LO-Net: Deep Real-time Lidar Odometry

Qing Li[1], Shaoyang Chen[1], Cheng Wang[1], Xin Li[2], Chenglu Wen[1], Ming Cheng[1], Jonathan Li[1]

[1]Xiamen University, Fujian, China

[2]Louisiana State University, Louisiana, USA

{hello.qingli,tinyyoh}@gmail.com, xinli@cct.lsu.ed, {cwang,clwen,chm99,junli}@xmu.edu.cn

Abstract

We present a novel deep convolutional network pipeline, LO-Net, for real-time lidar odometry estimation. Unlike most existing lidar odometry (LO) estimations that go through individually designed feature selection, feature matching, and pose estimation pipeline, LO-Net can be trained in an end-to-end manner. With a new mask-weighted geometric constraint loss, LO-Net can effectively learn feature representation for LO estimation, and can implicitly exploit the sequential dependencies and dynamics in the data. We also design a scan-to-map module, which uses the geometric and semantic information learned in LO-Net, to improve the estimation accuracy. Experiments on benchmark datasets demonstrate that LO-Net outperforms existing learning based approaches and has similar accuracy with the state-of-the-art geometry-based approach, LOAM.

1. Introduction

Estimating 3D position and orientation of a mobile platform is a fundamental problem in 3D computer vision, and it provides important navigation information for robotics and autonomous driving. Mobile platforms usually collect information from the real-time perception of the environment and use on-board sensors such as lidars, Inertial Measurement Units (IMU), or cameras, to estimate their motions. Lidar can obtain robust features of different environments as it is not sensitive to lighting conditions, and it also acquires more accurate distance information than cameras. Therefore, developing an accurate and robust real-time lidar odometry estimation system is desirable.

Classic lidar-based registration methods used in pose estimation include Iterative Closest Point (ICP) [3], ICP variants [24], and feature-based approaches [26]. But due to the nonuniformity and sparsity of the lidar point clouds, these methods often fail to match such data. ICP approaches find transformations between consecutive scans by minimizing distances between corresponding points from these scans, but points in one frame may miss their spatial coun-
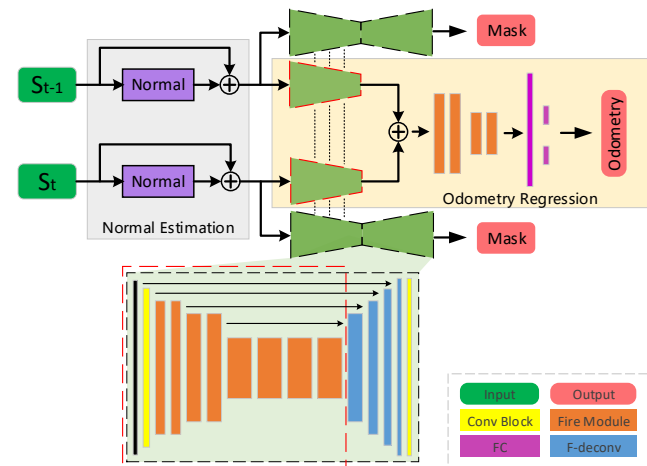

Figure 1. Top: Data stream of LO-Net. Bottom: Network architecture of feature extraction layers (red dashed line) and mask prediction layers (black dashed line) . Our network takes two consecutive lidar scans as input and infers the relative 6-DoF pose. The output data will be further processed by the mapping module.

terparts in the next frame, due to sparsity of scan resolution. Feature-based methods are less sensitive to the quality of scans, and hence, are more powerful. However, they are usually more computationally expensive. Furthermore, most feature-based methods are sensitive to another environmental factor, dynamic objects in the scene. These two issues inhibit many feature-based methods from producing effective odometry estimation.

Recently, deep learning-based methods have outperformed classic approaches in many computer vision problems. Many Convolutional Neural Networks (CNNs) architectures and training models have become the state-of-the-art in these tasks. However, the exploration of effective CNNs in some 3D geometric data processing problems, such as 6-DoF pose estimation, has not been this successful yet. Although quite a few CNN-based 6-DoF pose estimation (from RGB images) strategies [34, 43, 36, 39] have been explored recently, these methods often suffer from

the inaccurate depth prediction and scale drift. This paper proposes a new deep neural network architecture for lidar odometry estimation. We are inspired by the recent CNNs-based camera localization and pose regression works [43, 2, 16, 37] in the context of network structure design, and the traditional lidar odometry methods [41, 21, 7] in the aspect of lidar mapping. The pipeline is illustrated in Figure 1. Our method can better accumulate motion-specific features by incorporating pairwise scans, better interpret the spatial relations of scans by applying normal consistency, and better locate effective static regions using a simultaneously learned uncertainty mask.

The main contributions of this paper are as follows: 1) We propose a novel scan-to-scan lidar odometry estimation network which simultaneously learns to estimate the normal and a mask for dynamic regions. 2) We incorporate a spatiotemporal geometry consistency constraint to the network, which provides higher-order interaction between consecutive scans and helps better regularize odometry learning. 3) We couple an efficient mapping module into the estimation pipeline. By utilizing the normal and mask information learned from LO-Net, we can achieve a still real-time but more accurate odometry estimation.

We compared our approach with existing representative methods on commonly used benchmark datasets *KITTI* [10, 11] and *Ford Campus Vision and Lidar Data Set* [22]. Our approach achieves the state-of-the-art results. To our best knowledge, this is the first neural network regression model that is comparable to the state-of-the-art geometry feature based techniques in 3D lidar odometry estimation.

2. Related work

A. ICP variants

Most existing lidar-based pose estimation algorithms are built upon variants of the ICP method [3, 24]. ICP iteratively matches the adjacent scans and estimates the pose transformation, by minimizing distances between corresponded elements, until a specific termination condition is met. Despite its wide applicability, ICP is computationally expensive and sensitive to initial poses. Various ICP variants, such as point-to-plane or plane-to-plane ICP, were developed to improve ICP's convergence speed and robustness against local minima.

By combining point-to-point ICP and point-to-plane ICP, Segal *et al.* [28] introduce a plane-to-plane strategy called Generalized ICP (GICP). The covariance matrices of the local surfaces is used to match corresponding point clouds. Their results show better accuracy over original ICP. The Normal Iterative Closest Point (NICP) [29] takes into account the normal and curvature of each local surface point. Experiments show that NICP can offer better robustness and performance than the GICP.

Grant *et al.* [12] derive a novel Hough-based voting scheme for finding planes in laser point clouds. A plane-based registration method [23], that aligns detected planes, is adopted to compute the final transformation. For the non-uniformly sampled point clouds from laser sensors, Serafin *et al.* [30] present a fast method to extract robust feature points. They remove ground points by a flat-region-removal method, then extract key-points of lines and planes, and use them to estimate the transformation. They show comparative results against the NARF key-point detector [31] and highlight that the feature extraction algorithm is applicable to the Simultaneous Localization and Mapping (SLAM) problem [13, 5]. Similarly, Douillard *et al.* [8] remove the ground points by voxelization, cluster the remaining points into the segments, then match these segments through a modified ICP algorithm.

To overcome the sparsity of lidar point clouds, Collar Line Segments (CLS) [32] groups the points into polar bins, and generates line segments within each bin. Then, ICP can be used to register the corresponding lines and estimate the transformation between two scans. Although it produces better results than GICP [28], CLS is not real-time because the line segments computation is very slow.

Over the past few years, Lidar Odometry And Mapping (LOAM) [41, 42] has been considered as the state-of-the-art lidar motion estimation method. It extracts the line and plane features in lidar data, and saves these features to the map for edge-line and plane-surface matching. LOAM dose not consider the dynamic objects in the scene and achieves low-drift and real-time odometry estimation by having two modules running in parallel. The estimated motion of scan-to-scan registration is used to correct the distortion of point clouds and guarantee the real-time performance. Finally, the odometry outputs are optimized through a map.

B. Deep learning-based methods

Deep learning has achieved promising results on the issues of visual odometry (VO) [34, 43, 40, 36], image-based pose estimation or localization [17, 16, 4], and point cloud classification and segmentation [25, 19]. Unfortunately, using deep learning methods to solve 3D lidar odometry problems still remains challenging. Velas *et al.* [33] use CNNs to do lidar odometry estimation on lidar scan sequences. To train the CNNs, the original lidar data is transformed into dense matrix with three channels. However, this model formulates the odometry estimation as a classification problem, rather than a numerical regression, and it only estimates translational parameters. Hence, is not competent for accurate 6-DoF parameters estimation. The method of [33] and ours are conceptually similar, but our network better handles uncertain dynamic regions through mask prediction, and more effectively uses spatiotemporal consistency to regularize the learning for stable estimations.

3. Method

Odometry estimation algorithm uses consecutive point clouds obtained by a moving robot as inputs. Our LO-Net can interpret and exploit more information by performing sequential learning rather than processing a single scan. And the features learned by LO-Net encapsulate geometric information for the LO problem. As shown in Figure 1, our LO-Net architecture consists of a normal estimation sub-network, a mask prediction sub-network, and a Siamese pose regression main-network. LO-Net takes two consecutive scans $(S_{t-1}; S_t)$ as input and jointly estimates the 6-DoF relative pose between the scans, point-wise normal vector, and a mask of moving objects for each scan. Despite being jointly trained, the three modules can be used independently for inference. The odometry outputs of the LO-Net are then refined through a lidar mapping, which registers the lidar point clouds onto a globally constructed map. The final output is the transformation of scan S_t with respect to the initial reference position, namely, the pose of each S_t in the world coordinate system.

3.1. Lidar data encoding

As shown in Figure 2, the 3D lidar point clouds with ring structures are usually represented by Cartesian coordinates. Additional information includes intensity values. To convert the original sparse and irregular point clouds into a structured representation that can be fed into networks, we encode the lidar data into point cloud matrices by a cylindrical projection [6]. Given a 3D point $p = (x, y, z)$ in lidar coordinate system (X, Y, Z), the projection function is

$$
\begin{aligned}
\alpha &= \arctan(y/x)/\Delta\alpha \\
\beta &= \arcsin(z/\sqrt{x^2 + y^2 + z^2})/\Delta\beta
\end{aligned}
\tag{1}
$$

where α and β are the indexes which set the points' positions in the matrix. $\Delta\alpha$ and $\Delta\beta$ are the average angular resolution between consecutive beam emitters in the horizontal and vertical directions, respectively. The element at (α, β) of the point cloud matrix is filled with intensity value and range value $r = \sqrt{x^2 + y^2 + z^2}$ of the lidar point p. We keep the point closer to the lidar when multiple points are projected into a same position. After applying this projection on the lidar data, we get a matrix of size $H \times W \times C$, and C is the number of matrix channels. An example of the range channel of the matrix is shown in Figure 9.

3.2. Geometric consistency constraint

Normal estimation. As shown in Figure 2, given a 3D point X^i and its k neighbors X^{i_j}, $j = 1, 2, \ldots, k$ on the grid. The normal $\mathcal{N}(X^i)$ can be estimated by

$$
\arg\min_{\mathcal{N}(X^i)} \|[w_{i1}(X^{i_1} - X^i), \cdots, w_{ik}(X^{i_k} - X^i)]^T \mathcal{N}(X^i)\|_2
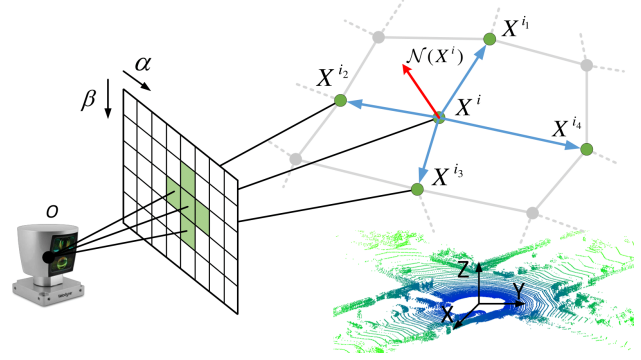\tag{2}
$$

Figure 2. Illustration of data encoding and normal estimation

where $(X^{i_k} - X^i)$ is a 3D vector, w_{ik} is the weight of X^{i_k} with respect to X^i, and $[\cdot]^T$ is a $k \times 3$ vector. $\|\mathcal{N}(X^i)\|_2 = 1$. We set $w_{ik} = \exp(-0.2|r(X^{i_k}) - r(X^i)|)$ to put more weight on points which have similar range value r with X^i, and less weight otherwise. A commonly adopted strategy for solving Equation (2) is to perform a Principal Component Analysis (PCA), and convert the normal estimation to the computation of eigenvectors/eigenvalues of the covariance matrix created from X^i's neighbors. In our work, this normal estimation needs to be embedded into the network. The computation of covariance matrices and their eigenvectors makes the training and estimation inefficient. Hence, we simplify the normal estimation by computing the weighted cross products over X^i's four neighbors. Then we smooth normal vectors using a moving average filter [20]. This can be formulated as

$$
\mathcal{N}(X^i) = \sum_{X^{i_k}, X^{i_j} \in \mathcal{P}} (w_{ik}(X^{i_k} - X^i) \times w_{ij}(X^{i_j} - X^i))
\tag{3}
$$

where $\mathcal{P}$ is the set of neighboring points of X^i, sorted counterclockwisely, as $\{X^{i_1}, X^{i_2}, X^{i_3}, X^{i_4}\}$ shown in Figure 2. The final normal vectors are normalized to 1.

Due to the temporal spatial geometric consistency of scan sequences, the points in a point cloud matrix should have the corresponding ones in another. Let $X_{t-1}^{\alpha\beta}$ and $X_t^{\alpha\beta}$ be the spatial corresponding point elements of the consecutive data matrices S_{t-1} and S_t, respectively. We can derive $\hat{X}_t^{\alpha\beta}$ from $X_{t-1}^{\alpha\beta}$ through

$$
\hat{X}_t^{\alpha\beta} = PT_tP^{-1}X_{t-1}^{\alpha\beta}
\tag{4}
$$

where T_t is the relative rigid pose transformation between the consecutive scans. P denotes the projection process and P^{-1} is its inverse operation. Therefore, $\hat{X}_t^{\alpha\beta}$ and $X_t^{\alpha\beta}$ are a pair of matching elements, and we can measure the similarity between the corresponding elements to verify the correctness of pose transformation. The noise inevitably exists in the coordinate and intensity values because of lidar

measurements. We compare the normal $\mathcal{N}(x)$ as it reflects smooth surface layouts and clear edge structures of the road environment (see Figure 8). Thus, the constraint of pose transformation can be formulated as minimizing

$$\mathcal{L}_n = \sum_{\alpha\beta} \|\mathcal{N}(\hat{X}_t^{\alpha\beta}) - \mathcal{N}(X_t^{\alpha\beta})\|_1 \cdot e^{|\nabla r(\hat{X}_t^{\alpha\beta})|} \quad (5)$$

where $\nabla r(\hat{X}_t^{\alpha\beta})$ is a local range smooth measurement, and ∇ is the differential operator with α and β. The item $e^{|\cdot|}$ allows the loss function to focus more on sharply changing areas in the scene.

3.3. Lidar odometry regression

To infer the 6-DoF relative pose between the scans S_{t-1} and S_t, we construct a two-stream network. The input to the network is encoded point cloud matrices with point-wise normal vectors. As shown in Figure 1, LO-Net concatenates the features coming from two individual streams of feature extraction networks, then pass these concatenated features to the following four convolution units. The last three fully-connected layers output the 6-DoF pose between the input scans. The last two fully-connected layers are of dimensions 3 and 4, for regressing the translation x and rotation quaternion q, respectively. Finally, we get a 7D vector, which can be converted to a 4×4 transformation matrix.

In order to reduce the number of model parameters and computation cost, we replace most of convolution layers of odometry regression network with fireConv [15], which has been used for object detection from lidar point clouds [35]. We follow [15] to construct our feature extraction layers. Since the width of intermediate features is much larger than its height, we only down-sample the width by using max-pooling during feature extraction. The detailed network architecture is illustrated in the supplementary material.

Learning position and orientation simultaneously. In our method, we choose the quaternion to represent the rotation as it is a continuous and smooth representation of rotation. The quaternions q and $-q$ map to the same rotation, so we need to constrain them to a unit hemisphere. We use $\mathcal{L}_x(S_{t-1}; S_t)$ and $\mathcal{L}_q(S_{t-1}; S_t)$ to demonstrate how to learn the relative translational and rotational components, respectively.

$$\mathcal{L}_x(S_{t-1}; S_t) = \|x_t - \hat{x}_t\|_l$$
$$\mathcal{L}_q(S_{t-1}; S_t) = \left\| q_t - \frac{\hat{q}_t}{\|\hat{q}_t\|} \right\|_l \quad (6)$$

where x_t and q_t are the ground truth relative translational and rotational components, $\hat{x}_t$ and $\hat{q}_t$ denote their predicted counterparts. l refers to the distance norm in Euclidean space, and we use the ℓ_2-norm in this work. Due to the difference in scale and units between the translational and

rotational pose components, previous works [17, 34] gave a weight regularizer λ to the rotational loss to jointly learn the 6-DoF pose. However, the hyper-parameter λ need to be tuned when using new data from different scene. To avoid this problem, we use two learnable parameters s_x and s_q to balance the scale between the translational and rotational components in the loss term [16].

$$\mathcal{L}_o = \mathcal{L}_x(S_{t-1}; S_t) \exp(-s_x) + s_x \\ + \mathcal{L}_q(S_{t-1}; S_t) \exp(-s_q) + s_q \quad (7)$$

We use the initial values of $s_x = 0.0$ and $s_q = -2.5$ for all scenes during the training.

3.4. Mask prediction

Given two consecutive scans S_{t-1} and S_t in a static rigid scene, we can get point matching relationships of the encoded data matrix pairs through transformation and cylindrical projection, as illustrated in Section 3.2. Lidar point clouds are considered as the 3D model of the scene, and often contain dynamic objects such as cars and pedestrians in the road environment. These factors may inhibit the learning pipeline of odometry regression.

Based on the encoder-decoder architecture (see Figure 1), we deploy a mask prediction network [43, 38] to learn the compensation for dynamic objects, and improve the effectiveness of the learned features and the robustness of the network. The encoding layers share parameters with feature extraction layers of the odometry regression network, and we train these two networks jointly and simultaneously. The deconvolution layers are variants of fireDeconv [15], and adopt the skip-connection. The detailed network architecture is described in the supplementary material.

The predicted mask $\mathcal{M}(X_t^{\alpha\beta}) \in [0, 1]$ indicates the area where geometric consistency can be modeled or not, and implicitly ensures the reliability of the features learned in the pose regression network. Therefore, the geometric consistency error as formulated in Equation (5) is weighted by

$$\mathcal{L}_n = \sum_{\alpha\beta} \mathcal{M}(X_t^{\alpha\beta})\|\mathcal{N}(\hat{X}_t^{\alpha\beta}) - \mathcal{N}(X_t^{\alpha\beta})\|_1 \cdot e^{|\nabla r(\hat{X}_t^{\alpha\beta})|}.$$
$$(8)$$

Note that there is no ground truth label or supervision to train the mask prediction. The network can minimize the loss by setting all values of the predicted mask to be 0. To avoid this, we add a cross-entropy loss as a regularization term

$$\mathcal{L}_r = -\sum_{\alpha\beta} \log P(\mathcal{M}(X_t^{\alpha\beta}) = 1). \quad (9)$$

In summary, our final objective function to minimize for odometry regression is

$$\mathcal{L} = \mathcal{L}_o + \lambda_n \mathcal{L}_n + \lambda_r \mathcal{L}_r \quad (10)$$

where λ_n and λ_r are the weighting factors for geometric consistency loss and mask regularization, respectively.

3.5. Mapping: scan-to-map refinement

Consecutive scan-to-scan matches could introduce accumulative error, and also, may suffer when available common feature points between consecutive frames are limited. Hence, we maintain a global map reconstructed from the previously scans, then use the registration between this map and the current scan to refine the odometry. Unlike the traditional scan-to-map approaches [41, 42], which directly match all the detected edge and plane points, we use the normal information (estimated by LO-Net) to select points from smooth area and use the mask (also an output of LO-Net) to exclude points from moving object.

At a test time t, let $[T_t, S_t]$ be the data from LO-Net. T_t is the odometry computed by LO-Net, which is used as the initial pose for mapping. S_t is a multi-channel data matrix containing the intensity, range, normal, and mask values of each point. The coordinates of each point can be calculated from its range value. The mapping takes the scan and the odometry as input, then matches the point cloud onto the global map. Figure 3 shows the diagram of our mapping module. The details of diagram are as follows:

$*$: Based on the normal channels of S_t, we define a term c to evaluate the smoothness of the local area.

$$c = \sum_{k=1}^{3}(K * \mathcal{N}_k)^2 \tag{11}$$

where $\mathcal{N}_k$ is the normal vector channel of S_t. The symbol $*$ denotes the convolution operation. K is a 3×5 convolution kernel. For K, the value of the central element is -14, and the others are 1. We compute c for each point in S_t, and sort the values in increasing order. The first n_c points in the list, except for marked points of the mask, are selected as planar points as they are in the smooth area.

$\mathbf{\Pi}$: Compute an initial estimate of the lidar pose relative to its first position: $\mathbf{M}_{init} = \mathbf{M}_{t-1}\mathbf{M}_{t-2}^{-1}\mathbf{M}_{t-1}$, where $\mathbf{M}_t$ is the lidar transformation at time t.

$\mathbf{\Psi}$: Eliminate the motion distortion of lidar point cloud from S_t by compensating for the ego-motion using a linear interpolation of T_t. Then, use $\mathbf{M}_{init}$ to transform the corrected scan S_t to the global coordinate system in which the map is located, and prepare for matching.

Suppose $\boldsymbol{p}_i = (p_{i_x}, p_{i_y}, p_{i_z}, 1)^T$ is a point in the scan S_t, $\boldsymbol{m}_i = (m_{i_x}, m_{i_y}, m_{i_z}, 1)^T$ is the corresponding point in the map built by the previous scans, $\boldsymbol{n}_i = (n_{i_x}, n_{i_y}, n_{i_z}, 0)^T$ is the unit normal vector at $\boldsymbol{m}_i$. The goal of mapping is to find the optimal 3D rigid-body transformation

$$\hat{\mathbf{M}}_{opt} = \underset{\hat{\mathbf{M}}}{\arg\min} \sum_{i}((\hat{\mathbf{M}} \cdot \boldsymbol{p}_i - \boldsymbol{m}_i) \cdot \boldsymbol{n}_i)^2 . \tag{12}$$
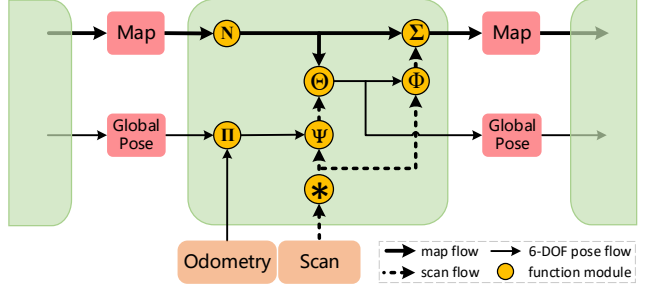

Figure 3. The mapping module consecutively computes the low-drift motion of the lidar in the world coordinate system and builds a 3D map for the traversed environment using the lidar data. The specific meanings of the function symbols are explained in the text.

$\mathbf{\Theta}$: Iteratively register the scan onto the map by solving Equation (12) until a maximum number of iteration n_{iter}. Then, calculate the final transformation $\mathbf{M}_t$ by accumulating the transformation during the iteration $\hat{\mathbf{M}}_k$ and the initial pose $\mathbf{M}_{init}$

$$\mathbf{M}_t = \prod_{k=1}^{n_{iter}} \hat{\mathbf{M}}_k \mathbf{M}_{init} . \tag{13}$$

$\mathbf{\Phi}$: Generate a new point cloud from the current scan S_t by linear interpolation of vehicle motion between $\mathbf{M}_{t-1}$ and $\mathbf{M}_t$.

$\mathbf{\Sigma}$, $\mathbf{N}$: Add the new point cloud to the map. Then, remove the oldest point cloud scans to only maintain n_m transformed scans in the map.

This mapping-based refinement is performed iteratively along with the scan sequence. It can further improve the accuracy of odometry estimation, as shown in Section 4.2.

4. Experiments

Implementation details. The point cloud data we use is captured by the Velodyne HDL-64 3D lidar sensor(64 laser beams, 10Hz, collecting about 1.3 million points/second). Therefore, during encoding the data matrix, we set $H = 64$ and $W = 1800$ by considering the sparseness of point clouds. The width of input data matrix are resized to 1792 by cropping both ends of the matrix. During the training, the length of input sequence is set to be 3. We form the temporal pairs by choosing scan $[S_{t-2}, S_{t-1}], [S_{t-1}, S_t], [S_{t-2}, S_t]$. LO-Net predicts relative transformation between the pairs. The whole framework is implemented with the popular Tensorflow library [1]. During the training, the mask prediction network is pre-trained using KITTI 3D object detection dataset, and all layers are trained simultaneously. The initial learning rate is 0.001 and exponentially decays after every 10 epochs until 0.00001. The loss weights of Equation (10) are set to be $\lambda_n = 0.15$ and $\lambda_r = 0.05$, and the batch size is 8. We
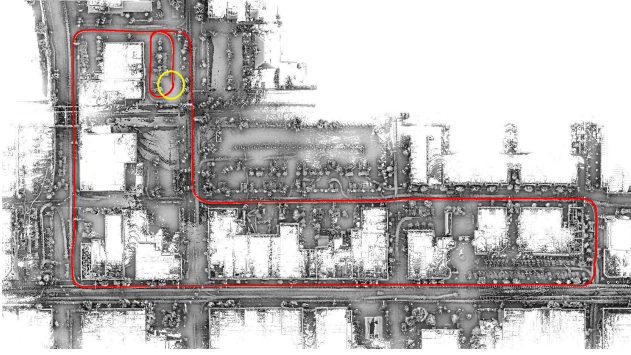
Figure 4. Trajectory (in red) and the built map of LO-Net+Mapping on Ford dataset 1. Our reconstructed trajectory (without enforcing loop closure) has small drift and forms closed loops accurately (yellow circle).

Figure 5. 3D trajectory plots of our method for KITTI Seq. 10. The mapping module effectively reduces the vertical drift generated in LO-Net.

choose the Adam [18] solver with default parameters for optimization. The network is trained on an NVIDIA 1080 Ti GPU. For the mapping, we set the number of points in each scan $n_c = 0.01HW$, the number of scans in the map $n_m = 100$, and the number of iterations $n_{iter} = 15$.

4.1. Datasets

KITTI. The KITTI odometry dataset [10, 11] consists of 22 independent sequences with stereo gray-scale and color camera images, point clouds captured by a Velodyne lidar sensor, and calibration files. Sequences 00-10 (23201 scans) are provided with ground truth pose obtained from the IMU/GPS readings. For sequences 11-21 (20351 scans), there is no ground truth available, and are provided for benchmarking purposes. The dataset is captured during driving in a variety of road environments with dynamic objects and vegetation, such as highways, country roads and urban areas. The driving speed is up to 90km/h.

FORD. The Ford Campus Vision and Lidar Data Set [22] consists time-synchronized 2D panoramic image, 3D lidar point clouds and IMU/GPS data. Like KITTI, the lidar dataset is captured using horizontally scanning 3D lidar mounted on the top of a vehicle. The dataset contains two loop closure sequences collected in different urban environments, and there are more moving vehicles than the KITTI dataset.

4.2. Odometry results

Baselines. We compare our approach with several classic lidar odometry estimation methods: ICP-point2point (ICP-po2po), ICP-point2plane (ICP-po2pl), GICP [28], CLS [32], LOAM [42] and Velas *et al.* [33]. The first two ICP methods are implemented using the Point Cloud Library [27]. As far as we know, [33] is the only deep learning based lidar odometry method that has comparable results, but it has no codes publicly available. We obtain the
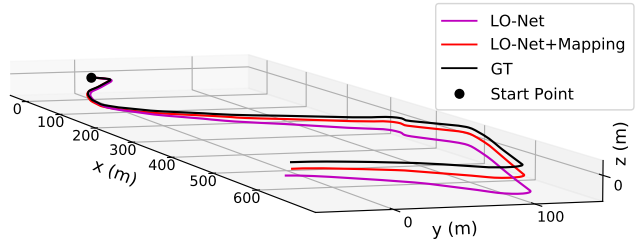
results of other evaluated methods using the publicly available code, and some of the results are even better than those in the originally published papers. For LOAM algorithm, it achieves the best results among lidar-based methods in the KITTI odometry evaluation benchmark [11]. In order to enable the map-based optimization of LOAM to run for every input scan and determine the full potential of the algorithm, we make some modifications and parameter adjustments in the originally published code. Loop closure detection is not implemented for all methods since we aim to test the limits of accurate odometry estimation.

We firstly conduct the training and testing experiments on the KITTI dataset. Then, based on the model trained only on the KITTI dataset, we directly test the model on the Ford dataset without any further training or fine-tuning. We use the KITTI odometry evaluation metrics [11] to quantitatively analyze the accuracy of odometry estimation. Table 1 shows the evaluation results of the methods on KITTI and Ford datasets. It can be seen that the results of LO-Net+Mapping are slightly better than LOAM and clearly superior to the others. Although there are differences between the two datasets, such as different lidar calibration parameters and different systems for obtaining ground truth, our approach still achieves the best average performance among evaluated methods. Figure 5 shows the 3D trajectory plots of our method at different stages. Some trajectories produced by different methods are shown in Figure 4 and 6. Figure 7 shows the average evaluation errors on KITTI Seq. 00-10. More estimated trajectories for KITTI and Ford datasets are shown in our supplementary material.

Ablation study. In order to investigate the importance of different loss components proposed in Section 3, we perform an ablation study on KITTI dataset by training and testing the LO-Net using different combinations of the losses. As shown in Table 2, the network achieves the best average performance when it is trained with the full loss.

4.3. Normal results

Since KITTI and Ford datasets do not provide a benchmark for normal evaluation, we compare the normal esti-

Table 1. Odometry results on KITTI and Ford datasets. Our network is trained on KITTI sequences and then tested on the two datasets.

Seq.	ICP-po2po		ICP-po2pl		GICP [28]		CLS [32]		LOAM [42][1]		Velas et al. [33][2]		LO-Net		LO-Net+Mapping	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00[†]	6.88	2.99	3.80	1.73	1.29	0.64	2.11	0.95	1.10 (**0.78**)	0.53	3.02	NA	1.47	0.72	**0.78**	**0.42**
01[†]	11.21	2.58	13.53	2.58	4.39	0.91	4.22	1.05	2.79 (1.43)	0.55	4.44	NA	**1.36**	0.47	1.42	**0.40**
02[†]	8.21	3.39	9.00	2.74	2.53	0.77	2.29	0.86	1.54 (**0.92**)	0.55	3.42	NA	1.52	0.71	1.01	**0.45**
03[†]	11.07	5.05	2.72	1.63	1.68	1.08	1.63	1.09	1.13 (0.86)	0.65	4.94	NA	1.03	0.66	**0.73**	**0.59**
04[†]	6.64	4.02	2.96	2.58	3.76	1.07	1.59	0.71	1.45 (0.71)	**0.50**	1.77	NA	**0.51**	0.65	0.56	0.54
05[†]	3.97	1.93	2.29	1.08	1.02	0.54	1.98	0.92	0.75 (**0.57**)	0.38	2.35	NA	1.04	0.69	0.62	**0.35**
06[†]	1.95	1.59	1.77	1.00	0.92	0.46	0.92	0.46	0.72 (0.65)	0.39	1.88	NA	0.71	0.50	**0.55**	**0.33**
07[*]	5.17	3.35	1.55	1.42	0.64	0.45	1.04	0.73	0.69 (0.63)	0.50	1.77	NA	1.70	0.89	**0.56**	**0.45**
08[*]	10.04	4.93	4.42	2.14	1.58	0.75	2.14	1.05	1.18 (1.12)	0.44	2.89	NA	2.12	0.77	**1.08**	**0.43**
09[*]	6.93	2.89	3.95	1.71	1.97	0.77	1.95	0.92	1.20 (**0.77**)	0.48	4.94	NA	1.37	0.58	**0.77**	**0.38**
10[*]	8.91	4.74	6.13	2.60	1.31	0.62	3.46	1.28	1.51 (**0.79**)	0.57	3.27	NA	1.80	0.93	0.92	**0.41**
mean[†]	7.13	3.08	5.15	1.91	2.23	0.78	2.11	0.86	1.35 (0.85)	0.51	3.12	NA	1.09	0.63	**0.81**	**0.44**
mean[*]	7.76	3.98	4.01	1.97	1.38	0.65	2.15	1.00	1.15 (**0.83**)	0.50	3.22	NA	1.75	0.79	**0.83**	**0.42**
Ford-1	8.20	2.64	3.35	1.65	3.07	1.17	10.54	3.90	1.68	0.54	NA	NA	2.27	0.62	**1.10**	**0.50**
Ford-2	16.23	2.84	5.68	1.96	5.11	1.47	14.78	4.60	1.78	0.49	NA	NA	2.18	0.59	**1.29**	**0.44**

[1]: The results on KITTI dataset outside the brackets are obtained by running the code, and those in the brackets are taken from [42].
[2]: The results on KITTI dataset are taken from [33], and the results on Ford dataset are not available.
[†]: The sequences of KITTI dataset that are used to train LO-Net.
[*]: The sequences of KITTI dataset that are not used to train LO-Net.
t_{rel}: Average translational RMSE (%) on length of 100m-800m.
r_{rel}: Average rotational RMSE ($°$/100m) on length of 100m-800m.
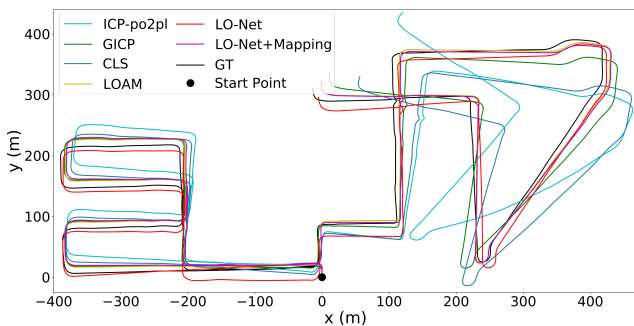

Figure 6. Trajectory plots of KITTI Seq. 08 with ground truth. Our LO-Net+Mapping produces most accurate trajectory.
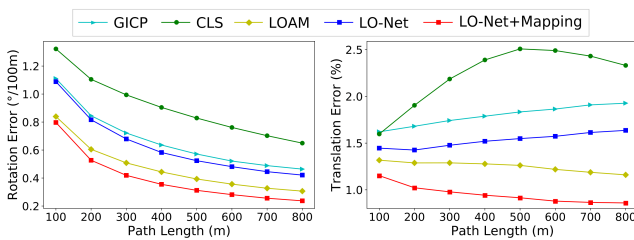

Figure 7. Evaluation results on the KITTI Seq. 00-10. We show the average errors of translation and rotation with respect to path length intervals. Our LO-Net+Mapping achieves the best performance among all the evaluated methods.

mation with that computed from the methods of PCA and Holzer [14]. The PCA estimates the surface normal at a point by fitting a least-square plane from its surrounding neighboring points. In our experiment, we choose the radius

Table 2. Comparison of different combinations of the losses. The mean values of translational and rotational RMSE on KITTI training and testing sequences are computed as in Table 1. $\mathcal{L}'_n$ indicates that the geometric consistency loss is not weighted by the mask.

Seq.	$\mathcal{L}_o$		$\mathcal{L}_o, \mathcal{L}'_n$		$\mathcal{L}_o, \mathcal{L}_n, \mathcal{L}_r$	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
mean[†]	1.46	1.01	1.18	0.70	**1.09**	**0.63**
mean*	2.03	1.50	1.80	0.82	**1.75**	**0.79**

$r = 0.5$m and $r = 1.0$m as the scale factor to determine the set of nearest neighbors of a point. As shown in Figure 8, our estimated normals can extract smooth scene layouts and clear edge structures.

For quantitative comparison purpose, the normals computed from PCA with $r = 0.5$m are interpolated and used as the ground truth. Then a point-wise cosine distance is employed to compute the error between the predicted normal and the ground truth.

$$e_i = \arccos(\boldsymbol{n}_i \cdot \hat{\boldsymbol{n}}_i), \quad \hat{\boldsymbol{n}}_i \in \mathcal{N} \qquad (14)$$

where the angle e_i is the normal error at point p_i, $\boldsymbol{n}_i$ and $\hat{\boldsymbol{n}}_i$ are ground truth and predicted normal vector of point p_i, respectively. $\hat{\boldsymbol{n}}_i \in \mathcal{N}$ denotes the point p_i is a valid point with ground truth normal. The normal evaluations performed on KITTI dataset are shown in Table 3, our approach outperforms the others under most metrics. The metrics include the mean and median values of e_i, and the percent-good-normals whose angle fall within the given thresholds [9, 37]. "GT median" which denotes that we set a normal direction for all points with the median value of the ground truth, is employed as a baseline. The evaluation demonstrates that
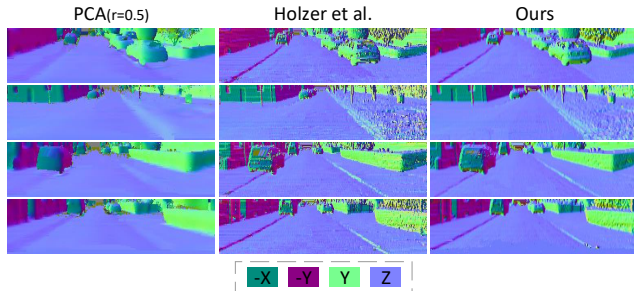
Figure 8. Visual comparison of normal results on KITTI dataset. Different colors indicate different normal directions. Our results show smooth surface layouts and clear edge structures. The images are cropped and reshaped for better visualization.

Table 3. Normal performances of our method and the baseline methods on KITTI dataset.

| Method | (Lower Better) | | (Higher Better) | | |
	Mean	Median	$< 11.25°$	$< 22.5°$	$< 30°$
GT median	23.38	5.78	0.632	0.654	0.663
PCA(r=1.0)	14.38	11.55	0.470	**0.895**	**0.946**
Holzer *et al.* [14]	13.18	5.19	0.696	0.820	0.863
Ours	**10.35**	**3.29**	**0.769**	0.865	0.897

our estimated normal can serve as a reliable property of road scene for the geometric consistency constraint.

4.4. Mask visualization

Examples of the mask predicted by our network are visualized in Figure 9. The highlighted areas suggest that LO-Net has learned to identify dynamic objects and tend to mask vegetation as unexplainable, and indicate that the network will pay less attention to these areas in the odometry regression. Dynamic objects and the relationships between scan sequences are important for odometry estimation problems. They are difficult to explicitly model but implicitly learned by our network.

4.5. Runtime

The lidar scanning point cloud are captured one by one over time, and processing these data in time is critical for robotic applications. Note that unlike image-based computer vision applications, the commonly used lidar sensors, such as Velodyne HDL-64 used in the KITTI and Ford dataset, rotate at a rate of 10Hz, that is 0.1s per-scan. Therefore, the real-time performance here means that the processing time of each scanning data is less than 0.1s. An NVIDIA 1080 Ti GPU and an Intel Core i7 3.4GHz 4-core CPU are chose as our test platform. In test-time, the data batch-size of LO-Net is set to 1. Table 4 shows the average running times on Seq. 00 of KITTI dataset. The average processing time of our framework is about 80.1ms per-scan in total. Reasonably, most of our runtime is spent on the mapping procedure. Compared with most of traditional lidar odom-
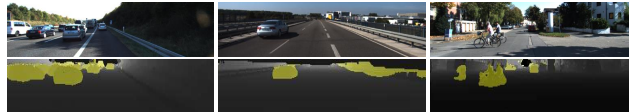

Figure 9. Sample visualizations of masks on range channel of the data matrix and its corresponding RGB images. The yellow pixels indicate the uncertain points in the surrounding environment for odometry estimation, such as points of moving cars, cyclists and others. The images are cropped and reshaped for better visualization.

Table 4. Average runtime on KITTI Seq. 00

Data preparation	Inference	Mapping	total
8.5ms on CPU	10.2ms on GPU	61.4ms on CPU	80.1ms

etry estimation methods, including the methods evaluated in Section 4.2, our map-based optimization is lightning fast since we consume a new representation of input data. Our approach enables real-time performance through a straightforward pipeline on a platform with GPU. For lower performance platforms, we can also speed up the processing through the parallelism of LO-Net and the mapping. Currently, some parts of our framework run on CPU, and we can implement them on GPU to further increase the speed.

5. Conclusions

We present a novel learning framework LO-Net to perform lidar odometry estimation. An efficient mapping module is coupled into the estimation framework to further improve the performance. Experiments on public benchmarks demonstrate the effectiveness of our framework over existing approaches.

There are still some challenges that need to be addressed: 1) The point clouds are encoded into data matrices to feed into the network. Direct processing of 3D point clouds could be more practical for 3D visual tasks. 2) Our current network is trained with ground truth data. This limits the application scenarios of the network. In our future work, we will investigate in more detail the geometry feature representation learned by the network. We also plan to incorporate recurrent units into this network to build temporal-related features. This may lead to an end-to-end framework without the need of costly collections of ground truth data.

Acknowledgment

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Noha Radwan Abhinav Valada and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *Proceedings Of The IEEE International Conference On Robotics And Automation (ICRA)*, May 2018.

[3] PJ Besl and ND McKay. A method for rgistration of 3-d shapes. *IEEE Transaction on Pattern Analisys and Machine Intelligence*, 14:239–256, 1992.

[4] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Mapnet: Geometry-aware learning of maps for camera localization. *arXiv preprint arXiv:1712.03342*, 2017.

[5] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3, 2017.

[7] Jean-Emmanuel Deschaud. Imls-slam: scan-to-model matching based on 3d data. *arXiv preprint arXiv:1802.08633*, 2018.

[8] Bertrand Douillard, A Quadros, Peter Morton, James Patrick Underwood, Mark De Deuge, S Hugosson, M Hallström, and Tim Bailey. Scan segments matching for pairwise 3d alignment. In *2012 IEEE International Conference on Robotics and Automation*, pages 3033–3040. IEEE, 2012.

[9] David F Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3d primitives for single image understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3392–3399, 2013.

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[12] W Shane Grant, Randolph C Voorhies, and Laurent Itti. Finding planes in lidar point clouds for real-time registration. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4347–4354. IEEE, 2013.

[13] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE, 2016.

[14] Stefan Holzer, Radu Bogdan Rusu, Michael Dixon, Suat Gedikli, and Nassir Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE, 2012.

[15] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[16] Alex Kendall, Roberto Cipolla, et al. Geometric loss functions for camera pose regression with deep learning. In *Proc. CVPR*, volume 3, page 8, 2017.

[17] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018.

[20] Frank Moosmann. *Interlacing self-localization, moving object tracking and mapping for 3d range sensors*, volume 24. KIT Scientific Publishing, 2013.

[21] Frank Moosmann and Christoph Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398. IEEE, 2011.

[22] Gaurav Pandey, James R McBride, and Ryan M Eustice. Ford campus vision and lidar data set. *The International Journal of Robotics Research*, 30(13):1543–1552, 2011.

[23] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. Fast registration based on noisy planes with unknown correspondences for 3-d mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010.

[24] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[26] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. Citeseer, 2009.

[27] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[28] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435, 2009.

[29] Jacopo Serafin and Giorgio Grisetti. Nicp: Dense normal based point cloud registration. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 742–749. IEEE, 2015.

[30] Jacopo Serafin, Edwin Olson, and Giorgio Grisetti. Fast and robust 3d feature extraction from sparse point clouds. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4105–4112. IEEE, 2016.

[31] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *Robotics and automation (icra), 2011 ieee international conference on*, pages 2601–2608. IEEE, 2011.

[32] Martin Velas, Michal Spanel, and Adam Herout. Collar line segments for fast odometry estimation from velodyne point clouds. In *ICRA*, pages 4486–4495, 2016.

[33] Martin Velas, Michal Spanel, Michal Hradis, and Adam Herout. Cnn for imu assisted odometry estimation using velodyne lidar. In *Autonomous Robot Systems and Competitions (ICARSC), 2018 IEEE International Conference on*, pages 71–77. IEEE, 2018.

[34] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.

[35] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[36] N. Yang, R. Wang, J. Stueckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision (ECCV)*, Sept. 2018. accepted as oral presentation, to appear, arXiv 1807.02570.

[37] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. Lego: Learning edge with geometry all at once by watching videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 225–234, 2018.

[38] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

[39] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

[40] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

[41] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[42] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.

[43] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.