error reduction in the average of all frames of the proposed method in comparison with LiTAMIN.

VI. Conclusion

In this study, an ICP method using symmetric KL-divergence was proposed to improve the speed of LiDAR-based SLAM significantly, and it was compared with other state-of-the-art SLAM methods. The proposed method achieved a computational speed of 500 fps to 1000 fps in the odometry frame rate, with the same level of accuracy as the other methods, which confirms that the proposed method is a great step forward from conventional methods. This is because the number of points used for registration was significantly reduced by voting the point cloud from LiDAR into each voxel grid and approximating the voted sub-point cloud into a single normal distribution. Although the proposed method reduced the number of points significantly, the ICP cost of the proposed symmetric KL-divergence allowed the data to be processed without reducing accuracy. These results were based on the KITTI Vision Benchmark dataset, and we believe that we need to investigate how to determine the appropriate voxel size when the usage environment changes.

TABLE II: Performance table for LiTAMIN2 with loop closure.

Voxel size	[m]	ICP cost	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10
Total time for all seq.	[sec]		3063	566	222	121	80	58	48	41	41	39	39	43	37	38	33	37	34	37	35	35
Odometry frame rate	[FPS]	LiTMIN2	7.9	45	118	225	363	510	664	814	992	1122	1255	1364	1387	1432	1420	1355	1302	1251	1122	1074
KITTI stats: rotation	[deg/100m]	ICP	1.51	0.42	0.35	0.37	0.38	0.38	0.40	0.48	0.54	0.61	0.73	0.77	0.90	1.08	1.11	1.34	1.58	1.60	1.80	2.16
KITTI stats: translation	[%]		6.01	1.42	1.11	1.00	0.92	0.89	0.88	0.99	1.09	1.36	1.57	1.67	1.88	2.32	2.49	3.26	4.76	4.37	5.52	5.77
Total time for all seq.	[sec]		4656	1058	453	251	161	119	94	81	70	65	63	64	69	64	66	68	68	71	80	88
Odometry frame rate	[FPS]	LiTMIN2	5.3	25	60	110	173	239	308	369	428	470	490	500	498	504	468	449	400	375	327	301
KITTI stats: rotation	[deg/100m]	ICP+Cov	1.46	0.56	0.38	0.36	0.37	0.33	0.36	0.44	0.50	0.58	0.77	0.85	0.97	1.35	1.36	1.71	1.89	1.94	2.41	2.44
KITTI stats: translation	[%]		5.85	1.84	1.24	0.96	0.98	0.85	0.91	0.97	1.11	1.34	1.58	1.75	2.06	2.79	2.87	3.74	4.87	4.84	6.99	7.02
Pointcloud reduction ratio	[%]		5.67	2.35	1.37	0.89	0.64	0.50	0.41	0.34	0.29	0.25	0.23	0.20	0.18	0.16	0.15	0.14	0.13	0.12	0.11	0.10

Total time for all seq. is the total processing time against all frames of all sequences. **Odometry frame rate** is the average frame rate for odometry processing for all frames. **KITTI stats** is the evaluation value of the KITTI Vision Benchmark against each voxel size. **Point cloud reduction ratio** is the ratio from the number of points from original raw scans.

TABLE III: KITTI stats for each SLAM method.

Method (Num. of frames)	Loop closure	Seq. 00 (4541)	Seq. 01 (1101)	Seq. 02 (4661)	Seq. 03 (801)	Seq. 04 (271)	Seq. 05 (2761)	Seq. 06 (1101)	Seq. 07 (1101)	Seq. 08 (4071)	Seq. 09 (1591)	Seq. 10 (1201)	KITTI stats [deg/100m] / [%]
LiTAMIN2 (ICP+Cov)	−	0.36/0.78	0.46/2.10	0.37/0.95	0.48/0.96	0.52/1.05	0.31/0.55	0.33/0.55	0.49/0.48	0.35/1.01	0.40/0.69	0.47/0.80	0.38 / 0.88
LiTAMIN2 (ICP+Cov)	✓	0.28/0.70	0.46/2.10	0.32/0.98	0.48/0.96	0.52/1.05	0.25/0.45	0.34/0.59	0.32/0.44	0.29/0.95	0.40/0.69	0.47/0.80	0.33 / 0.85
LiTAMIN2 (ICP)	−	0.42/0.75	0.40/1.88	0.45/0.99	0.43/0.84	0.41/0.90	0.32/0.74	0.23/0.45	0.57/0.55	0.55/1.25	0.32/0.74	0.59/1.36	0.45 / 0.95
LiTAMIN2 (ICP)	✓	0.33/0.70	0.40/1.88	0.37/0.92	0.43/0.84	0.41/0.90	0.28/0.50	0.31/0.50	0.34/0.43	0.48/1.16	0.27/0.81	0.59/1.36	0.38 / 0.89
LiTAMIN	−	0.46/0.91	0.45/11.3	0.46/1.30	0.56/1.17	0.47/18.7	0.39/0.75	0.29/0.59	0.34/0.48	0.42/1.04	0.45/0.99	0.90/3.78	0.46 / 1.60
LiTAMIN	✓	0.41/0.95	0.45/11.3	0.45/1.25	0.56/1.17	0.47/18.7	0.35/0.70	0.32/0.63	0.33/0.45	0.37/1.03	0.43/1.06	0.90/3.78	0.43 / 1.59
SuMa (Frame-to-Frame)	−	0.92/2.11	1.21/4.31	0.78/2.32	0.73/1.63	1.05/11.9	0.76/1.46	0.57/0.89	1.09/1.87	0.95/2.56	0.76/1.99	0.94/2.15	0.88 / 2.19
SuMa (Frame-to-Model)	−	0.30/0.72	0.47/1.77	0.39/1.06	0.46/0.57	0.27/0.39	0.23/0.50	0.14/0.39	0.33/0.37	0.35/1.01	0.25/0.47	0.27/0.69	0.33 / 0.84
SuMa (Frame-to-Model)	✓	0.22/0.64	0.47/1.77	0.41/1.23	0.46/0.57	0.27/0.39	0.20/0.42	0.28/0.51	0.52/0.65	0.35/1.15	0.20/0.57	0.27/0.69	0.32 / 0.89
LeGO-LOAM	−	1.05/2.17	1.02/13.4	1.01/2.17	1.18/2.34	1.01/1.27	0.74/1.28	0.63/1.06	0.81/1.12	0.94/1.99	0.98/1.97	0.92/2.21	1.00 / 2.49
hdl_graph_slam	−	1.00/3.92	7.62/93.5	1.84/11.2	0.92/1.71	1.21/96.0	0.69/1.41	0.94/11.1	1.10/1.28	0.99/2.17	0.93/4.32	0.75/2.36	1.49 / 9.57
LOAM	−	0.91/1.92	0.71/2.69	1.49/4.05	0.63/1.38	0.66/1.21	0.59/1.17	0.36/0.82	0.62/0.91	0.62/1.43	0.57/1.21	0.61/1.53	0.90 / 2.13
LOAM (from [10])	−	- /0.78	- /1.43	- /0.92	- /0.86	- /0.71	- /0.57	- /0.65	- /0.63	- /1.12	- /0.77	- /0.79	- / -
LO-Net (Frame-to-Frame)	−	0.72/1.47	0.47/1.36	0.71/1.52	0.66/1.03	0.65/0.51	0.69/1.04	0.50/0.71	0.89/1.70	0.77/2.12	0.58/1.37	0.93/1.80	- / -
LO-Net (Frame-to-Model)	−	0.42/0.78	0.40/1.42	0.45/1.01	0.59/0.73	0.54/0.56	0.35/0.62	0.33/0.55	0.45/0.56	0.43/1.08	0.38/0.77	0.41/0.92	- / -
DeepLO (from [23])	−	- / -	- / -	- / -	- / -	- / -	- / -	- / -	- / -	- / -	1.95/4.87	1.83/5.02	- / -

For LiTAMIN2, the size of the voxel was 3 m from the best accuracy result in Table II. The marks ✓and − represent with (✓) and without (−) loop closure, respectively, for each method.

TABLE IV: Absolute trajectory error for each SLAM method.

Method (Num. of frames)	Loop closure	Seq. 00 (4541)	Seq. 01 (1101)	Seq. 02 (4661)	Seq. 03 (801)	Seq. 04 (271)	Seq. 05 (2761)	Seq. 06 (1101)	Seq. 07 (1101)	Seq. 08 (4071)	Seq. 09 (1591)	Seq. 10 (1201)	Avg. of all frames [deg] / [m]
LiTAMIN2 (ICP+Cov)	−	1.6/5.8	3.5/15.9	2.7/10.7	2.6/0.8	2.3/0.7	1.1/2.4	1.1/0.9	1.0/0.6	1.3/2.5	1.7/2.1	1.2/1.0	1.8 / 5.1
LiTAMIN2 (ICP+Cov)	✓	0.8/1.3	3.5/15.9	1.3/3.2	2.6/0.8	2.3/0.7	0.7/0.6	0.8/0.8	0.6/0.5	0.9/2.1	1.7/2.1	1.2/1.0	1.2 / 2.4
LiTAMIN2 (ICP)	−	1.8/5.4	3.1/13.8	3.4/12.1	2.4/0.7	1.9/0.6	1.4/3.6	0.7/0.7	1.3/0.9	3.0/5.9	1.9/2.8	1.5/1.8	2.3 / 6.0
LiTAMIN2 (ICP)	✓	0.8/1.2	3.1/13.8	1.3/3.0	2.4/0.7	1.9/0.6	0.7/0.7	0.8/0.6	0.6/0.4	2.2/4.5	0.8/1.3	1.5/1.8	1.3 / 2.6
LiTAMIN	−	2.0/4.7	3.0/84.3	2.4/9.7	3.4/0.8	1.4/21.3	1.4/2.3	0.7/0.9	0.7/0.5	1.9/3.5	1.4/1.6	1.7/1.7	1.9 / 8.3
LiTAMIN	✓	1.1/1.5	3.0/84.3	1.8/3.7	3.4/0.8	1.4/21.3	0.9/1.0	0.8/0.8	0.6/0.3	1.6/2.8	1.3/1.4	1.7/1.7	1.5 / 6.2
SuMa (Frame-to-Frame)	−	6.4/19.7	8.2/34.9	5.4/21.3	4.1/1.2	3.4/13.4	2.9/5.1	1.5/2.0	2.1/2.9	6.2/15.9	2.4/5.0	2.4/3.4	4.8 / 14.1
SuMa (Frame-to-Model)	−	1.0/2.9	3.2/13.8	2.2/8.4	1.5/0.9	1.8/0.4	0.7/1.2	0.4/0.4	0.7/0.5	1.5/2.8	1.1/2.9	0.8/1.3	1.4 / 3.9
SuMa (Frame-to-Model)	✓	0.7/1.0	3.2/13.8	1.7/7.1	1.5/0.9	1.8/0.4	0.5/0.6	0.7/0.6	1.1/1.0	1.2/3.4	0.8/1.1	0.8/1.3	1.1 / 3.2
LeGO-LOAM	−	2.8/6.3	3.8/119.4	4.1/14.7	4.1/0.9	3.3/0.8	1.9/2.8	1.4/0.8	1.5/0.7	2.5/3.5	2.2/2.1	1.9/1.8	2.8 / 11.1
hdl_graph_slam	−	5.4/41.8	34.0/635.8	22.3/153.0	2.3/1.0	3.4/93.4	2.5/5.7	3.3/43.0	2.2/1.6	6.2/13.8	4.6/15.9	1.8/3.5	9.3 / 76.7
LOAM	−	5.8/19.4	6.1/21.0	21.7/111.6	3.3/1.0	2.2/0.5	2.2/4.6	0.9/1.1	1.2/1.3	3.0/6.7	1.9/5.3	1.5/1.9	7.0 / 29.7

For LiTAMIN2, the size of the voxel was 3 m from the best accuracy result in Table II. The marks ✓and − represent with (✓) and without (−) loop closure, respectively, for each method.

TABLE V: Computation time for building a map, and the odometry frame rate.

Method (Num. of frames)	Loop closure	Seq. 00 (4541)	Seq. 01 (1101)	Seq. 02 (4661)	Seq. 03 (801)	Seq. 04 (271)	Seq. 05 (2761)	Seq. 06 (1101)	Seq. 07 (1101)	Seq. 08 (4071)	Seq. 09 (1591)	Seq. 10 (1201)	Total time / Avg. rate [sec] / [FPS]
LiTAMIN2 (ICP)	✓	8.4/597	6.1/189	13.3/545	2.1/434	1.2/282	4.9/610	3.4/370	2.2/625	9.9/431	4.2/432	2.0/599	58 / 508.9
LiTAMIN2 (ICP+Cov)	✓	15.9/299	15.9/70.1	24.2/243	4.4/193	2.6/109	9.4/305	7.3/159	3.8/323	21.7/191	9.4/181	4.4/294	119 / 238.8
LiTAMIN	✓	87.9/53.0	70.4/15.8	108.1/44.0	18.9/43.4	10.1/27.3	53.3/53.0	32.7/34.4	20.2/56.2	97.4/42.9	43.3/37.5	23.5/52.5	566 / 45.2
SuMa (Frame-to-Model)	✓	90.1/55.1	20.2/57.1	77.6/65.7	12.5/65.6	5.4/51.9	57.2/54.6	21.4/55.4	18.7/65.7	74.9/58.0	31.4/52.5	22.4/55.2	432 / 58.4
LeGO-LOAM	−	78.4/69.9	27.5/69.4	91.8/64.5	16.5/62.8	5.4/64.9	50.1/67.7	24.7/64.9	17.5/73.4	80.1/65.9	32.7/63.2	21.1/66.8	445 / 66.1
hdl_graph_slam	−	800/5.7	197/5.6	1107/4.2	163/4.9	53.1/5.1	596/4.6	293/3.8	209/5.3	819/5.0	440/3.6	252/4.8	4929 / 4.8
LOAM	−	414/11.0	83.2/13.5	448/10.4	75.2/10.4	24.6/10.2	266/10.3	102/10.6	97.6/11.1	383/10.6	151/10.4	114/10.4	2156 / 10.7

For LiTAMIN2, the size of the voxel was 3 m from the best accuracy result in Table II. The marks ✓and − represent with (✓) and without (−) loop closure, respectively, for each method.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN: LiDAR Based Tracking and MappINg by Stabilized ICP for Geometry Approximation with Normal Distributions," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[3] 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.

[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] 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.

[6] T. Whelan, S. Leutenegger, R. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM Without A Pose Graph," in *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[7] F. Moosmann and C. Stiller, "Velodyne SLAM," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[8] D. Droeschel and S. Behnke, "Efficient Continuous-time SLAM for 3D Lidar-based Online Mapping," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2018.

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

[10] J. Zhang and S. Singh, "Low-drift and Real-time Lidar Odometry and Mapping," *Autonomous Robots*, vol. 41, no. 2, p. 401–416, February 2017.

[11] 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.

[12] 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.

[13] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[14] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[15] 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.

[16] 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.

[17] 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.

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

[19] H. Pfister, M. Zwickery, J. van Baar, and M. Grossy, "Surfels: Surface Elements as Rendering Primitives," in *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 2000.

[20] M. Botsch and L. P. Kobbelt, "High-quality point-based rendering on modern GPUs," in *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, 2003.

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

[22] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "LO-Net: Deep Real-time Lidar Odometry," in *Proc. of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] Y. Cho, G. Kim, and A. Kim, "Unsupervised Geometry-Aware Deep LiDAR Odometry," in *Proc. of International Conference on Robotics and Automation (ICRA)*, 2020.

[24] Y. Cho, G. Kim, and A. Kim, "DeepLO: Geometry-Aware Deep LiDAR Odometry," in *arXiv preprint arXiv:1902.10562*, 2019.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[26] J. Goldberger, S. Gordon, and H. Greenspan, "An efficient image similarity measure based on approximations of KL-divergence between two gaussian mixtures," in *Proc. of International Conference on Computer Vision (ICCV)*, 2003.

[27] M. I. Lourakis and A. A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math.Software*, 2009.

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

[29] 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.