Unsupervised Geometry-Aware Deep LiDAR Odometry

Younggun Cho[1], Giseop Kim[1] and Ayoung Kim[1*]

***Abstract*— Learning-based ego-motion estimation approaches have recently drawn strong interest from researchers, mostly focusing on visual perception. A few learning-based approaches using Light Detection and Ranging (LiDAR) have been reported; however, they heavily rely on a supervised learning manner. Despite the meaningful performance of these approaches, supervised training requires ground-truth pose labels, which is the bottleneck for real-world applications. Differing from these approaches, we focus on *unsupervised learning* for LiDAR odometry (LO) without trainable labels. Achieving trainable LO in an unsupervised manner, we introduce the uncertainty-aware loss with geometric confidence, thereby allowing the reliability of the proposed pipeline. Evaluation on the KITTI, Complex Urban, and Oxford RobotCar datasets demonstrate the prominent performance of the proposed method compared to conventional model-based methods. The proposed method shows a comparable result against SuMa (in KITTI), LeGO-LOAM (in Complex Urban), and Stereo-VO (in Oxford RobotCar). The video and extra-information of the paper are described in https://sites.google.com/view/deeplo.**

I. Introduction & Background

Odometry (ego-motion) estimation is a core module in robot navigation that presents various applications to an autonomous robot [1] and 3D mapping [2, 3]. Traditionally, most odometry modules have focused on a model-based approach using cameras [4, 5, 6] and LiDARs [7, 8, 9]. Despite their superior performances and maturity, model-based methods are exposed to challenges such as vulnerability to environmental disturbance and parameter selection. Therefore, recent studies have examined learning-based methods but mostly for visual odometry in both a supervised [10, 11] and an unsupervised [12, 13] manner.

Efforts toward learning-based odometry using a *range sensor (e.g. LiDAR)* have been recently initiated. However, the major challenge is handling a dense point cloud by feeding it into a deep neural network, while several recent studies have focused on feeding the point cloud directly to the network [14, 15]. Before describing the proposed method, we briefly summarize the model-based and learning-based LiDAR odometry in the literature.

A. Model-based Odometry Estimation

For range sensors such as an RGB-D camera and LiDAR, many model-based methods [8, 17, 9] including the odometry module, have been proposed that minimize the error between two consecutive frames or a frame and a map using the Iterated Closest Point (ICP) [18, 19, 20, 2]. Recently, Behley and Stachniss proposed a surfel-based mapping method, called SuMa, for 3D laser range data [9]. In the stream of LO studies, lidar odometry and mapping (LOAM) [21] presented an impressive performance by utilizing local feature tracking and mapping. Later, Shan and Englot [22] proposed a lightweight extension of LOAM called LeGO-LOAM exploiting ground-plane.

B. Learning-based Odometry Estimation

Unlike visual sensors, there are few learning-based methods for range sensors. The range sensor data (e.g., 3D point cloud) is sparse and irregular; this fact makes it difficult to directly employ conventional modules such as 2D convolution and upconvolution due to a memory inefficiency issue. Methods that consume irregular point cloud data directly and achieve permutation invariance have been recently proposed for object recognition or the segmentation problem [14, 23]. However, direct leverage of point cloud is still challenged due to the data type and computational cost. To alleviate these issues, studies instead proposed rasterized image-based learning methods for 3D LiDAR odometry [24, 25, 26]. Early work in Nicolai et al. [24] learned from labels to estimate 2D LiDAR motion. Later, Velas et al. [25] reported 3D motion estimation by solving the classification problem. Li et al. [26] reported meaningful results by outperforming existing model-based approaches, LOAM. However, when achieving the performance, the authors leveraged pre-trained 3D object semantic labels together with the ground-truth pose labels.

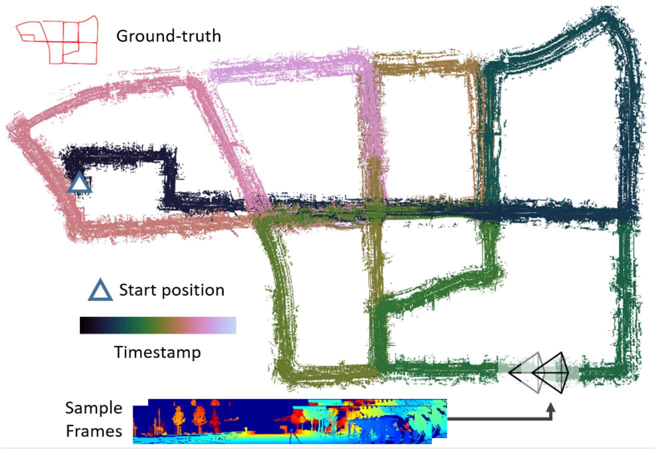


Fig. 1: A point cloud map using learned LiDAR odometry. The figure shows a sequence of the Complex Urban dataset [16]. The triangle indicates the start position, and point clouds are colored with respect to timestamps (mission time). A sample LiDAR frame is also depicted at the bottom.

[1]Y. Cho, G. Kim and A. Kim are with the Department of Civil and Environmental Engineering, KAIST, Daejeon, S. Korea `[yg.cho, paulgkim, ayoungk]@kaist.ac.kr`

This work is fully supported by [Deep Learning based Camera and LIDAR SLAM] project funded by Naver Labs Corporation.

C. Contributions

For robotics application, we believe that unsupervised learning is the key of achieving generality. Unlike the existing learning-based odometry training in a supervised manner, we introduce losses and a training strategy for unsupervised deep LiDAR odometry. For efficiency, we utilize 2D spherical projection for input representation. Also, we formulate the loss function using point-to-plane ICP with uncertainty estimation;thus, an overall training pipeline was conducted in an unsupervised manner. Fig. 1 shows the trajectory of the Complex Urban dataset [16] trained without ground-truth pose labels. This figure indicates that our method successfully captures the relative motion of a large scale trajectory (11 km) without ground-truth. To the best of our knowledge, our work is the first unsupervised learning-based odometry for a range sensor. Our contributions are as follow:

- For efficient unsupervised training and inference, we used vertexes as inputs and used them on a geometry-aware consistency loss calculation. By doing so, the time-consuming labeling procedure was alleviated in an unsupervised fashion.
- The proposed learning system can be generally used for a LiDAR point cloud (submap) regardless of the hardware type or configuration (e.g., the 3D surrounding of the KITTI dataset and the 2D push broom of the Complex Urban and Oxford RobotCar dataset).
- This paper aims the generalizability of the algorithm to various data types and environments without ground-truth. We introduce and validate a transfer learning strategy of using initial weight trained from KITTI to successfully train more challenging datasets.

II. Proposed Method

Our approach (Fig. 2) is composed of feature networks (*VertexNet*) and a pose network (*PoseNet*). *VertexNet* encodes of consecutive frames and decodes point uncertainties. *PoseNet* estimates the relative motion from encoded feature maps.

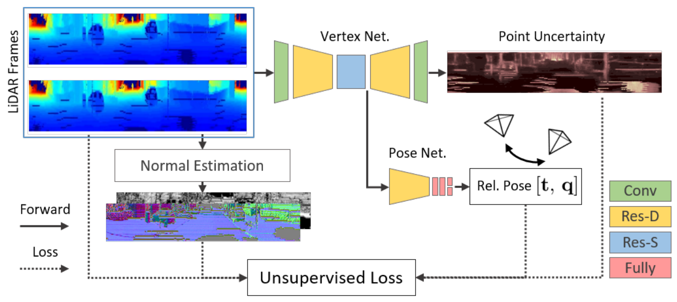


Fig. 2: The proposed network and our unsupervised training scheme. The network is composed of *VertexNet* and *PoseNet*. *VertexNet* encodes input frames and decodes point uncertainties, and features maps are forwarded into *PoseNet* to estimate the relative motion.

A. Proposed Network

As shown in Fig. 2, the proposed network is composed of 2 parts (*VertexNet* and *PoseNet*) with 4 blocks (Conv, Res-D, Res-S, and Fully).The spherically projected input frames are used as the input of the *VertexNet*. The encoded feature maps from the *VertexNet* are piped into *PoseNet*, which is designed as fully connected networks that transfer features to predict translation and rotation between the input frames. Inspired by [27], *VertexNet* consists of 4 pairs of encoding-decoding layers of convolution (Conv) and residual blocks (Res-D) with horizontal and vertical strides $(2, 1)$, while bottleneck layers have residual blocks with stride $(1, 1)$ (Res-S). *PoseNet* is composed of 4 residual blocks with stride $(2, 2)$ to encode the feature map to the feature vectors size of 1024 and fully connected layers (Fully) to estimate the relative pose $\mathbf{x}_{t,t+1} = [\mathbf{t} \in \mathbb{R}^3, \mathbf{q} \in \mathbb{R}^4]_{t,t+1}$.

B. Input Representation

1) Spherical Projection: To cope with the unordered characteristics of a LiDAR point cloud, we reformulate it using an image-coordinate-parameterized representation. Unlike a rasterized image (e.g., the range image in [25]), this representation preserves the 3D point information as real numbers. We employ projection function $\pi(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to project the 3D point cloud into a 2D image plane on spherical coordinates. Each 3D point $\mathbf{p} = (p^x, p^y, p^z)$ in a sensor frame is mapped onto the 2D image plane $\mathbf{u} = (u, v)$ represented as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} (f_h/2 - \arctan(p^y, p^x))/\delta_h \\ (f_{vu} - \arctan(p^z, d))/\delta_v \end{pmatrix}, \tag{1}$$

where depth is $d = ({p^x}^2 + {p^y}^2)^{1/2}$; f_h and f_v are the horizontal (azimuth) and vertical (elevation) field of view (FOV), respectively; and vertical FOV $f_v = f_{vu} + f_{vl}$ is composed of upper (f_{vu}) and lower (f_{vl}) parts. Here, δ_u and δ_v are the horizontal and vertical resolutions for pixel representation. If several 3D points are projected onto the same pixel coordinates, then we choose the nearest point

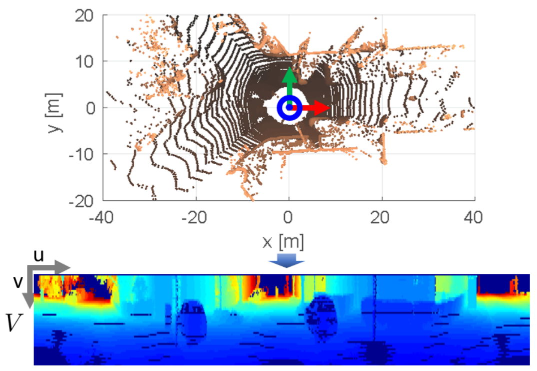


Fig. 3: A LiDAR-induced vertex (V) map is used as input for the network. The first row shows raw point clouds with the local axis in the center (RGB represents the XYZ axis). The bottom row is input vertex and color-coded with respect to the range from the origin for visualization.

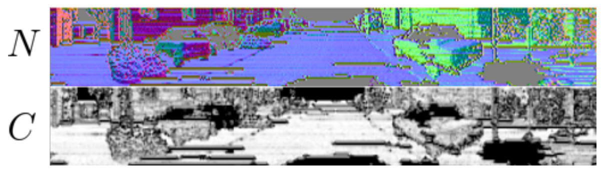


Fig. 4: Estimated normal map (top) and confidence map (bottom). For visualization, normal vectors are shifted to have values of 0 to 1. The confidence map represents smoothness of normal vectors.

as a pixel value. We define the mapped representation as vertex map V that has 2D coordinates $\mathbf{u}$ and 3-channel values $\mathbf{v} = [v^x, v^y, v^z]$ as a 3D point. Fig. 3 shows an example point cloud P_t on timestamp t, corresponding vertex map V_t.

2) Normal Vector Estimation: To enforce geometric consistency over the training process, we need to estimate a normal map N which consists of normal vectors $\mathbf{n}$ of vertices $\mathbf{v}$ in a vertex map V. Among various normal vector estimation methods, optimization-based methods that apply Singular Value Decomposition (SVD) or principal component analysis (PCA) are well-known for accurate estimation results. However, it is difficult to integrate direct optimization approaches into the learning process.

To embed the normal vector estimation into training with back-propagated gradients, we adopt averaging methods using pairs of neighbor pixels. We choose 4 neighbor conventions to compute normal vectors similar to [28, 29].

Considering point sparsity and discontinuity, we place more weight on a vertex that is geometrically closer to the center vertex by introducing weight w. Formally, the weight is expressed as $w_{1,2} = \exp\{-0.5|r(\mathbf{v_1}) - r(\mathbf{v_2})|\}$. Each normal vector $\mathbf{n}$ of a target normal map N is represented as

$$\mathbf{n}_p = \sum_{p_i \in \mathcal{P}} w_{p_0,p}(\mathbf{v}_{p_0} - \mathbf{v}_p) \times w_{p_1,p}(\mathbf{v}_{p_1} - \mathbf{v}_p), \qquad (2)$$

where $r(\,\cdot\,)$ is range value and $\mathcal{P}$ represents 4 pairs of the center vertex $\mathbf{v}_p = V(\mathbf{u}_p)$ for the center pixel p. Another key weighting factor for our loss is a confidence map that denotes the similarity of normal vectors with respect to the neighbors. The confidence map (C) is expressed as

$$C(\mathbf{u}_p) = \sum_{p_i \in \mathcal{P}} \frac{1}{4}(1 - \cos(\mathbf{n}_p, \mathbf{n}_{p_i})) \qquad (3)$$

where the confidence level represents the reliability and smoothness of surface normals. Estimated normal map (N) and corresponding confidence map C are displayed in Fig. 4. As expected, the confidence map shows higher values for flat surfaces such as walls and the ground and lower values for trees or vehicles. We filter out the inconsistent normal vectors using the confidence below the threshold ($\delta_c = 0.4$).

C. Geometry-aware Losses

1) Uncertainty-weighted ICP Loss: For unsupervised training, we integrate the geometric loss into the deep-learning framework. Given the predicted relative motion $\mathbf{x}_{t,t+1}$, we define the orthogonal distance of the point correspondences as the loss value. For the correspondence search, projective data association is utilized. Each vertex in the vertex map $\mathbf{v}_{t+1} \in V_{t+1}$ was transformed into a frame t as $\mathbf{v}'_t = T_{t,t+1}\mathbf{v}_{t+1}$, where $T_{t,t+1} \in \mathbb{R}^{4\times4}$ is a transformation matrix. Next, the corresponding vertex and normal vectors are assigned via a projection function $\pi(\,\cdot\,)$,

$$\bar{\mathbf{v}}_t = V(\pi(\mathbf{v}'_t)) \qquad (4)$$

$$\bar{\mathbf{n}}_t = N(\pi(\mathbf{v}'_t)). \qquad (5)$$

Using $\bar{\mathbf{v}}_t$ and $\bar{\mathbf{n}}_t$, corresponding vertex and normal vectors of $\mathbf{v}_{t+1}$ on frame t, we can compute the orthogonal distance of associated points. The distance function is represented as

$$d(\mathbf{v}_{t+1}) = \bar{\mathbf{n}}_t \cdot (T_{t,t+1}\mathbf{v}_{t+1} - \bar{\mathbf{v}}_t)C(\mathbf{v}_{t+1}). \qquad (6)$$

Also, predicted point uncertainty of *VertexNet* is applied for robust optimization. Following [30], we formulate the ICP loss as

$$\mathcal{L}_{icp,w} = \sum_{\mathbf{v_{t+1}} \in V_{t+1}} \frac{|d(\mathbf{v}_{t+1})|_1}{\sigma^2_{t+1}} + \log \sigma^2_{t+1}, \qquad (7)$$

where $\sigma_{t+1} = \Sigma(\mathbf{u}_{t+1})$ represents estimated uncertainty on a pixel $\mathbf{u}_{t+1} = \pi(\mathbf{v}_{t+1})$.

2) FOV Loss: We also introduce FOV loss $\mathcal{L}_{fov}$, which prevents divergence training of the out of FOV condition. During unsupervised training, the estimation tends to fall into a trivial solution by making zero covisibility between frames. This is because the ICP loss $\mathcal{L}_{icp}$ is zero when there are no correspondences, and a naïve ICP loss may lead the network to a large relative motion that yields no correspondences. To avoid such cases, we use a penalty loss as a hard-counting loss of out of FOV points. The FOV loss is expressed as

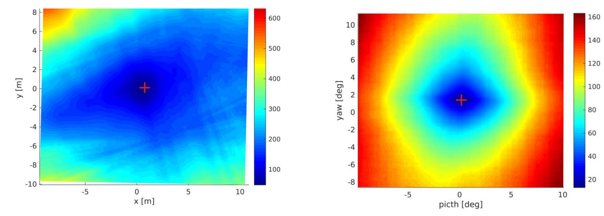
$$\mathcal{L}_{fov} = \sum_{\mathbf{v} \in V_{t+1}} \mathbb{I}(\pi(T_{t,t+1}\mathbf{v}) - (w,h)) + \mathbb{I}(-\pi(T_{t,t+1})) \qquad (8)$$

where $\mathbb{I}$ represents the Heaviside function and (w, h) are the width and height of the vertex map, respectively.

3) Overall Loss: Finally, the overall unsupervised loss is obtained as

$$\mathcal{L} = \mathcal{L}_{icp,w} + \lambda\mathcal{L}_{fov} \qquad (9)$$

where λ is a balancing factor. The characteristics of loss $\mathcal{L}$ on motion perturbation is depicted in Fig. 5. Each curve on



(a) Perturbation on translation (b) Perturbation on rotation

Fig. 5: The tendency of the Unsupervised loss ($\mathcal{L}$) (z-axis) with respect to motion perturbation (x and y-axis), plotted with the ground-truth pose (red cross). The figure represents the validity of the loss over large motion perturbation (±10 m on translation and $\pm10^\circ$ on rotation). Colors in error bars indicate the magnitude of unsupervised loss $\mathcal{L}$.

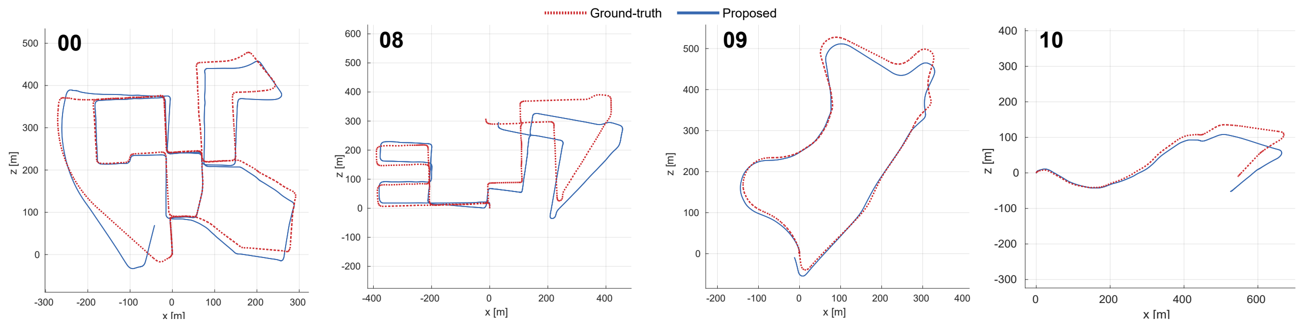


Fig. 6: KITTI trajectory comparison of the proposed method against the ground-truth trajectory. Two sample trajectories (`00` and `08`) from training sequences and two test sequence results (`09` and `10`).

translation and rotation has a convex shape around ground-truth. This indicates that the tendency of loss supports the validity of the proposed loss on training.

III. Experimental Results

In this section, we evaluate the proposed model via qualitative and quantitative comparisons using the publicly available datasets; KITTI [31], Complex Urban [16], and Oxford Robotcar [3] datasets.

A. Implementation and Training

The proposed network was implemented using PyTorch and trained with an NVIDIA GTX 1080ti. We employed the Adam solver [32] with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $w_{decay} = 10^{-5}$. We started the training with an initial learning rate of 10^{-4} and controlled it by a step scheduler with a step size of 20 and $\gamma = 0.5$. We set horizontal FOV $f_h = 360°$ and vertical FOV $f_v = 26°$ to process raw point clouds to the vertex map. The corresponding horizontal and vertical resolutions were $\delta_h = 0.5°$ and $\delta_v = 0.5°$, and the size of the input vertex map was 720×52.

B. Datasets

To compare the performance of the algorithms in various environments, we verified the results on KITTI, Complex Urban, and Oxford Robotcar dataset. As shown in Fig. 7, each dataset has a different LiDAR configuration and environmental characteristics.

KITTI Odometry Benchmark. The KITTI odometry dataset consists of 10 sequences with LiDAR point clouds from Velodyne HDL-64E recorded at 10 Hz and associated a ground-truth pose. Similar to previously reported learning-based odometry methods [13, 33], we used sequences `00-08` for the training and `09-10` for the test.

Complex Urban Dataset. Due to the high complexity, it is difficult to guarantee the performance of the existing LiDAR odometry methods. The training and test sequences both have various rotation intervals and long path lengths. Unlike the KITTI dataset, the Complex Urban dataset provides 2D LiDAR scan data of a push-bloom type. Therefore, we made a submap with sufficient length ($80\,\mathrm{m}$ in our work) of accumulated 2D scans. The `Urban 09` sequence was used for learning, and the `Urban 28` sequence with different driving routes in the same place was used for the test.

Oxford Robotcar Dataset. As in the Complex Urban dataset, this dataset uses a push-broom style 2D LiDAR and the submap was generated in a similar fashion. Since the focus of the Oxford RobotCar dataset is on seasonal diversity, we used the `Long` sequences (`2015-02-03-08-45-10` and `2015-03-10-14-18-10`) as training data and the other types as test data.

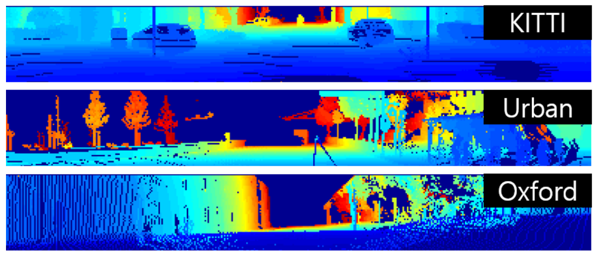


Fig. 7: Sample vertex (V) map from each evaluation dataset.

TABLE I: KITTI odometry evaluation.

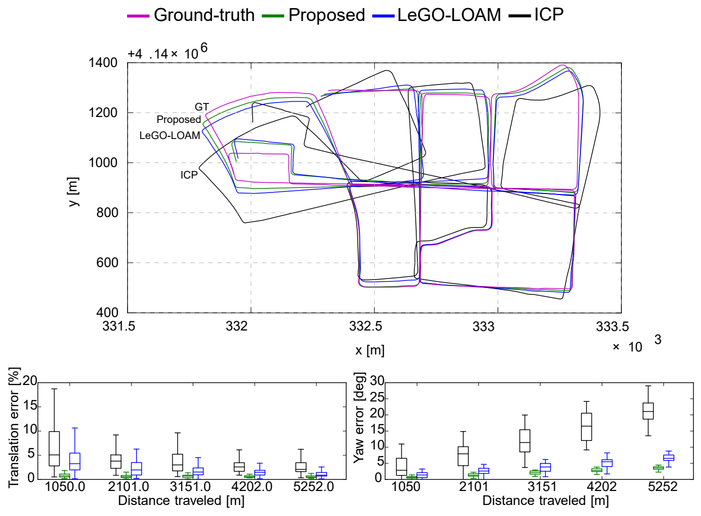
	Training (00-08)		Seq. 09		Seq. 10	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Proposed	3.68	0.87	4.87	1.95	5.02	1.83
Zhu et al.	5.72	2.35	8.84	2.92	6.65	3.89
SfMLearner	28.52	4.67	18.77	3.21	14.33	3.30
UnDeepVO	4.54	2.55	7.01	3.61	10.63	4.65
SuMa	3.06	0.89	1.90	0.80	1.80	1.00

TABLE II: Translation $t_{rel}(\%)$ and rotation $r_{rel}(°/100m)$ RMSE drift on length of $100\,\mathrm{m}$ to $800\,\mathrm{m}$ are presented. The RMSE values of the other methods were obtained from [33] and [9].

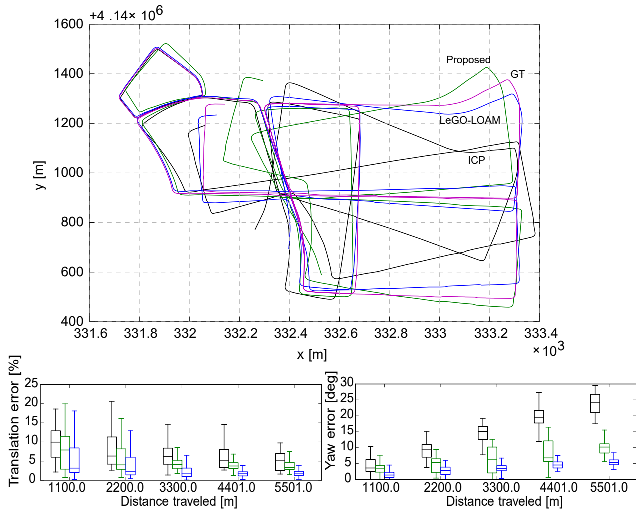
C. Evaluation on the KITTI Dataset

Fig. 6 shows the trajectories from the proposed method on the KITTI dataset. The figure includes two sequences (`00` and `08`) together with two test sequences (`09` and `10`). As in the figure, our method presents well-estimated trajectories on both training and test sequences, which have independent and different environmental characteristics.

Table I contains the details of the results; the average translation $t_{rel}(\%)$ and rotation $r_{rel}(°/100m)$ RMSE drift on length of $100\,\mathrm{m}$ to $800\,\mathrm{m}$. To verify the performance



(a) Urban 09 (Trajectory and Relative Errors)



(b) Urban 28 (Trajectory and Relative Errors)

Fig. 8: Results of the Complex Urban dataset. Estimated trajectories and relative pose errors of the proposed method, point-to-plane ICP, and LeGO-LOAM on (a) `Urban 09` (training sequence) and (b) `Urban 28` (test sequence). Box plots represent the error statistics: the median (center line), 25% and 75% quantiles (box), and minimum and maximum errors (whisker)

of unsupervised learning and training losses, we evaluated our method against several previously reported unsupervised learning methods, namely UndeepVO [13], SfMLeaner [12], and Zhu's method [33]; the resulting values for these methods were taken from the result in [33]. We also compared our method to SuMa [9] which is a recent model-based simultaneous localization and mapping (SLAM) method using LiDAR measurements. Since SuMa used similar point-to-plane projective loss, the qualitative comparison ensured the general performance of the proposed method. The values of SuMa are referenced from frame-to-frame estimation results of the paper [9]. Compared to other unsupervised learning methods, our method shows meaningful performance on both training and test sequences. Although the proposed method is not superior to SuMa in KITTI dataset, it yields a comparable performance even when trained in an unsupervised manner.

D. Evaluation with the Complex Urban Dataset

For evaluation within dynamic and realistic urban environments, we evaluated the proposed method over the Complex Urban dataset by comparing it against the point-to-plane ICP (ICP-p2l) and LeGO-LOAM [22]. By doing so, we aimed to ensure the generality of the proposed method.

It is notable that the Complex Urban dataset has more challenging environments for LO than KITTI. Thus we applied transfer learning to secure network generality and improve learning stability by using the weight of the network learned by the KITTI dataset as the initial weight. We found that this strategy significantly improved the training phase by enabling the learning for the dataset with less diversity for training.

Fig. 8 shows the learned and test paths for the sequences of `Urban 09` and `Urban 28` together with evaluation in terms of Relative Error (RE) [34]. For trajectory alignment, $SE(3)$ transformation was estimated and applied to the tested methods. Next to each trajectory is the quantitative result compared to the ground-truth in terms of relative translation error (%) and heading error (deg). When computing REs, subtrajectory segments of tested methods were selected along with various travel distances. Each subtrajectory was aligned using the first state, and the error was calculated for all the subtrajectories.

Although we utilized a similar objective function as in ICP, the proposed method resulted in a better estimation result than ICP-p2l. Model-based ICP methods iteratively search optimal solution between two frames, whereas network-based methods train the network to find optimal results for the entire data. Unlike ICP, which easily result in local minima depending on the environment, the proposed method shows more general performance improvement. As can be seen from the comparison of the relative translation and rotation of Fig. 9, the proposed method also reported better

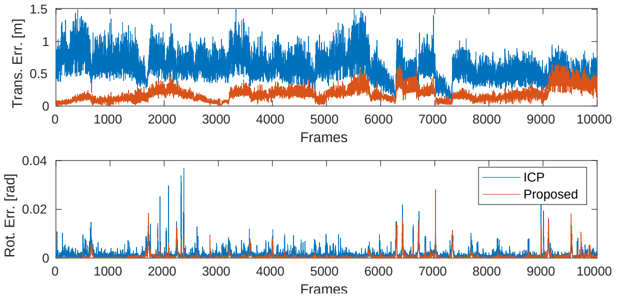
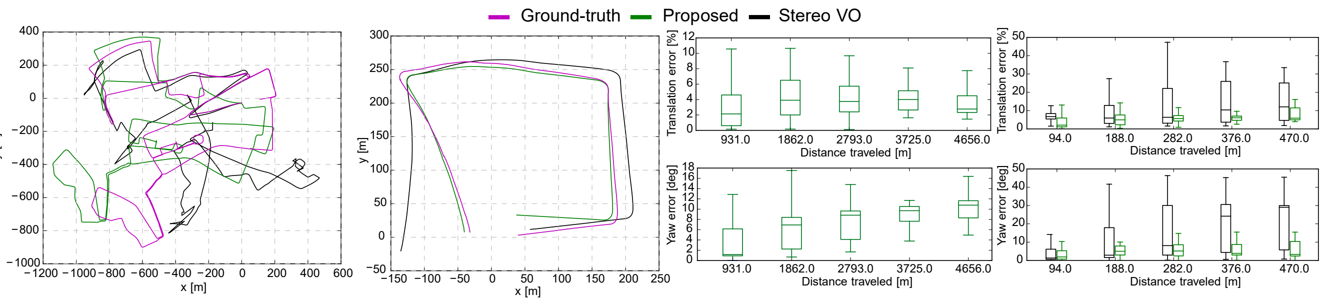


Fig. 9: Detailed RE variation plot compared against ICP. The proposed method revealed smaller RE over entire mission time.



(a) 2015-03-10-14-18-10 (Trajectory Long) (b) 2014-06-23-15-41-25 (Trajectory Alternate) (c) 2015-03-10-14-18-10 Evaluation (d) 2014-06-23-15-41-25 Evaluation

Fig. 10: Evaluation of the proposed methods via the Oxford RobotCar dataset; (a)-(b) trajectories, (c) relative translation, and (d) relative heading errors depicted over distance. Each column shows the trajectories and evaluation results with different types of sequences. (a) Training: `2014-03-10-14-18-10 Long`, (b) Testing: `2014-06-23-15-41-25 Alternate`. Also, corresponding relative errors are reported in (c) and (d). Box plots represent the same as in Fig. 8.

TABLE III: Trajectory Errors on Oxford RobotCar dataset

		Absolute Trajectory Error (RMSE)	
Datetime	Type	StereoVO [3]	Proposed
2014-05-14 -13-50-20	Alternate	37.74	19.73
2014-05-14 -13-59-05	Alternate Reverse	34.55	22.87
2014-06-23 -15-41-25	Alternate	36.09	12.92

results.

Comparing against LeGO-LOAM, the proposed unsupervised deep LiDAR odometry performed better in the training sequence and achieved comparable estimation performance in the test sequence. Excluding the mapping module from LeGO-LOAM would be the more fair comparison but could not yield a meaningful estimation thus has been excluded in the comparison. Note that however, ours produced a comparable result when compared to LeGO-LOAM with the mapping module without requiring a ground-truth pose label during the training phase.

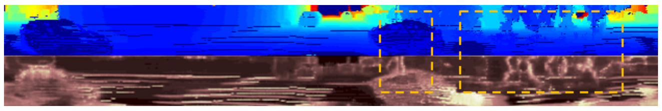
E. Evaluation with the Oxford RobotCar Dataset

We compared our method with the ground-truth trajectory and stereo visual odometry given in the Oxford RobotCar dataset. Since all trajectories have different frame rates, we evaluated each method following the trajectory evaluation method [34] as in Complex Urban dataset.

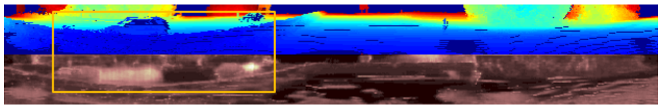
Similarly, as in Complex Urban dataset, trajectory estimation and RE plots are presented in Fig. 10. In the case of the training sequence, the stereo trajectory was not accurate enough to be used as the baseline and was thus excluded from the comparison. As can be seen, our methods showed better performances for all of the sequences with the trained networks being able to capture the relative motion of test sets. Table III lists Absolute Trajectory Error (ATE)s over test sequences. This evaluation quantifies the quality of the whole trajectory.

F. Uncertainty Visualization

The uncertainty estimation of the proposed method is an important measure to improve the learning efficiency and to measure the quality of projected points. Fig. 11 represents



(a) Depth Discontinuity



(b) Moving Objects

Fig. 11: Estimated point uncertainty. Brighter pixels represent high uncertainty. (a) Point uncertainty on depth discontinuity. Dotted boxes show high uncertainties about the points of object boundaries which have large depth differences. (b) Point uncertainty on moving objects. The solid box presents pixels with high uncertainties about moving objects.

the predicted uncertainties from the proposed network. As can be seen from the figure, uncertainty is emphasized in the part where geometric ambiguity occurs. Examples of regions with high uncertainty include the area where the distance suddenly changes, the vehicle window area where LiDAR points become noisy, and moving objects. As can be seen in Fig. 11(b), moving objects are highlighted with large uncertainty because the network learned to increase the association ambiguity during the learning process.

IV. Conclusion

In this paper, we demonstrated a learning-based LiDAR odometry estimation pipeline that is trainable in the unsupervised manner. We showed that the proposed unsupervised loss could capture the geometric consistency of point clouds. To the best of our knowledge, ours is the first unsupervised approach for deep-learning-based LiDAR odometry which is the extension of our previous approach [35]. In addition, our method showed prominent performance compared to other learning-based or model-based methods in various environments. We also derived training adaptation via transfer learning in heterogeneous environments.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2015, pp. 742–749.

[3] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Intl. J. of Robot. Research*, vol. 36, no. 1, pp. 3–15, 2017.

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

[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2014, pp. 15–22.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[7] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3D range sensor with application to mobile mapping," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1104–1119, 2012.

[8] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Proc. Robot.: Science & Sys. Conf.*, Berkeley, USA, July 2014.

[9] J. Behley and C. Stachniss, "Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments," in *Proc. Robot.: Science & Sys. Conf.*, Pittsburgh, Pennsylvania, June 2018.

[10] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2017, pp. 2043–2050.

[11] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep tracking and mapping," in *Proc. European Conf. on Comput. Vision*, 2018, pp. 851–868.

[12] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, vol. 2, no. 6, 2017, pp. 1851–1860.

[13] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVo: Monocular visual odometry through unsupervised deep learning," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2018, pp. 7286–7291.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," 2017, pp. 652–660.

[15] M. A. U. G. H. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, pp. 4470–4479, 2018.

[16] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.

[17] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *Intl. J. of Robot. Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[18] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[19] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. IEEE Intl. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2001, pp. 145–152.

[20] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot.: Science & Sys. Conf.*, Seattle, USA, June 2009.

[21] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[22] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, Oct 2018, pp. 4758–4765.

[23] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution On X-Transformed Points," in *Advances in Neural Information Processing Sys. Conf.*, 2018, pp. 828–838.

[24] A. Nicolai, R. Skeele, C. Eriksen, and G. A. Hollinger, "Deep learning for laser based odometry estimation," in *RSS workshop Limits and Potentials of Deep Learning in Robotics*, 2016.

[25] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *Proc. Intl. Conf. Aut. Rob. Sys. and Comp.*, 2018, pp. 71–77.

[26] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "Lo-net: Deep real-time lidar odometry," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, June 2019.

[27] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[28] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2009, pp. 1977–1982.

[29] Z. Jia, "Using cross-product matrices to compute the svd," *Numerical Algorithms*, vol. 42, no. 1, pp. 31–61, May 2006.

[30] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[31] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. IEEE Conf. on Comput. Vision and*

Pattern Recog., 2012.

[32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[33] A. Z. Zhu, W. Liu, Z. Wang, V. Kumar, and K. Daniilidis, “Robustness meets deep learning: An end-to-end hybrid pipeline for unsupervised learning of egomotion,” *arXiv preprint arXiv:1812.08351*, 2018.

[34] Z. Zhang and D. Scaramuzza, “A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 7244–7251.

[35] Y. Cho, G. Kim, and A. Kim, “Deeplo: Geometry-aware deep lidar odometry,” *arXiv preprint arXiv:1902.10562*, 2019.