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

ENCODE: a dEep poiNt Cloud ODometry nEtwork

Yihuan Zhang[1], Liang Wang[1], Chen Fu[2], Yifan Dai[1] and John M. Dolan[2]

Abstract— **Ego-motion estimation is a key requirement for the simultaneous localization and mapping (SLAM) problem. The traditional pipeline goes through feature extraction, feature matching and pose estimation, whose performance depends on the manually designed features. In this paper, we are motivated by the strong performance of deep learning methods in other computer vision and robotics tasks. We replace hand-crafted features with a neural network and directly estimate the relative pose between two adjacent scans from a LiDAR sensor using ENCODE: a dEep poiNt Cloud ODometry nEtwork. Firstly, a spherical projection of the input point cloud is performed to acquire a multi-channel vertex map. Then a multi-layer network backbone is applied to learn the abstracted features and a fully connected layer is adopted to estimate the 6-DoF ego-motion. Additionally, a map-to-map optimization module is applied to update the local poses and output a smooth map. Experiments on multiple datasets demonstrate that the proposed method achieves the best performance in comparison to state-of-the-art methods and is capable of providing accurate poses with low drift in various kinds of scenarios.**

I. INTRODUCTION

Ego-motion estimation is also known as odometry, and it is a vital task in SLAM for intelligent robots and self-driving cars. Accurate ego-motion is a fundamental requirement for navigation and path planning. Over the past few decades, much research has been performed trying to solve the ego-motion estimation problem in different scenarios, including camera-based and LiDAR-based methods. Unlike cameras, LiDAR can not only provide high-precision range measurements, but it also is insensitive to illumination, which makes it more suitable in SLAM than camera-based systems. Especially with the recent development of high-resolution LiDARs, such as the Velodyne HDL-64E, Ouster OS1-128, etc., it has become more feasible for LiDARs to capture the details in the 3D environment. Therefore, LiDAR-based odometry and mapping methods are the focus of this paper.

Typically, LiDAR odometry is estimated by finding the transformation between two adjacent point clouds. The Iterative Closest Point (ICP) method [1] is used to align two sets of points iteratively until the stopping criteria are satisfied. As ICP takes all points into account, its efficiency cannot be guaranteed. Feature-based matching methods were developed to avoid matching a full point cloud. LiDAR odometry and mapping (LOAM) [2], [3] is one of the most classic feature-based matching methods for low-drift and real-time state estimation and mapping. Nevertheless, the odometry estimation problem becomes unfriendly to feature-matching methods when the relative pose variation is large and no predicted alignment is given.

Recently, with the increase of computing power, many deep learning-based methods have been developed which can outperform traditional methods in many computer vision problems. LiDAR-based deep learning strategies [4], [5], [6], [7] were developed to estimate the 6 Degree-of-Freedom (DoF) pose with a Deep Neural Network (DNN) framework. The major challenge is handling a dense point cloud by feeding it into a deep neural network. A range image is the most used method to represent a point cloud so that some vision-based network can be applied easily. However, the geometry constraints can not be fully used and additional input like Inertial Measurement Unit (IMU) information or normal vectors are applied to improve the odometry estimation precision.

In this paper, ENCODE is proposed to replace the LiDAR Odometry (LO) module in the SLAM pipeline. Unlike most range image-based point cloud pre-processing methods, we transform the point cloud to a vertex map using spherical projection with geometry constraints. Then two adjacent multi-channel vertex maps are concatenated to estimate the odometry throughout a deep neural network backbone. By attaching a mapping module, the high-frequency odometry results are optimized and a low-frequency map can be acquired. The paper's main contributions are as follows:

- A neural network structure is designed to replace hand-craft features, which makes the odometry more adaptive to various scenarios.
- We concatenate the inputs of two adjacent scans as a multi-channel vertex map, which enhances the geometry constraints and requires less computing cost.
- Comprehensive experiments and ablation studies are carried out to evaluate the proposed method. Results show that ENCODE is capable of replacing the conventional feature-based odometry module.

This paper is organized as follows: Section II reviews the related works on both feature-based and deep learning-based LiDAR SLAM approaches. Section III details the proposed method for odometry estimation. Section IV gives experimental results and a comparison with existing works, followed by the conclusion in Section V.

II. RELATED WORKS

In this section, the related works on LiDAR odometry and SLAM are briefly reviewed, including feature-based methods and deep learning-based methods.

[1]Yihuan Zhang, Liang Wang and Yifan Dai are with the Intelligent Connected Vehicle Center, Tsinghua Automotive Research Institute, Suzhou, China, 215000, zhangyihuan,wliang,daiyifan@tsari.tsinghua.edu.cn

[2]Chen Fu and John M. Dolan are with the Robotic Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, chenfu@cs.cmu.edu, jmd@cs.cmu.edu

A. Feature-based methods

As ICP [1] uses all the points as the input feature, its computational cost is large. Thus, many ICP-variant methods have been proposed. A point-to-plane ICP method [8] was applied to match points to local planar patches. Generalized-ICP [9] introduced a plane-to-plane matching strategy to match corresponding point clouds using local surfaces. Considering the normal and curvature features of the point cloud, Normal-ICP [10] demonstrated better performance and robustness than previous ICP methods.

In order to get better correspondences between point clouds, many hand-crafted feature extraction methods have been proposed. Persistent Feature Histograms (PFH) [11] describes the curvature around a point by estimating surface normals between neighboring points. Fast Point Feature Histograms (FPFH) [12] is a faster version of PFH that can be implemented in real-time applications. A Kanade-Tomasi corner detector [13] was applied to extract general-purpose features, but there were many outliers, which limited the matching performance. A robust feature extraction method [14] was presented to enhance the stability of features. The key-points were extracted from lines and planes with the removal of ground points, and the estimation of transformations outperformed previous methods.

However, due to point cloud sparsity, it is difficult to balance the quality and number of the features. Over the past few years, LOAM [2], [3] has been considered the state-of-the-art odometry estimation and mapping method. The edge and surface features are extracted based on the smoothness of each point. The odometry module was used to estimate motion based on edge-line and surface-plane matching at a high frequency ($\sim$10Hz), and a mapping module was applied to register the point cloud to the local map via an optimization step. Similar to LOAM, a lightweight and ground-optimized LiDAR odometry and mapping (LeGO-LOAM) method [15], [16] was proposed for the ground vehicle mapping task. By applying ground plane extraction and point cloud segmentation, LeGO-LOAM filtered out unreliable features and showed great stability in areas covered with noisy objects. Recently, some IMU-based LiDAR SLAM methods were proposed to enhance the performance of odometry estimation and mapping [17], [18], [19], all these methods use a pipeline similar to LOAM's, which may encounter the same problem when the relative pose variation is large or no predicted alignment is given.

B. Deep learning-based methods

A recent survey [20] on deep learning for localization and mapping provides a guide for researchers interesting in the SLAM problem. In the visual odometry domain, deep learning-based methods have achieved promising results, including monocular visual odometry [21], stereo visual odometry [22] and unsupervised visual odometry [23], [24]. Due to the difference between camera and LiDAR, some deep learning-based methods were proposed for one or more modules in the SLAM pipeline. For point cloud registration, many recent methods [25], [26], [27], [28] have been proposed trying to detect more accurate correspondences from feature extraction. Following the direction of iterative optimization methods such as ICP, the methods [29], [30] trained the network directly using the transformation from two scans. Compared with point cloud registration, the odometry estimation problem is more challenging, since it requires much higher accuracy under noisy environments to minimize drifting.

A Convolutional Neural Network (CNN) [4] was applied to estimate odometry from LiDAR scan sequences. The original point cloud was transformed into a dense matrix with three channels, and the odometry estimation was formulated as a classification problem rather than a numerical regression problem. Furthermore, only 3-DoF translation was estimated. DeepPCO [5] and LO-Net [6] were designed to estimate 6-DoF odometry as an end-to-end numerical regression problem. A dual-branch architecture was used in DeepPCO to predict the translation and rotation separately. In LO-Net, the point cloud was encoded into a range image containing the intensity value and range value. In this paper, the 6-DoF ego-motion is estimated through a single neural network and the geometry constraints are enhanced by concatenating two adjacent scans as a multi-channel vertex map.

III. Proposed Method

In this section, the framework of the proposed method (see Fig. 1) is described in detail. The input point cloud is pre-processed and fed into ENCODE, which outputs a 6-DoF odometry estimate into the mapping module.

A. Data Pre-processing

A LiDAR sensor point cloud is typically represented by a set of sparse and unordered 3D points. As shown in Fig. 1, the original point cloud scan S with ring structures is represented in Cartesian coordinates. Each point $p = (x, y, z, \mu, \nu, \kappa)$ is projected onto a 2D vertex map (u, v) in spherical coordinates using the following equation:

$$\begin{aligned} u &= \frac{1}{2}[1 - \arctan(y, x)/\pi]N_w \\ v &= [1 - (\arcsin(z/d) + f_{up})/f]N_h \end{aligned} \tag{1}$$

where N_w and N_h are the width and height of the desired vertex map representation, and $f = f_{up} + f_{down}$ is the vertical field-of-view of the sensor. μ represents the remission/intensity value of the Velodyne and Ouster LiDAR sensors, ν and κ are the reflectivity and ambient value of Ouster LiDAR sensors. For each vertex, we build 5-dimensional features for the Velodyne LiDAR sensors and 7-dimensional features for the Ouster LiDAR sensors, including range $d = \sqrt{x^2 + y^2 + z^2}$, x, y, z, μ, ν and κ. Due to noise in the point cloud data, there may be more than one point at one vertex, and in that case, the closest point will be selected. By applying the projection, a multi-channel vertex map is prepared for the next part, unlike most existing deep learning-based methods [4], [5], [6], which separately feed the two adjacent point cloud data into the network backbone. Then the two outputs are concatenated before the odometry

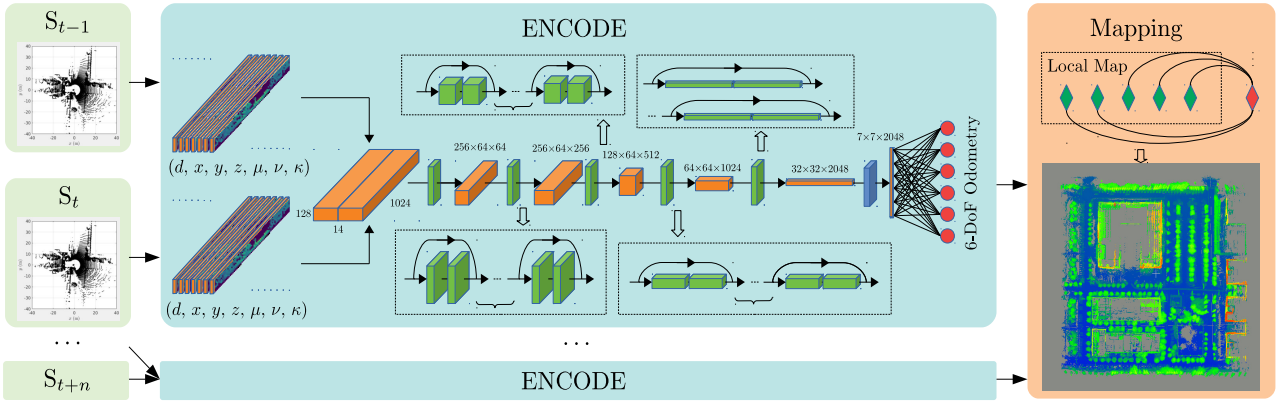


Fig. 1. The framework of this paper. S represents the LiDAR sensor scans with respect to time t. The pre-processing module consists of spherical projection and vertex map generation. A multi-layer backbone and fully connected head are included in the ENCODE module. A map-to-map optimization method is included in the mapping module.

head, and the spatial relationship between two inputs will be dropped. In this paper, we concatenate the vertex map from two adjacent point cloud data in order to retain the geometry constraint and reduce the computation cost at the same time.

B. ENCODE Architecture

The overview of the proposed ENCODE architecture is depicted in Fig. 1. The key idea is to jointly learn and estimate position and rotation vectors using a single neural network. The trainable layers are listed in Tab. I.

TABLE I
LAYERS OF THE NETWORK ARCHITECTURE.

	Operator	Stride	Filters	Size	Output
Backbone	Conv2D	2×2	64	7×3	512×64×64
	MaxPool	2×1	64	3×3	256×64×64
	Conv2D	1×1	256	1×1	256×64×256
	Conv2D	1×1	256	3×3	256×64×256
			×2		
	Conv2D	1×1	512	1×1	256×64×512
	Conv2D	2×1	512	3×3	128×64×512
			×2		
	Conv2D	1×1	1024	1×1	128×64×1024
	Conv2D	2×1	1024	3×3	64×64×1024
			×2		
	Conv2D	1×1	2048	1×1	64×64×2048
	Conv2D	2×2	2048	3×3	32×32×2048
			×2		
	AvgPool	4×4	2048	7×7	7×7×2048
Head	FC	–	–	–	6

The main backbone in ENCODE is designed based on ResNet [31]. ResNet is an artificial neural network which uses skip connections, or shortcuts to jump over the layers. Such a structure can avoid the problem of vanishing gradients when the network is deep. The original ResNet was designed for image recognition and in this paper, it is modified to fit the particular input type and form factor. The plain network consists of six blocks. The first layer has a 2D convolutional network and a max pool layer, which downsample the input into a relatively small size. There are four residual blocks in the middle, each block containing two sets of convolutional layers and a shortcut connecting the input and output. Finally, an average pool layer is used to smooth the features.

The head in this paper is designed to estimate the 6-DoF motion between two scans. Based on the output of the backbone, we flatten a $7 \times 7 \times 2048$ tensor into a 1D tensor of size 100,352. Then a single fully-connected layer is used to output the estimated odometry, consisting of relative translation and relative rotation. The odometry is denoted as $\hat{X}$, containing six elements, and the estimated odometry results are used to define the training loss as follows:

$$\mathcal{L}(S_{t-1}, S_t) = \begin{cases} \frac{1}{2}(X - \hat{X}) & \text{if } |X - \hat{X}| \leq \delta \\ \delta|X - \hat{X}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \quad (2)$$

where X is the ground truth of the odometry and the Huber loss is used in this paper because it is less sensitive to outliers in data than the squared error loss. In addition, it is differentiable at zero and the hyperparameter δ can be tuned to change the threshold of the absolute error form.

C. Mapping

After the odometry information is inferred through ENCODE, a mapping module is applied to improve accuracy with respect to the original, generalized LOAM framework [2], [3]. The details of the mapping module are introduced below.

The mapping module runs at a lower frequency than the odometry estimation module. We build up a sweep containing n scans as $(\hat{X}_{t-n+1}, \cdots, \hat{X}_t, \mathcal{F}_{t-n+1}, \cdots, \mathcal{F}_t)$, where $\mathcal{F}$ consists of the edge feature $\mathcal{F}^e$ and the surface feature $\mathcal{F}^s$. The features are projected onto a sub-map in each sweep, then a map-to-map matching method is used to refine the final pose of the current sweep. The GaussNewton method is used to solve the optimization problem defined as:

$$\min_{\mathcal{T}_{t,t+1}} \left\{ \sum_{p_{t,i} \in \mathcal{F}_t^e} d_{e_i} + \sum_{p_{t,i} \in \mathcal{F}_t^s} d_{s_i} \right\} \quad (3)$$

$$d_{e_i} = \frac{\left|(p^e_{t+1,i} - p^e_{t,j}) \times (p^e_{t+1,i} - p^e_{t,k})\right|}{\left|p^e_{t,j} - p^e_{t,k}\right|} \tag{4}$$

$$d_{s_i} = \frac{\left|(p^s_{t+1,i} - p^s_{t,j}) \times (p^s_{t,j} - p^s_{t,k}) \times (p^s_{t,j} - p^s_{t,l})\right|}{\left|(p^s_{t,j} - p^s_{t,k}) \times (p^s_{t,j} - p^s_{t,l})\right|} \tag{5}$$

where d_{e_i} represents the distance from edge point $p^e_{t+1,i}$ to the line $(p^e_{t,j}, p^e_{t,k})$, and d_{s_i} denotes the distance from surface point $p^s_{t+1,i}$ to the surface $(p^s_{t,j}, p^s_{t,k}, p^s_{t,l})$. The relative transformation $\mathcal{T}_{t,t+1}$ is optimized to update the current pose.

IV. Experiments

In this section, the proposed method is evaluated via qualitative and quantitative comparisons using the publicly available dataset KITTI [32] and a dataset called TSARI recorded by the self-driving car platform. The implementation code of this paper is open-sourced and the TSARI dataset will be publicly available soon[1].

A. Datasets

1) KITTI: The KITTI odometry dataset [32], [33] consists of 22 independent sequences with gray-scale and color camera images, point clouds captured by a Velodyne HDL-64E LiDAR sensor, and calibration parameters. 23,201 scans from sequences 00-10 are provided with ground truth pose obtained from the GPS/IMU readings. For 20,351 scans from sequences 11-21, ground truth is available only indirectly, by submitting to the KITTI benchmarking facility. 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, and the driving speed is up to 90 km/h.

2) TSARI: The TSARI dataset was recorded by the self-driving car with an Ouster OS1-128 LiDAR sensor mounted on the top. The point cloud data are synchronized with GPS/IMU data. There are three main scenarios included in this dataset: urban, park and highway. The detailed description is shown in Tab. II. Note that the duration, length and speed are average numbers in each scenario. In addition, the data are not recorded in rush hour, so there is no traffic jam in any of the sequences.

TABLE II
TSARI DATASET DESCRIPTION

	Sequences	Duration(s)	Length(m)	Speed(km/h)
Urban	00-05	598.27	5418.09	32.08
Park	06-13	505.28	1684.51	11.74
Highway	14-15	362.35	5605.53	55.87

[1]The code in this paper is open-sourced for public testing at `https://github.com/stzyhian/ENCODE.git`

B. Implementation details

In the KITTI dataset, we set $N_w = 2048$ and $N_h = 64$ in Eqn. 1, and the channel in the vertex map is set to be 5, including range, x, y, z and intensity. After concatenating two adjacent point clouds, the final input is $2048 \times 64 \times 10$. In the TSARI dataset, $N_w = 1024$ and $N_h = 128$, and the input is $1024 \times 128 \times 14$. The network settings are the same in both datasets. We choose the RAdam [34] solver with default parameters for optimization. The initial learning rate is 1e-4 and the cosine annealing scheduler is used to dynamically adjust the learning rate. The weight decay in loss is set to be 1e-4, the dropout ratio is 0.1, and the batch size is 20. The hyperparameter δ in the loss function is set to 1. The network is trained on four NVIDIA 2080 Ti GPUs. The parameters in the mapping module are the same as those in LOAM and the mapping process is performed every 5 frames.

C. Odometry Results

Several baselines are compared in this paper, including the classical ICP, GICP [9], LOAM [3], a CNN-based LiDAR odometry method [4] and LO-Net [6]. The ICP method is implemented using the Point Cloud Library [35] and the fast-gicp [36] method is used to obtain the GICP results. LOAM achieves the best performance on the KITTI odometry benchmark. The latest code is not open-sourced, so we only use the original code to evaluate the odometry results. The two kinds of deep learning-based methods, CNN and LO-Net, are not open-sourced yet, thus the results from their paper are directly copied for comparison.

1) KITTI dataset: For fair comparison, we follow the setting of LO-Net, which uses sequences 00-06 for training and sequences 07-10 for testing. Tab. III shows the evaluation results of the methods on KITTI the datasets. We use the KITTI odometry evaluation metrics [33], which consist of the average relative translation and rotation errors in $\%$ and $^\circ/100m$ for segments of 100 to 800m.

We compare ENCODE to LO-Net and CNN as deep learning-based methods which only estimate end-to-end odometry without mapping. It can be seen that LO-Net performs slightly better than ENCODE in training sequences, but much worse on the testing sequences, which may be caused by overfitting. The CNN-based method's performance is worse than that of ENCODE and LO-Net, which may be caused by the rotation assumption, which limits the range of rotations. In a highway scenario like sequence 01, it is difficult for methods like ICP or GICP to match the longitudinal movements without high-level feature extraction.

In addition, we compare ENCODE with mapping to the state-of-the-art method LOAM and LO-Net with mapping. In most sequences, the proposed method has the best performance. As we are using a similar mapping method to that of LOAM, it is clear that with a better odometry estimate, the mapping result can be more accurate. Fig. 2 shows the trajectories from different methods in KITTI sequence 08 with ground truth. The drift of ENCODE is large at the end of the sequence, which is inevitable because the odometry only estimates the relative movements of two adjacent point

TABLE III

ODOMETRY RESULTS ON KITTI DATASETS.

Dataset	ICP		GICP [9]		LOAM [3]		CNN [4]		LO-Net [6]		ENCODE		LO-Net+M[1]		ENCODE+M[1]	
	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}
KITTI-00†	3.80	1.73	1.29	0.64	0.78	0.53	3.02	–	1.47	0.72	0.73	0.31	0.78	0.42	**0.67**	**0.29**
KITTI-01†	13.53	2.58	4.39	0.91	1.43	0.55	4.44	–	**1.36**	0.47	1.45	0.52	1.42	**0.40**	1.51	0.44
KITTI-02†	9.00	2.74	2.53	0.77	**0.92**	0.55	3.42	–	1.52	0.71	1.13	0.42	1.01	0.45	**0.95**	**0.37**
KITTI-03†	2.72	1.63	1.68	1.08	0.86	0.65	4.94	–	1.03	0.66	1.15	0.76	**0.73**	0.59	0.90	**0.48**
KITTI-04†	2.96	2.58	3.76	1.07	0.71	0.50	1.77	–	0.51	0.65	0.97	0.41	**0.56**	0.54	0.74	**0.40**
KITTI-05†	2.29	1.08	1.02	0.54	0.57	0.38	2.35	–	1.04	0.69	1.27	0.61	0.62	0.35	**0.50**	**0.24**
KITTI-06†	1.77	1.00	0.92	0.46	0.65	0.39	1.88	–	0.71	0.50	1.14	0.42	**0.55**	0.33	0.56	**0.27**
KITTI-07▽	1.55	1.42	0.64	0.45	0.63	0.50	1.77	–	1.70	0.89	1.15	0.74	0.56	0.45	**0.38**	**0.22**
KITTI-08▽	4.42	2.14	1.58	0.75	1.12	0.44	2.89	–	2.12	0.77	1.18	0.46	1.08	0.43	**0.80**	**0.31**
KITTI-09▽	3.95	1.71	1.97	0.77	0.77	0.48	4.94	–	1.37	0.58	0.93	0.36	0.77	0.38	**0.70**	**0.35**
KITTI-10▽	6.13	2.60	1.31	0.62	0.79	0.57	3.27	–	1.80	0.93	0.83	0.34	0.92	0.41	**0.75**	**0.32**
KITTI-Mean†	5.15	1.91	2.23	0.78	0.85	0.51	3.12	–	1.09	0.63	1.12	0.49	**0.81**	0.44	0.83	**0.36**
KITTI-Mean▽	4.01	1.97	1.38	0.65	0.83	0.50	3.22	–	1.75	0.79	1.02	0.48	0.83	0.42	**0.66**	**0.30**

[1]: M means mapping module. Both LO-Net and ENCODE are evaluated with and without mapping module attaching.
t_{rel}: average translational RMSE(%) on length of 100-800m. r_{rel}: average rotational RMSE($^\circ/100m$) on length of 100-800m.
†: the sequences for training. ▽: the sequences for testing. Symbols remain the same below.

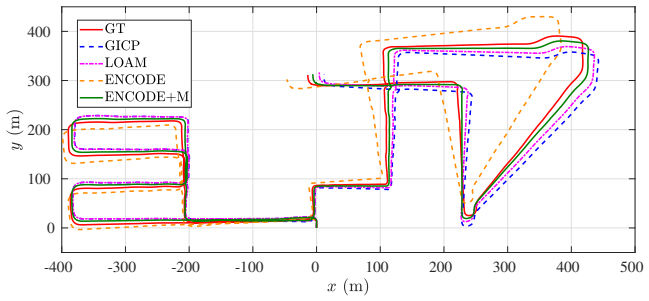
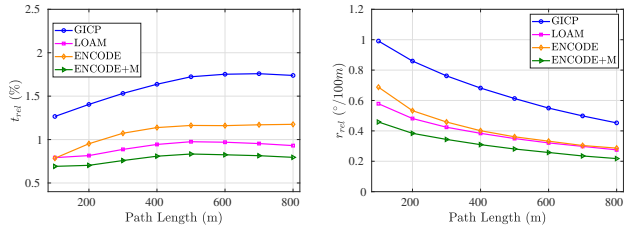


Fig. 2. Trajectories of KITTI-08 with ground truth(GT). The ENCODE with Mapping method produces the most accurate result.



(a) Average translation error (b) Average rotation error

Fig. 3. Evaluation results on the KITTI 00-10 sequences. The average errors of translation and rotation with respect to path length intervals.

clouds. The average evaluation errors on KITTI sequences 00-10 are shown in Fig. 3. ENCODE with mapping achieves the best performance among other baselines.

For KITTI sequences 11-21, there is no ground truth provided, so we submit the results to the KITTI benchmark to fairly compare with other methods. Tab. IV shows the KITTI benchmark submission results. As LO-Net has not been submitted to KITTI, we are unable to compare with its results, and only submitted deep learning-based methods are listed for comparison. Our performance on testing sequences is the best. However, the average translation error is higher than the result of sequences 00-10. The main reason is that there are more highway scenarios in sequences 11-21, which may cause large drift. The estimation errors in sequence 11 and 15 are smaller than average because these are urban scenarios.

TABLE IV

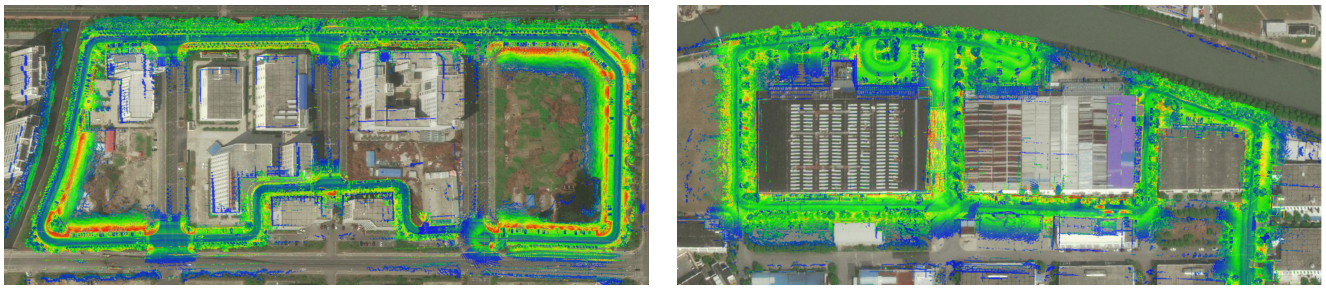
SUBMISSIONS ON KITTI.

	t_{rel}^{avg}	r_{rel}^{avg}	t_{rel}^{11}	r_{rel}^{11}	t_{rel}^{15}	r_{rel}^{15}
DeepCLR [37]	4.19	0.87	1.79	0.84	1.67	0.57
BLF [4]	3.49	1.28	4.13	1.32	2.20	1.40
ENCODE	**1.23**	**0.38**	**0.80**	**0.37**	**0.81**	**0.30**

2) TSARI dataset: We choose Urban-00 to 03, Park-06 to 11 and Highway-14 for training and the rest for testing. The results are shown in Tab. V, and the evaluation metrics are the same as for the KITTI dataset. The GICP method is unable to provide odometry in highway scenarios because the point clouds are too similar. It can be seen that the park scenarios are easier than urban scenarios because parks are relatively small and have more distinguishable features. The proposed method is slightly worse than LOAM in park scenarios and outperforms both compared methods in urban and highway scenarios. Fig. 4 shows the mapping results of TSARI Park-08 and Park-12; it can be seen that the mapping result fits the satellite image.

D. Ablation Study

In order to explore the impact of various components of the proposed ENCODE, the following ablation experiments were conducted including the input representation, network backbone and head. The purpose is to check whether the design of each component is better than other structures. The KITTI dataset is used in the ablation study.



(a) TSARI Park-08 (b) TSARI Park-12

Fig. 4. TSARI Park-08 and Park-12 mapping results. The point cloud is colored according to the intensity value and is overlaid on the satellite image.

TABLE V
ODOMETRY RESULTS ON TSARI DATASETS.

Dataset	GICP [9]		LOAM [3]		ENCODE+M	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Park†	2.43	1.23	0.64	0.37	**0.59**	**0.37**
Urban†	4.34	1.01	1.93	0.65	**1.69**	**0.56**
Highway†	67.6	0.90	2.13	0.53	**1.60**	**0.47**
Park$^{\triangledown}$	1.56	1.04	**0.62**	**0.37**	0.69	0.41
Urban$^{\triangledown}$	2.09	0.65	1.35	0.49	**1.24**	**0.44**
Highway$^{\triangledown}$	63.1	1.31	2.40	0.47	**1.63**	**0.46**

1) Input representation: Tab. VI shows the results of different input representations. 'Sepa.' and 'Conc.' respectively denote separation and concatenation of the input data of two adjacent scans. It can be seen that with the concatenation of two input scans, the inference efficiency is improved. In addition, integration of x, y, z information into the vertex map causes the odometry estimate results to be more accurate.

TABLE VI
ABLATION STUDY ON USAGE OF INPUT MODALITIES.

	$t^{\dagger}_{rel}$	$r^{\dagger}_{rel}$	$t^{\triangledown}_{rel}$	$r^{\triangledown}_{rel}$	Time(ms)
(d, μ)-Sepa.	1.40	0.64	1.78	0.83	41.8
(d, μ, x, y, z)-Sepa.	1.36	0.60	1.55	0.76	43.6
(d, μ)-Conc.	**1.10**	0.56	1.30	0.55	**20.1**
(d, μ, x, y, z)-Conc.	1.12	**0.49**	**1.02**	**0.48**	21.3

2) Network backbone: Tab. VII demonstrates the results of different network backbones, noting that all four backbones are slightly modified to fit the input. All the parameters are set to be the same and the learning epoch is set to be 200. DarkNet-53 [38] was first proposed to solve the object detection problem in real time and achieved a remarkable performance. However, the performance in odometry estimation is not outstanding, and the average processing rate is 20 fps, which is not suitable for real-time tasks. SqueezeNet [39] is a smaller neural network with fewer parameters that can more easily fit into computer memory and can more easily be transmitted over a computer. It is able to provide a 50-fps processing rate and has a slightly worse performance than ENCODE. ResNet [31] is the structure we used in ENCODE backbone. ResNet-50 is deeper than ResNet-18 and has more parameters, requiring more computation cost. In comparison to ResNet-50, the estimation error is similar in the training dataset and smaller in the testing dataset, which may be caused by the over-fitting in a deeper network. In addition, the inference time is half that of ResNet-50, and we are able to estimate the odometry in real time.

TABLE VII
ABLATION STUDY ON USAGE OF BACKBONE MODALITIES.

	$t^{\dagger}_{rel}$	$r^{\dagger}_{rel}$	$t^{\triangledown}_{rel}$	$r^{\triangledown}_{rel}$	Time(ms)
DarkNet-53 [38]	1.21	0.63	1.29	0.61	51.2
SqueezeNet [39]	1.19	0.64	1.19	0.69	**20.5**
ResNet-50 [31]	1.14	0.60	1.25	0.59	43.7
ENCODE	**1.12**	**0.49**	**1.02**	**0.48**	21.3

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an end-to-end deep neural network named ENCODE for the point cloud odometry estimation task. Two adjacent point clouds are projected and concatenated into one vertex map, then a multi-layer backbone is designed to extract the abstract features and a fully connected head is adopted to estimate the odometry. A map-to-map optimization method is proposed to refine the odometry result and generate a smooth map. The approach shows good performance on multiple datasets and effectiveness over existing methods.

However, there are still many challenges that need to be addressed as our future work. Firstly, directly encoding the point cloud in 3D format could retain the original geometry information, which may be more practical for odometry estimation and other 3D visual tasks. Secondly, we are planning to expand ENCODE framework by integrating a mapping module so that we do not need to design any hand-crafted features. Lastly, during this work, we discovered that when it comes to a large scenario, odometry drift will be large and loop-closure is necessary to optimize the map.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under Contract 2018YFB0105000, the Natural Science Foundation of Jiangsu Province under Contract BK20200271 and Suzhou Science and Technology Project under Contract SYG202014.

REFERENCES

[1] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[2] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[3] ——, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[4] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for imu assisted odometry estimation using velodyne lidar," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 71–77.

[5] W. Wang, M. R. U. Saputra, P. Zhao, P. Gusmao, B. Yang, C. Chen, A. Markham, and N. Trigoni, "Deeppco: End-to-end point cloud odometry through deep parallel neural network," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3248–3254.

[6] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "Lo-net: Deep real-time lidar odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8473–8482.

[7] M. Valente, C. Joly, and A. de La Fortelle, "Deep sensor fusion for real-time odometry estimation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6679–6685.

[8] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[10] J. Serafin and G. Grisetti, "Nicp: Dense normal based point cloud registration," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 742–749.

[11] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2008, pp. 3384–3391.

[12] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.

[13] Y. Li and E. B. Olson, "Structure tensors for general purpose lidar feature extraction," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1869–1874.

[14] J. Serafin, E. Olson, and G. Grisetti, "Fast and robust 3d feature extraction from sparse point clouds," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4105–4112.

[15] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[16] T. Shan, J. Wang, B. Englot, and K. Doherty, "Bayesian generalized kernel inference for terrain traversability mapping," in *Conference on Robot Learning*, 2018, pp. 829–838.

[17] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3144–3150.

[18] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A Lidar-Inertial state estimator for robust and efficient navigation," in *2020 International Conference on Robotics and Automation (ICRA)*, 2020.

[19] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," 2020.

[20] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," *arXiv preprint arXiv:2006.12567*, 2020.

[21] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[22] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 817–833.

[23] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[24] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. 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*, 2018, pp. 340–349.

[25] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *European Conference on Computer Vision*. Springer, 2018, pp. 630–646.

[26] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "Deepvcp: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 12–21.

[27] H. Deng, T. Birdal, and S. Ilic, "3d local features for direct pairwise registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3244–3253.

[28] G. D. Pais, S. Ramalingam, V. M. Govindu, J. C. Nascimento, R. Chellappa, and P. Miraldo, "3dregnet: A deep neural network for 3d point registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7193–7203.

[29] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3523–3532.

[30] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7163–7172.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[34] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proceedings of the Eighth International Conference on Learning Representations (ICLR)*, April 2020.

[35] R. B. Rusu and S. Cousins, "3d is here: Point Cloud Library (PCL)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[36] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3d point cloud registration," EasyChair, Tech. Rep., 2020.

[37] M. Horn, N. Engel, V. Belagiannis, M. Buchholz, and K. Dietmayer, "Deepclr: Correspondence-less architecture for deep end-to-end point cloud registration," *arXiv preprint arXiv:2007.11255*, 2020.

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

[39] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.