DeepLO: Geometry-Aware Deep LiDAR Odometry

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

***Abstract*— Recently, learning-based ego-motion estimation approaches have drawn strong interest from studies mostly focusing on visual perception. These groundbreaking works focus on unsupervised learning for odometry estimation but mostly for visual sensors. Compared to images, a learning-based approach using Light Detection and Ranging (LiDAR) has been reported in a few studies where, most often, a supervised learning framework is proposed. In this paper, we propose a novel approach to geometry-aware deep LiDAR odometry trainable via both supervised and unsupervised frameworks. We incorporate the Iterated Closest Point (ICP) algorithm into a deep-learning framework and show the reliability of the proposed pipeline. We provide two loss functions that allow switching between *supervised* and *unsupervised* learning depending on the ground-truth validity in the training phase. An evaluation using the KITTI and Oxford RobotCar dataset demonstrates the prominent performance and efficiency of the proposed method when achieving pose accuracy. The overall algorithm is presented in https://youtu.be/Y2s08dv-Mq0.**

I. Introduction

Odometry (ego-motion) estimation is a core module in simultaneous localization and mapping (SLAM) which presents various applications to an autonomous robot [1] and 3D mapping [2, 3]. So far, most odometry modules have been focused on model-based using cameras [4, 5, 6] and LiDAR [7, 8, 9]. For example, visual-LiDAR odometry and mapping (V-LOAM) records the first place in the KITTI odometry benchmark and has shown remarkable accuracy. Despite their superior performances, model-based methods are exposed to challenges such as vulnerability to environmental disturbance and parameter selection. Therefore, recent studies have started to examine learning-based methods mostly for visual odometry in both a supervised [10, 11] and an unsupervised [12, 13] manner.

Similar to vision, some effort toward learning-based odometry using a *range sensor (e.g. LiDAR)* has been initiated. However, the major challenge is to handle a dense point cloud by feeding it into a deep neural network, and several recent studies have focused on feeding the point cloud directly to the network [14, 15], albeit for object-sized point cloud data. As an example for odometry, learning-based approaches for point clouds were presented in [16, 17] where the authors relied on a supervised method requiring the ground-truth with labeled sequences. Unlike these previous approaches, we examine an unsupervised manner for deep LiDAR odometry in order to achieve scalability and flexibility in the training phase.

[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 supported by the Korea MOLIT (19CTAP-C142170-02).

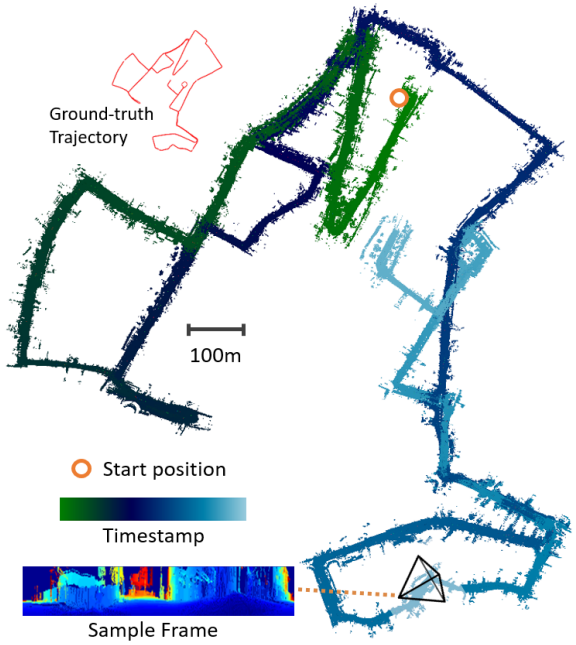


Fig. 1: A point cloud map using learned LiDAR odometry. The figure shows long sequences of the Oxford RobotCar dataset [3]. The oragne circle indicates the start position and point clouds are colored with respect to timestamps (mission time). A sample LiDAR frame is also presented.

In this paper, we propose unsupervised deep LiDAR odometry, called *DeepLO*. For efficiency, we feed a rendered vertex map and a normal map into a network and regress a 6D relative pose between two frames, while the NICP-like loss [2] is calculated using the aforementioned representations; thus, an overall training pipeline is conducted in an unsupervised manner. Fig. 1 shows the learned trajectory of Oxford Robotcat dataset [3] with an unsupervised manner. This figure indicates that our method successfully captures the relative motion of a long sequence trajectory (10 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:

- We propose a general pipeline for deep-learning-based LiDAR odometry that can be trained in both a supervised and an unsupervised manner.
- For efficient unsupervised training and inference, we use a vertex and normal map as inputs and use them on loss calculation. By doing so, the time-consuming labeling procedure is alleviated in an unsupervised fashion.

These summarized representations can be exploited in both the training and inference stages.
- The proposed learning system can generally be used for a LiDAR point cloud (submap) regardless of the hardware type or configuration (e.g., the 3D surrounding by KITTI dataset and the 2D push-broom of the Oxford RobotCar dataset).

The remaineder of the paper is composed as follows. In Section II, recent works on model and learning-based odometry estimation methods are discussed. In Section III, we introduce the architecture and loss functions of our learning-based LiDAR odometry. The performance of our method is compared with other state-of-the-art methods in Section IV. Conclusions and ideas for further work are shared in Section V.

II. Related Works

In this section, we provide a summary of existing model- and learning-based methods for odometry estimation.

A. Model-based Odometry Estimation for a Range Sensor

For range sensors such as an RGB-D camera and LiDAR, many model-based methods [8, 18, 9] including the odometry module, have been proposed which minimize the error between two consecutive frames or a frame and a map using the ICP [19, 20, 21, 2]. LOAM extracts edge and planar features for matching and run two parallel modules of different frequencies for fast and accurate ego-motion estimation [8]. Recently, Behley and Stachniss proposed a surfel-based mapping method, called SuMa, for 3D laser range data [9]. The SuMa representation is considered to be efficient and accurate for dense mapping such as ElasticFusion [18] using an RGB-D camera. This type of research [18, 9] also minimizes the error between the current frame and the rendered view of a map using ICP. By leveraging rendered-image coordinate-parameterized normal and vertex information, SuMa can almost employ points for ICP (unlike feature-based methods such as LOAM) while minimizing a loss of a piece of original information (e.g., from filtering, rasterization, or taking features).

B. Learning-based Odometry Estimation

1) Visual Sensor: Recently a number of learning-based visual odometry methods have been developed. Wang et al. [10] proposed a supervised visual odometry network using a recurrent network structure, while Zhou et al. [11] introduced a method of configuring a tracking and mapping process in a network structure. However, since it is challenging to construct a lot of ground-truth data to train a network, many unsupervised learning methods which leverage photo-consistency have recently been introduced. Zhou et al. [12] reported that the metric depth could be learned from monocular sequences by warping the consecutive images. Li et al. [13] proposed the self-supervised learning of depth and ego-motion through spatial and temporal stereo consistency. Recently, [22] proposed the joint learning of the depth, odometry, and optical flow of consecutive scenes with consideration of the outlier (dynamic) pixels for robustness. In addition, [23] introduced a hybrid pipeline for robust ego-motion estimation. This method learns disparity and depth via deep networks and predicts relative motion with model-based random sample consensus (RANSAC) outlier rejection.

2) Range Sensor: Unlike the aforementioned methods for visual sensors, there are few learning-based methods for range sensors because 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 memory inefficiency issue. Recently, some methods that consume irregular point cloud data directly and achieve permutation invariance have been proposed for object recognition or the segmentation problem [14, 24]. However, to the best of our knowledge, there have not yet been any empirical reports for odometry estimation that directly leverages point cloud. To avoid the these issues, a few studies proposed rasterized image-based learning methods for 3D LiDAR odometry [16, 17]. However, they lose the original point cloud information (e.g., the 3D point coordinates as real numbers) via the rasterization, and their training is performed in a supervised manner, thus they have low scalablility for the emergently available 3D point cloud data as LiDAR becomes more popular.

In contrast to the aforementioned methods, we propose a deep LiDAR odometry network in both an unsupervised and a supervised manners. By incorporating traditional model-based point errors into the deep architecture, we can build the unsupervised learning pipeline of LiDAR-based odometry without ground-truth relative poses.

III. Proposed Method

In this section, we describe the details of our approach. Our system is composed of feature networks (*FeatNet*) and a pose network (*PoseNet*). FeatNets extract the feature vectors of consecutive frames and PoseNet estimates the relative motion of the frames from features. The networks can be trained in both a supervised and an unsupervised manner. A proper training strategy can be chosen according to the availability of ground-truth labels. The overall pipeline for training and inference is depicted in Fig. 4.

A. Input Representation

Before describing the details, we first explain the input representation given a LiDAR point cloud. To cope with the unordered characteristics of a LiDAR point cloud, we reformulate it using an image coordinate-parameterized representation which unlike rasterized image (e.g., range image in [17]), 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 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 (u, v) represented as

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

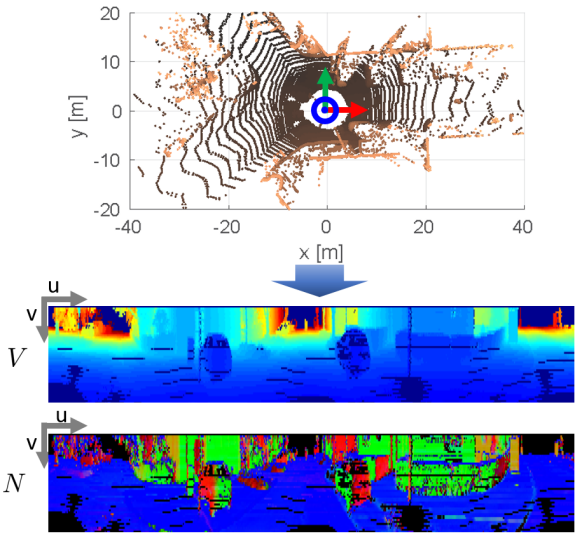


Fig. 2: LiDAR-induced vertex (V) and normal (N) maps were used as input for the network. The first row shows raw point clouds with the local axis in the center (RGB represents the XYZ axis). The bottom rows are input vertex and normal maps. The vertex map is color-coded with respect to the range from the origin for visualization.

where depth $d = (p^{x2}+p^{y2})^{1/2}$; f_h and f_v are the horizontal (azimuth) and vertical (elevation) field-of-view, respectively; vertical field-of-view $f_v = f_{vu} + f_{vl}$ 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, we choose the nearest point as a pixel value. We define the mapped representation as vertex map V which has 2D coordinates $(u, v) \in \mathbb{R}^2$ and 3-channel values $\mathbf{v} = [v^x, v^y, v^z]$ as a 3D point. Fig. 2 shows an example point cloud P_t on timestamp t, corresponding vertex map V, and normal map N on frame $F_t = [V_t, N_t]$.

We then assign a normal vector $\mathbf{n}$ of each vertex $\mathbf{v}$ adopting normal estimation methods in [2]. The normal vector of each vertex $\mathbf{v}$ is computed by the nearest vertices in the vertex map. Because we already built a vertex map, extensive queries on a kd-tree are not required. For reliable normal vector estimation, we discard the distance vertices from the center vertex. In this paper, we set the threshold range as $50\,\mathrm{cm}$ and filtered simply by the depth values computed on vertex map generation.

To verify the frame representation F, we compare the proposed representation with the existing range-based representation which utilizes a point range and extra characteristics (e.g., intensity and height) as pixel values. Fig. 3 shows the comparisons of reconstructed point clouds of each descriptions: vertex map (blue) and range map (orange) representation. The left plot is a sample frame of an urban scene (left), and a right plot represents a top view of a wall (plane). As in the enlarged view of wall, the range map-based representation [17, 16] has offset of the reconstructed point clouds due to the angular discretization of the range image.

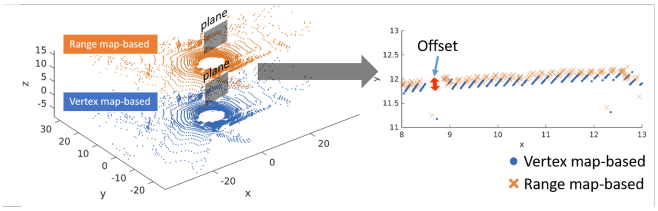


Fig. 3: A comparison between two point cloud representations. The reconstructed point cloud from the proposed method (blue) and the range map-based method (orange). As shown in the enlarged plot over a plane (right), the range map-based representation has discretization errors on the points.

On the other hand, the vertex map is represented by a raw point cloud, discretization offsets can be prevented.

B. Proposed Network

The proposed network is composed of two parts, as shown in Fig. 4: a vertex network (*VertexNet*), a normal network (*NormalNet*), and the pose networks (*PoseNet*). *VertexNet* and *NormalNet* use the vertex and normal maps as input in consecutive frames.

VertexNet is used to infer the scale information of motion. We use vertex maps of consecutive frames as input and embed the translational motion into the feature. *NormalNet* is configured to extract the rotation information between two frames. The output from both networks is represented as a feature vector size of 1024, and the sum of the two feature vectors is used as input for *PoseNet*, which is designed as fully-connected networks that transfer features for metric information, and predicts translation and rotation separately.

VertexNet and *NormalNet* are designed based on residual blocks [25] with fully convolutional networks. For *PoseNet*, we construct the decoupled pose estimation $\mathbf{x}_{t,t+1} = [\mathbf{t}, \mathbf{q}]$ composed of translation $\mathbf{t} \in \mathbb{R}^3$ and rotation as quaternion $\mathbf{q} \in \mathbb{R}^4$.

C. Objective Losses

In this section, we introduce two types of objective losses: unsupervised and supervised. Both losses are selectively used according to the validity of the training data.

1) Unsupervised Loss: For unsupervised training, we integrate the ICP method 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 used to obtain point correspondences. Each vertex in the vertex map $\mathbf{v}_{t+1} \in V_{t+1}$ is 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 $\pi(\cdot)$, the mathematical expression for which is

$$\bar{\mathbf{v}}_t = V(\pi(\mathbf{v}'_t)) \quad (2)$$

$$\bar{\mathbf{n}}_t = N(\pi(\mathbf{v}'_t)) \quad (3)$$

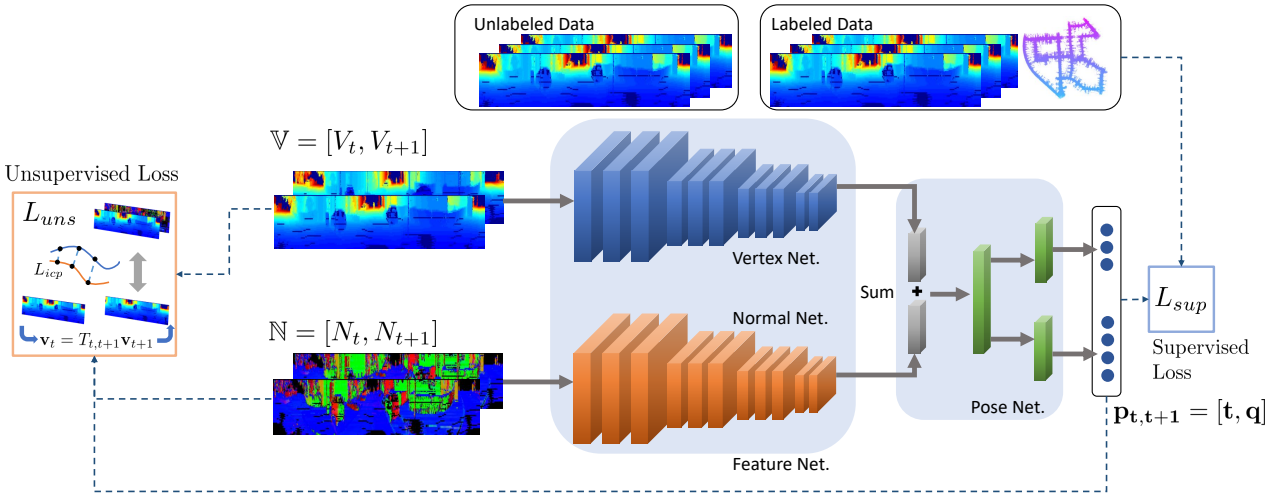


Fig. 4: The proposed network and our unsupervised training scheme. The network is composed of *FeatNet* (composed of *VertexNet* and *NormalNet*) and *PoseNet*. First, as a common part, sequential frames (F_t and F_{t+1}) are fed into the feature extractor (*FeatNet*) for a compact representation. Subsequently, each frame feature is summed as a single vector and forwarded into *PoseNet* which predicts the relative motion $\mathbf{p}_{t,t+1} = [\mathbf{t}, \mathbf{q}]$ of two frames. For the supervised mode when ground-truth motion is available, the predicted motion is converted to the Euler form and directly compared to the ground-truth ($\mathcal{L}_{sup}$). For the unsupervised mode, the output is used to compute the unsupervised loss (ICP loss $\mathcal{L}_{icp}$ and FOV loss $\mathcal{L}_{fov}$) without any ground-truth motion.

where $\bar{\mathbf{v}}_t$ and $\bar{\mathbf{n}}_t$ are corresponding vertex and normal vectors of $\mathbf{v}_{t+1}$ on frame t, respectively. Given the point correspondences, the ICP loss $\mathcal{L}_{icp}$ is

$$\mathcal{L}_{icp} = \sum_{\mathbf{v} \in V_{t+1}} \bar{\mathbf{n}}_t \cdot (T_{t,t+1}\mathbf{v}_{t+1} - \bar{\mathbf{v}}_t), \tag{4}$$

where $\mathcal{L}_{icp}$ represents the sum of the normal distances of the point correspondences.

We also introduce field-of-view loss (FOV loss) $\mathcal{L}_{fov}$ which prevent divergence training to the out of field-of-view condition because although the ICP loss is essential for training convergence, additional regularization is needed for a stable training process. Because the ICP loss $\mathcal{L}_{icp}$ is zero when there are no correspondences, a naïve ICP loss may lead the network to a large relative motion which yields no correspondences. To avoid such cases, we used a penalty loss as a hard-counting loss of out-of-FOV points. The FOV loss is expressed as

$$\mathcal{L}_{fov} = \sum_{\mathbf{v} \in V_{t+1}} \mathbb{I}(\pi(T_{t,t+1}\mathbf{v}) - (w,h)) + \mathbb{I}(-\pi(T_{t,t+1})) \tag{5}$$

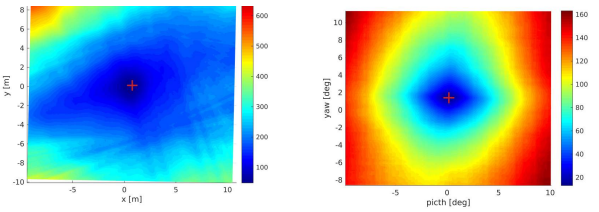
where $\mathbb{I}$ represents the heaviside function and (w, h) are the width and height of the vertex map, respectively. Finally, the overall unsupervised loss is obtained as

$$\mathcal{L}_{uns} = \mathcal{L}_{icp} \exp(-s_{icp}) + s_{icp} + \mathcal{L}_{fov} \exp(-s_{fov}) + s_{fov} \tag{6}$$

where s_{icp} and s_{fov} are trainable scaling factors which balance the magnitude of each loss.

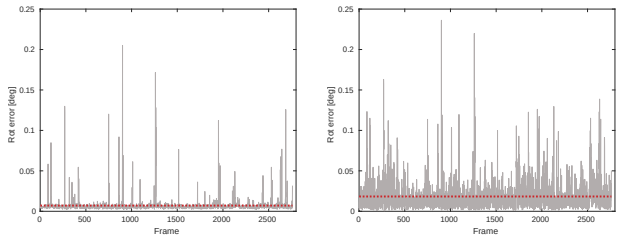
The characteristics of loss $\mathcal{L}_{uns}$ on motion perturbation is depicted in Fig. 5. A red cross sign (+) in each subplot is the loss value on the ground-truth relative pose. We simply add perturbation on translation Fig. 5(a) and rotation Fig. 5(b), and track the unsupervised loss transitions on the motion errors. Each curve on translation and rotation has convex shape around ground-truth. This indicates that the tendency of loss supports the validity of the proposed loss on training.

2) Supervised Loss: In this section, we describe the supervised loss that can be applied when a ground-truth pose or a reference pose (visual odometry) is available. Similar to [26, 27], our network estimates relative rotation as quaternion form which has bounded magnitude of rotation $[-\pi, \pi]$. However, if quaternion subtraction is used as the training loss, normalization is not considered and thus the rotation difference is not reflected correctly. To overcome this issue, we transform the quaternion $\mathbf{q}$ into the Euler angle $\mathbf{r} = [r^x, r^z, r^y]$ in degree. In practice, we find that Euler transformed representation shows better performance and convergence of training. Finally, the supervised loss $\mathcal{L}_{sup}$



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

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



(a) Euler Loss (b) Quaternion Loss

Fig. 6: The rotation error of each frame. (a) errors from Euler-transformed loss. (b) errors from quaternion loss. The red dotted lines are the averages of the errors for all of the frames. We tested both losses under the same training conditions. As can be seen the Euler-transformed loss showed a more stable result on training compared to the quaternion reprsentation.

is expressed as follows,

$$\mathcal{L}_{sup} = \mathcal{L}_t \exp(-s_t) + s_t + \mathcal{L}_r \exp(-s_r) + s_r \qquad (7)$$

where $\mathcal{L}_t = ||\mathbf{t} - \hat{\mathbf{t}}||_l$ is the translation loss with ground-truth $\hat{\mathbf{t}}$, $\mathcal{L}_r = ||\mathbf{r} - \hat{\mathbf{r}}||_l$ is the rotation loss with ground-truth $\hat{\mathbf{r}}$, and $s_{(\cdot)}$ represents trainable scale factors to balance the translation and rotation losses during training. We used L1-norm as loss values.

Fig. 6 clarifies the effect of the proposed rotation loss. We verified both the Euler format Fig. 6(a) and the quaternion format Fig. 6(b) with the same training setup. This plot shows the rotation error of each frame after training had finished. As can be seen, using the proposed representation method guides the network to learn the rotation better.

IV. Experimental Results

In this section, we evaluated the proposed supervised model (DeepLO-Sup) and unsupservised model (DeepLO-Uns) via both qualitative and quantitative comparisons using publicly available datasets, KITTI and Oxford RobotCar.

A. Implementation and Training

The proposed network was implemented using PyTorch and trained with an NVIDIA GTX 1080ti. We employed the Adam solver [28] 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 $\gamma = 0.5$. The scaling factors were initialized with $s_{(\cdot)} = -3$ for automatic scale learning on loss functions (6) and (7), and we set horizontal field of view $f_h = 360°$ and vertical field of view $f_v = 26°$ 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.

Unlike supervised learning, unsupervised learning needs guided training at the start. The initial relative motion from the network has random values, thus we first trained the network with fixed motion beforehand; we employed a simple forward motion (1 m moving forward with no rotation) for the first 20 iterations and then switched the training to unsupervised mode.

B. Evaluation with the KITTI Dataset

We evaluated our method on the well-known odometry datasets of KITTI Vision Benchmark [29]. The KITTI dataset contains 3D point clouds from Velodyne HDL-64E with ground-truth global 6D pose. This dataset has 10 sequences from different environments having dynamic objects (e.g., urban, highways, and streets).

Training details. Similar to previously reported learning-based odometry methods [13, 23], we used sequences `00-08` for the training and `09-10` for the test. In addition, we verified our method via both training strategies (i.e., the supervised and the unsupervised).

Evaluation. Fig. 7 shows the trajectory comparisons of the proposed methods with different strategies. The performance of the trajectories represents the soundness of the network fit (training: `00-08`) and the generality of our method (test: `09-10`). Note that all of the tests had the same parameter settings on both learning models: DeepLO-Sup and DeepLO-Uns.

The trajectories from the training sets (`00` to `08`) of DeepLO-Sup showed well-fitted result to ground-truth, but the performance was relatively low with the test sequences. However, with DeepLO-Uns, the results with both the training and test sequences conveyed similar performance. This finding indicates that unsupervised learning attained a better performance for the generality aspect.

Table I contains the details of the results; the average translation $t_{rel}(\%)$ and rotation $r_{rel}(°/100m)$ RMSE drift on length of $100 - 800$ m. We evaluated our method against several previously reported ones, namely UndeepVO [13], SfMLeaner [12], and Zhu's method [23]; the result values for these were taken from the result in [23]. We also compared our method to SuMa [9] which is recent model-based SLAM using LiDAR measurements. The values of SuMa are referenced from frame-to-frame estimation results.

Compared to learning-based methods, we can see that our method gives better results for learning data sets. Unlike DeepLO-Sup, however, DeepLO-Uns showed a large translation error in sequence `01`. Sequence `01` (highway) includes has dynamic objects, fewer structures and large translational motions than other sequences. Thus, it is difficult to capture true translation with the unsupervised loss which relies on geometric consistency of structures. This aspect also can be seen in other learning-based methods. For the test sequences, DeepLO-Uns performed better than other learning-based methods when DeepLO-Sup was slightly worse than DeepLO-Uns for the test sequences. This is because DeepLO-Sup is overfitted to the training sets, and it is expected that better results will be obtained by adding more validation sets to prevent overfitting. Compared to SuMa, DeepLo-Uns showed excellent performance for learning data because it repeatedly learns the geometric and motion characteristics of the dataset. To achieve better performance in test sequences, we could train more sequences with various

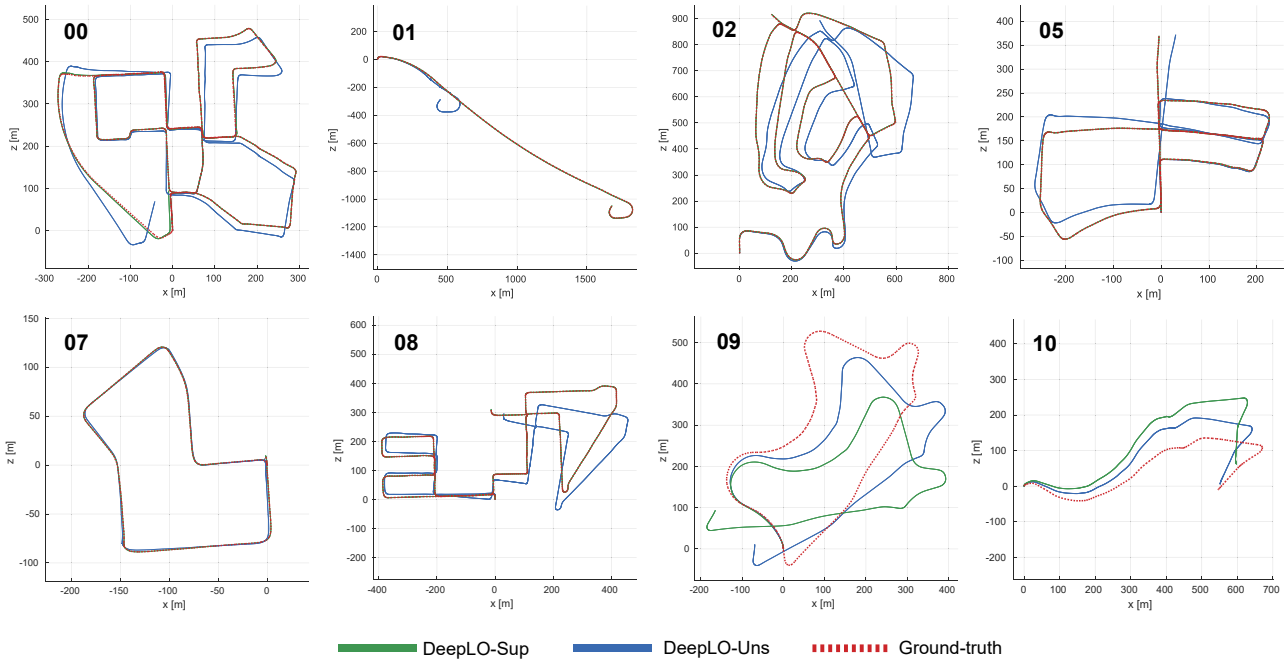


Fig. 7: KITTI trajectory comparison of the proposed method with the supervised learning (DeepLO-Sup) and the unsupervised loss (DeepLO-Uns) and the ground-truth trajectory. As in previous approaches [13, 23, 12], we used sequences `00-08` for training and `09-10` for test.

TABLE I: KITTI odometry evaluation.

	Sequence	0		1		2		3		4		5	
		t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}
Proposed	DeepLO-Uns	1.90	0.80	37.83	0.86	2.05	0.81	2.85	1.43	1.54	0.87	1.72	0.92
	DeepLO-Sup	0.32	0.12	0.16	0.05	0.15	0.05	0.04	0.01	0.01	0.01	0.11	0.07
Learning-based	Zhu et al. [23]	4.56	2.46	78.98	3.03	5.89	2.16	6.84	2.42	9.12	1.42	3.93	2.09
	SfMLearner [12]	66.35	6.13	35.17	2.74	58.75	3.58	10.78	3.92	4.49	5.24	18.67	4.10
	UnDeepVO [13]	4.41	1.92	69.07	1.60	5.58	2.44	5.00	6.17	4.49	2.13	3.40	1.50
Model-based	SuMa [9]	2.10	0.90	4.00	1.20	2.30	0.80	1.40	0.70	11.90	1.10	1.50	0.80

	Sequence	6		7		8		9		10	
		t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}	t_{rel}
Proposed	DeepLO-Uns	0.84	0.47	0.70	0.67	1.81	1.02	6.55	2.19	7.74	2.84
	DeepLO-Sup	0.03	0.07	0.08	0.05	0.09	0.04	13.35	4.45	5.83	3.53
Learning-based	Zhu et al. [23]	7.48	3.76	3.13	2.25	4.81	2.24	8.84	2.92	6.65	3.89
	SfMLearner [12]	25.88	4.80	21.33	6.65	21.90	2.91	18.77	3.21	14.33	3.30
	UnDeepVO [13]	6.20	1.98	3.15	2.48	4.08	1.79	7.01	3.61	10.63	4.65
Model-based	SuMa [9]	1.00	0.60	1.80	1.20	2.50	1.00	1.90	0.80	1.80	1.00

Translation $t_{rel}(\%)$ and rotation $r_{rel}(^\circ/100m)$ RMSE drift on length of $100\text{m} - 800\text{m}$ are presented. Our model was trained on sequences `00-08` along with the compared methods. The RMSE values of the other methods were obtained from [23] and [9].

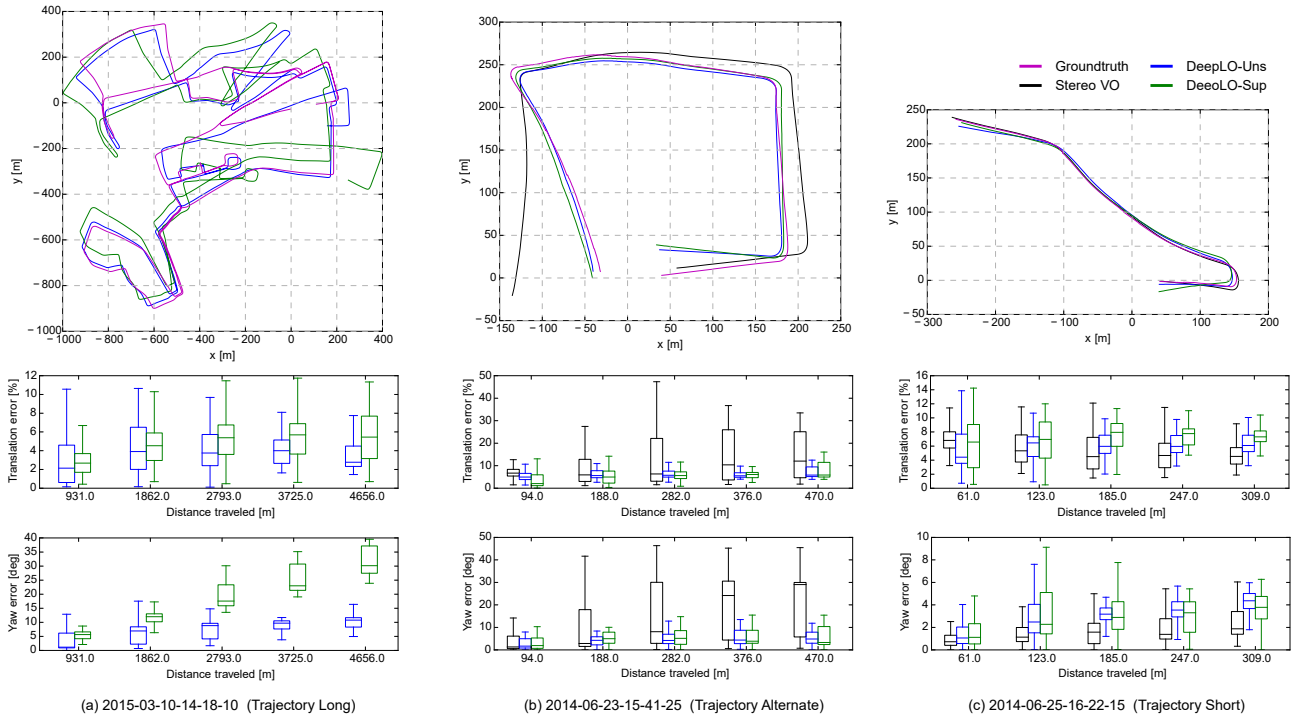
motions and guide the training loss with robust kernels which reduce the effects of dynamic objects.

C. Evaluation with the Oxford RobotCar Dataset

The Oxford RobotCar dataset [3] comprises data collected by repeating the same path dozens of times for long-term autonomy researches.

Dataset preparation. Because this dataset uses a push-broom style 2D LiDAR, information from a single scan is not enough to train a network and infer the robot pose. Therefore, we made a submap with sufficient length (80 m in our work) of accumulated 2D scans using each scan pose interpolated from the inertial navigation system (INS) data. We determined that the ground-truth pose corresponding to the submap was the interpolated global pose (in the world frame) of a center scan. The 3D point coordinates of the submap are represented in the robot frame where the ground-truth global pose of the submap was considered as the origin. Each submap as a 3D scan is sampled per every $1 - 2\text{m}$ using a truncated normal distribution.

Training details. The Oxford Robotcar dataset can be divided into three types: `Long`, `Alternate` and `Short`. Since the focus of the Oxford RobotCar dataset is on seasonal diversity, we used



(a) 2015-03-10-14-18-10 (Trajectory Long) (b) 2014-06-23-15-41-25 (Trajectory Alternate) (c) 2014-06-25-16-22-15 (Trajectory Short)

Fig. 8: Evaluation of the proposed methods via the Oxford RobotCar dataset; trajectories (first row), relative translation errors over distance (second row), and relative heading errors over distance (third row). Each column shows the evaluation with diffent type of sequences. (a) Training: `2014-03-10-14-18-10 Long`, (b) Test: `2014-06-23-15-41-25 Alternate`, and (c) Test: `2014-06-25-16-22-15 Short`. Box plots represent the error statistics: the median (center line), 25% and 75% quantiles (box), and minimum and maximum errors (whisker).

`Long` sequences (`2015-02-03-08-45-10` and `2015-03-10-14-18-10`) as training data and the other two types (`Alternate` and `Short`) as test data. It is notable that the Oxford RobotCar dataset has less trajectory diversity than the KITTI dataset, thus we applied transfer learning to secure network generality and improve learning stability, and we used 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.

Evaluation. We compared our method with the ground-truth trajectory and stereo visual odometry given in the Oxford RobotCar dataset. Since all trajectories have different framerate, we evaluated each method using the trajectory evaluation method [30].

To compute relative errors, sub-trajectory segments of tested methods are selected along different travel distances. Each sub-trajectory is aligned using the first state, and the error are calculated for all the sub-trajectories. Fig. 8 shows the aligned trajectories using the entire trajectory statement. The figure includes the trajectory comparisons to the ground-truth and corresponding error plots (relative translation error (%) and heading error (deg)).

For trajectory alignment, $SE(3)$ transformation was estimated and applied to the tested methods. Each trajectory in Fig. 8 represents the results on the learned sequence (Fig. 8(a)) and test sequences (Fig. 8(b) and (c)), and below each trajectory is the quantitative result compared to the ground-truth. The trajectories might be transformed from the initial position due to the alignment. We compared our methods with the stereo odometry provided by the dataset.

TABLE II: Trajectory Errors on Oxford RobotCar dataset

		Absolute Trajectory Error (RMSE)		
Datetime	Type	StereoVO [3]	DeepLO-S	DeepLO-U
2014-05-14 -13-50-20	Alternate	37.74	14.71	19.93
2014-05-14 -13-59-05	Alternate Reverse	34.55	12.89	22.93
2014-06-23 -15-41-25	Alternate	36.09	16.94	12.80
2014-06-25 -16-22-15	Short	4.22	9.53	6.78

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 stable and comparable performances for all of the sequences with the trained networks being able to capture the relative motion of test sets. Note that our proposed models achieved performances that were better or close to StereoVO [3].

Table II represents absolute trajectory errors (ATE) on test sequences. We compared translation error (m) of all

trajectories. First two sequences of the table are `Alternate` and `Alternate` with reverse direction which is not introduced in Fig. 8. This evaluation quantifies the quality of the whole trajectory. Since ATE is sensitive to the initial results on the path, interesting results can be seen compared to relative errors. Looking at the results for sequence `2014-06-25-16-22-15`, we found that the error at the beginning of DeepLO-Uns and DeepLO-Sup also affected the absolute trajectory.

V. Conclusion

We demonstrated a novel learning-based LiDAR odometry estimation pipeline in an unsupervised and a supervised manner. We suggested surfel-like representations (vertex and normal map) as network inputs without precision loss. 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. 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. In future work, we plan to design the networks and loss functions for large and fast motion such as in sequence `01` of the KITTI dataset and extend our framework to sequential approaches with recurrent neural networks.

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] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding," *arXiv preprint arXiv:1810.06125*, 2018.

[23] 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.

[24] 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.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2016, pp. 770–778.

[26] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-dof camera relocalization," in *Proc. IEEE Intl. Conf. on Comput. Vision*, 2015, pp. 2938–2946.

[27] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2018, pp. 6939–6946.

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

[29] 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.

[30] 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.