DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network

Wei Wang, Muhamad Risqi U. Saputra, Peijun Zhao, Pedro Gusmao,
Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni

Abstract— Odometry is of key importance for localization in the absence of a map. There is considerable work in the area of visual odometry (VO), and recent advances in deep learning have brought novel approaches to VO, which directly learn salient features from raw images. These learning-based approaches have led to more accurate and robust VO systems. However, they have not been well applied to point cloud data yet. In this work, we investigate how to exploit deep learning to estimate point cloud odometry (PCO), which may serve as a critical component in point cloud-based downstream tasks or learning-based systems. Specifically, we propose a novel end-to-end deep parallel neural network called DeepPCO, which can estimate the 6-DOF poses using consecutive point clouds. It consists of two parallel sub-networks to estimate 3-D translation and orientation respectively rather than a single neural network. We validate our approach on KITTI Visual Odometry/SLAM benchmark dataset with different baselines. Experiments demonstrate that the proposed approach achieves good performance in terms of pose accuracy.

I. INTRODUCTION

Visual odometry (VO) estimation is one of the most fundamental research tasks in the field of computer vision and robotics. It incrementally estimates an agent's relative pose by examining changes in projected geometrical features captured from a monocular camera. Conventional visual odometry relies on feature extraction and matching which has been well studied for a number of decades. However, recent advances in deep learning bring another paradigm to VO by directly inferring the 6 Degree-of-Freedom (6-DoF) camera poses through end-to-end learning [18]. This leads to several advantages in terms of not requiring hand-crafted features and being able to learn directly from large datasets. Deep learning-based VO approaches achieve very impressive results compared to conventional geometry-based approaches in some benchmark datasets [20].

Point cloud data, such as that generated by a Lidar or Depth Camera, can provide richer information about the 3-D structure of the environment. This intuitively provides a better perspective on how the sensor moves over time. Conventional point-cloud based odometry relies on geometric approaches and has shown excellent performance [23]. However, these techniques suffer from issues of robustness such as being sensitive to outliers and having low adaptability to different environments [2]. Scan matching, which is a fundamental task in point-cloud based odometry, is prone to outliers introduced by hardware failure, agent vibration

*The authors are with the Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom. {firstname.lastname}@cs.ox.ac.uk
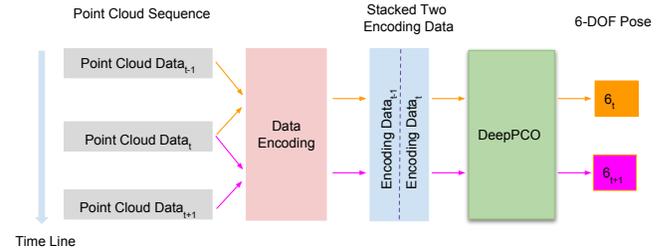
Fig. 1: System overview of the proposed DeepPCO framework. A sequential point cloud data stream is sent to the system. Two consecutive point cloud data are sent in parallel to the data encoding module, which implements the 2-D panoramic projection method. Projected depth images are then stacked to feed to DeepPCO neural network for predicting 6-DOF pose. These relative transformations are further combined to form a moving trajectory.

when collecting data, or unexpected sensor movements. Moreover, when the environment is featureless or objects within the environment are deformable, scan matching may also fail. These systems usually need other sensors such as the Inertial Measurement Unit (IMU) to compensate for these failures [24]. Another complexity is that operators need to perform manual hand-engineering to fine-tune the large number of model parameters.

Motivated by the success of deep learning as applied to VO, we consider whether similar successes can be achieved in estimating odometry from point cloud data, obviating the need for manually engineering features and model parameters, which we define as the Point Cloud Odometry (PCO) task. In particular, deep learning may eliminate the problem of noisy scan matching, potentially improving robustness. Furthermore, solving point clouds 6-DOF pose is significant for point cloud-related tasks in the realm of 3-D vision. Additionally, it is a natural fit to integrate learning-based odometry components into learning-based systems or SLAM. Nevertheless, a challenging problem for using point clouds is that it is unordered, which poses issues for convolutional kernels which inherently assume a structured input.

In this paper, we investigate two key issues which impact the performance of PCO using end-to-end deep learning. The first is the representation (encoding) of the point cloud itself. This ranges from the raw point cloud to various approaches to projecting it to a 2-D scene. The second issue comes from the choice of the model architecture itself. In particular,

we consider estimating the 6-DOF pose (translation and orientation) in a single regression network, or splitting the translation and orientation estimation tasks and regressing them in two sub-networks.

Based on these investigations, we propose a novel approach, termed DeepPCO, which combines the use of 2-D panoramic depth projections with two sub-networks to achieve accurate odometry, as shown in Fig.1. DeepPCO eliminates the intermediate modules (e.g. scan matching, geometric estimation) of a classical pipeline. We compare against various baselines using point cloud data from the KITTI Vision Benchmark Suite [7] which were collected using a 360° Velodyne laser scanner.

Our main contributions are as follows:

- We demonstrate that point cloud odometry problem can be effectively solved in an end-to-end fashion, and our proposed architecture outperforms existing learning-based approaches by a significant margin, and received comparable performance to conventional ones.
- We adopt a dual-branch architecture to infer 3-D translation and orientation separately instead of a single network.
- Comprehensive experiments and ablation studies have been done to evaluate our proposed method. Results show that DeepPCO achieves good performance with respect to different kinds of neural network architectures.

II. RELATED WORK

In this section, we review deep learning for odometry tasks, especially Visual Odometry (VO), followed by the introduction of geometric and learning-based approaches for point cloud odometry. Previous works mainly focus on point cloud data generated by Lidar sensors, so we call this kind of work as Lidar Odometry (LO).

A. Deep Learning for Visual Odometry

Instead of manually estimating the geometry of the scenes, learning-based systems automatically learn the feature correspondences and relationships between images. These types of systems are usually trained in an end-to-end manner and save great efforts in engineering compared to the classic ones. Wang et al. [20] proposed an end-to-end framework for monocular VO consisting of a CNN branch built upon FlowNet [6], followed by a Recurrent Convolutional Neural Network structure (RCNN). Features are first extracted for two consecutive images through CNN and then forwarded to LSTM. Clark et al. [4] treated visual-inertial odometry as a sequence-to-sequence problem and made use of LSTM to embed IMU information to predict pose. Chen et al. [3] proposed an selective sensor fusion approach to solve visual-inertial odometry task. Constante et al. [5] presented a novel motion estimation network called LS-VO, which is based on Auto-Encoder (AE) scheme to find a non-linear representation of the Optical Flow (OF) manifold. [15] used raw optical flow images as inputs and proposed a VO system called Flowdometry, whose architecture consists of CNN. Although

the translation and rotation error evaluation does not excel the state-of-the-art approaches, due to careful engineering design, it achieved 23.796x speedup over existing learning-based VO. Gomez-Ojeda et al [8] developed a learning-based image enhancement approach to solve the robustness problem in VO. The network architecture consists of CNN and LSTM, and the LSTM was proved to help reduce noises by incorporating the temporal information from past sequences. Li et al. [14] presented UnDeepVO to estimate the 6-DOF poses of a monocular camera and the depth of its view. Stereo image pairs were harnessed for recovering absolute scale and loss function was defined on spatial and temporal dense information. Yang et al. [22] discussed a novel approach called Deep Virtual Stereo Odometry (DVSO), which first predicts depth using monocular images, and then incorporate them into Direct Sparse Odometry (DSO) as direct virtual stereo measurements. All the mentioned works take 2-D inputs to their neural networks.

B. Geometry and Deep Learning for Lidar Odometry

Classical geometry-based LO usually employs variations of Iterative Closest Point (ICP) [1] for scan matching. [19] introduced a probabilistic framework combining ICP and 'point-to-plane' algorithms to model locally planar surface structure from both scans, which can be applied to the registration of frame-to-frame scans. The state-of-the-art LO system, named LOAM, was proposed by [23], which consists of feature point extraction, feature correspondence calculation, motion estimation, and mapping components. The key idea is to optimize a large number of variables via two algorithms. One algorithm employs low fidelity to estimate velocity and the other one performs very low frequency for fine matching and point cloud registration. These two stages can be concluded as fast scan-to-scan and precise scan-to-map. Nicolai et al. [16] proposed a two-stream CNN architecture for frame-to-frame point cloud odometry estimation. They pre-processed point cloud data into 2-D images as inputs for their neural network. They collected point cloud data using the VLP-16 Lidar sensor mounted on Turtlebot. Although their experimental results are not superb, they demonstrate that it is possible to apply deep learning to LO task.

III. APPROACHES FOR POINT CLOUD ENCODING

Convolutional neural networks require highly structured data as inputs whereas point cloud data is unordered and irregularly sampled. In this section, we will discuss different point cloud encoding approaches and compare their relative merits for the task of point cloud odometry. An example of point-cloud data is shown in Fig 2, as obtained by a car-mounted LIDAR. Note the irregular density and gaps due to obstructions.

A. 2-D Encoding of Point Clouds

A straightforward approach to encode point clouds for odometry is to transform into a 2-D depth image. The projected depth image can be top-view, front-view, or
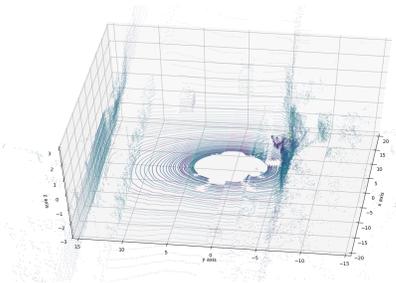
Fig. 2: A sample point cloud from KITTI dataset.

(a) Original Point Cloud (b) Panoramic Depth Image

(c) PointNet 3-D Encoding (d) PointGrid 3-D Encoding

Fig. 3: Visualization of various point cloud encoding approaches for deep learning.

panoramic-view. Among all of our experiments, we find that panoramic-view projection performs the best among all developed models with different parameter settings. Therefore, here we discuss panoramic-view projection equations that we have adopted in our work. Specifically, Li et al. [13] proposed an approach to project point cloud into panoramic depth image by the following equations:

$$\theta = \arctan 2(y, x) \tag{1}$$
$$\phi = \arcsin(z/\sqrt{x^2 + y^2 + z^2}) \tag{2}$$
$$r = \lfloor \theta/\Delta\theta \rfloor \tag{3}$$
$$c = \lfloor \phi/\Delta\phi \rfloor \tag{4}$$

where (x,y,z) are 3-D point coordinates, θ is azimuth angle, ϕ is elevation angle, (r, c) is the 2-D image position of 3-D point projection, $\Delta\theta$ is average horizontal angle resolution and $\Delta\phi$ is vertical angle resolution. For a detailed explanation, we refer readers to the original paper. In our work, we used the same equations above, and then normalized depth values to the range [0, 255]. Points closer to the sensor are assigned higher values. Indeed, the inverse normalization has been successfully applied to many 2-D vision tasks.

B. 3-D Encoding of Point Clouds

3-D point cloud data can also be discretized as a 3-D voxel grid and has been employed in VoxNet. Nevertheless, the voxelization of point clouds is impractical. For example, each point cloud in KITTI [7] contains approximately 120,000 points, which would require large memory size and computation resources to maintain the high resolution of the 3-D grid. Even adopting a sampling strategy to reduce the number of points, voxelization of point clouds is challenging for volumetric CNNs. Sampling would also affect the accuracy of our odometry task. Other than the voxelization approach, [17] proposed PointNet, a straightforward approach that treats every single point as a 3-dimensional vector (x, y, z) for 2-D CNN architecture. Thus, the input points are encoded as $n \times 3$, and this approach eliminates the use of 3-D CNN architecture which greatly speeds up the training process. However, their task and other following works mainly deal with classification or segmentation [21]. Moreover, the introduced transformation sub-network in the
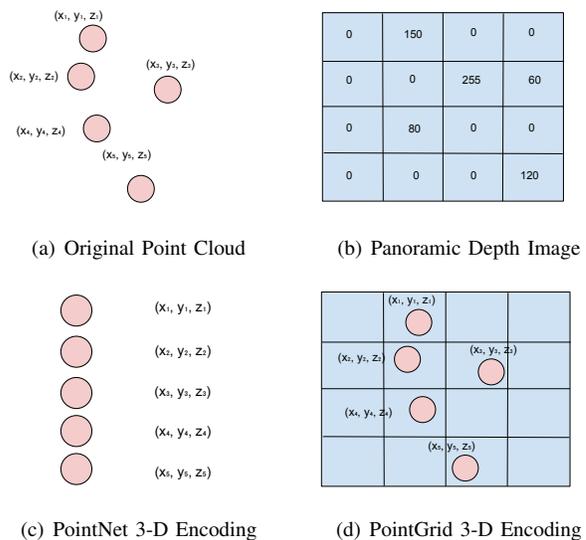
PointNet alters the translation that we need to accurately predict in our PCO task. Therefore, we reuse the PointNet architecture except we eliminate the T-Net in our experiment. More recently, [12] proposed a 3-D CNN architecture named PointGrid to integrate the benefits of point and grid for better representation of the local geometry shape given the limits of PointNet's ability to capture contextual neighborhood structure.

A graphical representation of all point cloud encoding approaches for deep learning in this work have been summarized in Fig.3. We used 2-D encoding for our proposed neural network architecture, termed DeepPCO, in which the point cloud data is projected to panoramic depth images. In the experimental results, we compare its performance with the PointNet and PointGrid 3-D encoding approaches.

IV. DEEPPCO ARCHITECTURE

Our DeepPCO is composed of two sub-networks, a Translation Sub-Network and FlowNet Orientation Sub-Network, both of which form a deep parallel neural network architecture. The entire DeepPCO architecture is illustrated in Fig. 4.

A. End-to-End Network Architecture

Our key idea is that instead of jointly learning and predicting position and rotation vectors using a single neural network as adopted by existing work, we design two separate neural networks - one is adept at predicting translation while the other specializes in inferring rotation. As will be shown in the experimental results, this approach leads to superior overall performance. The inputs of DeepPCO are two consecutive 2-D panoramic-view projection images which are stacked together. The Translation Sub-Network is responsible for the prediction of translation between two point clouds, while the FlowNet Sub-Network infers the rotation between
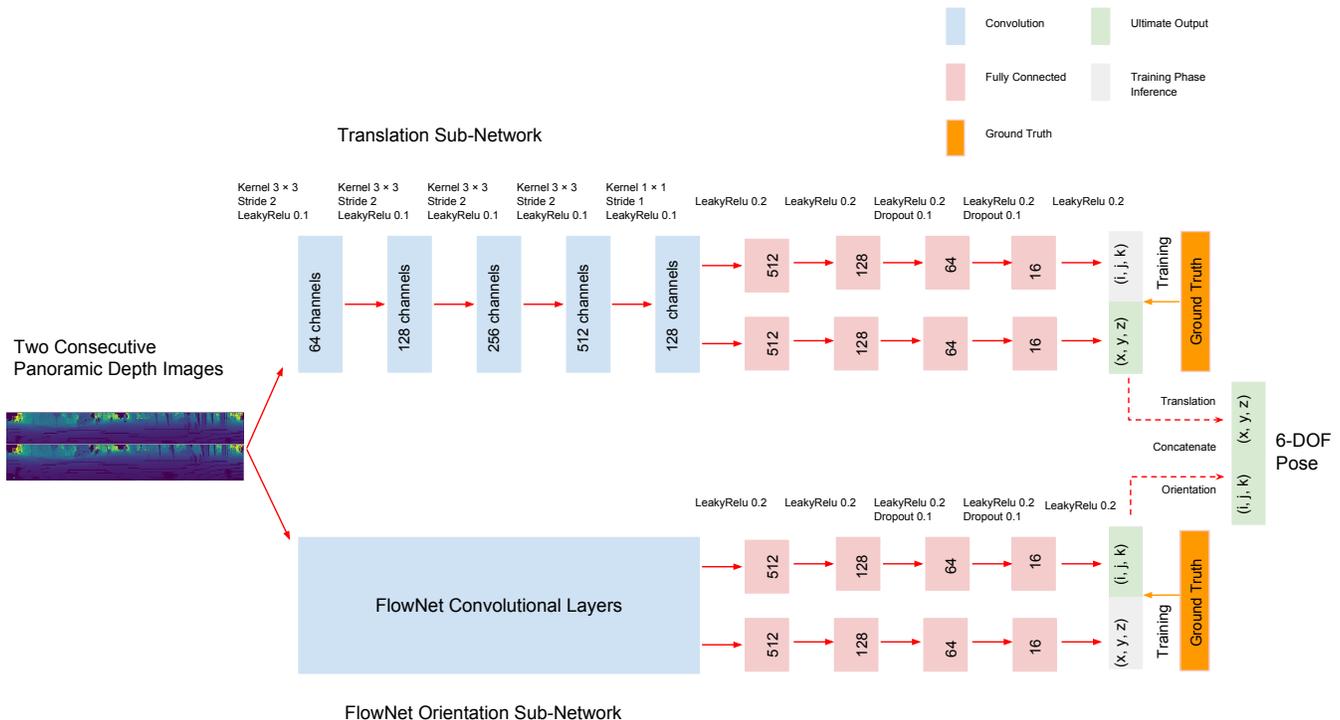
Fig. 4: The architecture of DeepPCO. Two consecutive point clouds are encoded into panoramic depth images. These image pairs are sent to two Sub-Networks simultaneously, both of which are trained using 6-DOF pose ground truth. The Translation Sub-Network is used to predict translation vector (x, y, z) while the FlowNet Orientation Sub-Network infers the orientation vector (i, j, k). The translation and orientation vector are concatenated to output the desired 3-D transformation. All the initial configurations are listed, and we employ Dropout to prevent overfitting during training. Values shown in convolutional layers represent the number of channels/depth.

them. We used FlowNet as it has been proved to effectively extract geometrical features useful for odometry task. In Translation Sub-Network, fully convolutional layers are utilized to extract features from projected depth images, while in FlowNet Sub-Network, we adopt the same configuration of convolutional layers from FlowNet [6]. Our intuition to take advantage of FlowNet is that it is constructed upon optical flow, which is heavily used in VO tasks to attain good rotation performance. Since the number of input channels of FlowNet is six, we simply expand each depth image to three channels by replicating it twice. Leaky Rectified Linear Units (Leaky ReLU) are applied after every convolutional layer in our architecture with the angle of negative slope set to 0.1 and using in-place operation. We employ transfer learning to initialize the weights of convolutinal layers in our FlowNet Sub-Network using pretrained FlowNet convolutional layers' weights. Note that we do not use any max pooling layers or batch normalization layers in our architecture since our comprehensive experiments indicate that adding either of these two layers significantly decreases the prediction accuracy. Moreover, both of these two sub-networks have the same fully connected layers settings as in the [16], which consists of two branches jointly trained by 6-DOF ground truth. Although their work used their own collected indoor

point cloud dataset to design the network, our experiment empirically proves that such a setting of fully connected layers has the best performance for PCO. Another important design is that we jointly learn poses for each sub-network. For example, for the Translation Sub-Network, although our final goal of this sub-network is to output translation vector, during the training phase, we jointly trained translation and orientation together as illustrated in Fig. 4 (orange color stands for 6-DOF pose). This is inspired by Grimes et al. [9], which demonstrated that regressing position and orientation separately performed worse compared to the full 6-DOF pose for training. To summarize, our neural network architecture is end-to-end, and easy to be trained since it only employs a few 2-D CNN layers and a small fully connected layers, eliminating the use of large RCNN architecture.

B. Output and Cost Function

Our network outputs a 6-DOF pose vector $\mathbf{v}$:

$$\mathbf{v} = [\mathbf{p}, \mathbf{q}] \qquad (5)$$

where $\mathbf{p}$ is a 3-D position and $\mathbf{q}$ is an Euler angle. We express orientation of pose as Euler angle rather than quaternion based on the reason that quaternion is subject to an extra unit constraint affecting the optimization of our network. Since the two sub-networks are trained in a supervised manner

simultaneously, our final objective loss function L across two sub-networks is defined as follows:

$$L = \|\mathbf{p} - \hat{\mathbf{p}}\|_2^2 + k * \|\mathbf{q} - \hat{\mathbf{q}}\|_2^2 \qquad (6)$$

where $\hat{\mathbf{p}}$ is ground truth of translation vector, $\hat{\mathbf{q}}$ is ground truth of orientation vector, $\| * \|_2^2$ measures the mean squared error (squared L2 norm) between each element in the estimation vector and ground truth vector, and k is a scale factor to balance the errors of position and orientation to be approximately equal.

V. EXPERIMENTAL EVALUATION

In this section, we introduce our experiment and training details. We compare our network to 5 baseline approaches, which are designed and tested on their best variations for our task. The **two-stream** network [16] performs the same task as ours and is set as the benchmark for the PCO task. We modified their original network to infer 6-DoF pose, and we retrain it from scratch on the same input data as the DeepPCO. **DeepVO** and **ResNet18** [10] are deep-learning approaches for visual odometry tasks and we use 2-D panoramic depth images as inputs. We select **PointNet** and **PointGrid** as comparisons for the 3-D encoding of point cloud due to their excellent performance in tasks such as semantic segmentation. For the PointNet, we remove the input transformation part.

A. Dataset

The KITTI VO/SLAM benchmark dataset is chosen for the experiment. The dataset consists of 11 sequences (Sequences 00-10), which are associated with ground truth pose. Each sequence contains consecutive point cloud data collected by a laser sensor for trajectory estimation. Since the dataset is relatively small compared to other benchmark datasets, we selected Sequence 00-03, 05-09 as training datasets, and Sequence 04 and 10 as test datasets which are chosen by most VO work since they can represent the major scenarios for agent or sensor movements in real world. Considering our loss function adopts Euler angle while the orientation of ground truth in KITTI is quaternion, we transformed orientation inside our experimental datasets to the Euler angle. All poses in KITTI are based on absolute transformation, so we converted them to relative transformation for the purpose of training the network.

B. Implementation and Settings

Our architecture is implemented with PyTorch framework. For evaluating our results, we choose Root Mean Square Error (RMSE) as our metric. Adam [11] optimizer was employed to train our neural network with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the learning rate from 0.0001 and then we applied 50% weight decay for every 10 epochs. Our network was trained for 30 epochs according to different encoding approaches and architectures. Batch size was set to 8. The scale factor k of the objective function is 100. We use 8-fold cross-validation for choosing hyperparameters. During the training phase, we also used data shuffling to prevent overfitting.

TABLE I: Test results on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement units are **m**).

Model	Sequence 04		Sequence 10	
	t_rel	r_rel	t_rel	r_rel
Two-stream [16]	0.0554	0.0830	0.0870	0.1592
ResNet18 [10]	0.1094	0.0602	0.1443	0.1327
DeepVO [20]	0.2157	0.0709	0.2153	0.3311
PointNet [17]	0.0946	0.0442	0.1381	0.1360
PointGrid [12]	0.0550	0.0690	0.0842	0.1523
DeepPCO (Ours)	**0.0263**	**0.0305**	**0.0247**	**0.0659**

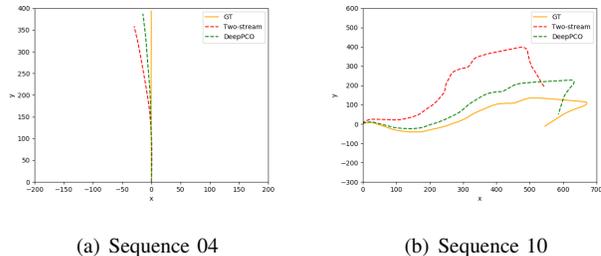(a) Sequence 04   (b) Sequence 10

Fig. 5: Trajectories of test results on Sequence 04 and 10.

C. Test Results

The test results are presented in Table I. Our experiment indicates that the proposed architecture outperforms all the baselines for both test sequences. A deeper convolutional neural network architecture, like ResNet18, is not necessary for achieving excellent odometry results. Among 3-D encoding approaches, PointGrid is more suitable for performing 6-DOF tasks than PointNet. In particular, our fully connected layers share very similar structure with the Two-stream approach. We trained them using the same settings on the same KITTI datasets, and the main difference is the convolutional parts, which we utilize more filters for feature extraction. Hence, it indicates that good feature extraction plays a significant role for odometry.

D. Trajectory Evaluation

Qualitative results are shown in Fig. 5 where we plot predicted trajectories of DeepPCO compared with TwoStream, as the most comparable baseline.

The proposed DeepPCO system can produce accurate pose estimation with respect to ground truth. Compared to the baseline approach, DeepPCO shows significant improvement. As seen in Fig. 5, during the first 200 metres of Sequence 04, DeepPCO has very low drift. However, after 200 meters, the drift gradually increases. In order to investigate the reason, we plotted the projected depth images as shown in Fig. 6. The plot suggests that large open areas can degrade the accuracy of pose prediction to some degree. Combining with inertial measurements could overcome some of these issues.
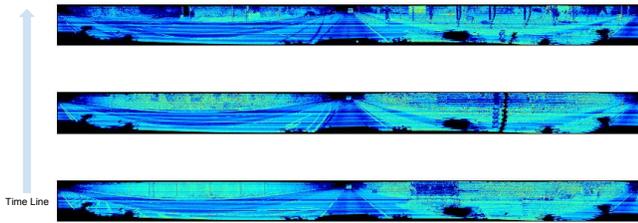
Fig. 6: Sample projected depth images from sequence 04 in the latter timeline. Images are plotted using depth values from the range [0, 255]. The surroundings are empty and lack features.

E. Comparison to Conventional Approaches

In our experiments, we also compared our work to the conventional approach. We run open-sourced LOAM using point clouds. For a fair comparison, we did not use any IMU data, and we employed the official evaluation tool released for KITTI to examine these two approaches. We take sequence 04 as an example. The percentage of translation error and deg/m of rotation error of LOAM are 2.3245% and 0.0108, while the results of DeepPCO are 3.1012% and 0.0177 respectively. We notice that the conventional approach outperforms the proposed technique, but DeepPCO still achieves good performance. One reason may be that the training datasets are limited, especially for rotation-related data in the KITTI datasets. The other reason may be that we did not utilize any geometric awareness in our network, which could be further explored in the future.

F. Ablation Study

In order to explore the impact of various components of our DeepPCO, we conduct the following ablation experiments. First, we split each sub-network to individually infer 6-DOF poses and examine whether our parallel architecture can outperform the individual ones. Second, for each sub-network, we keep the convolutional layers the same but use combined fully connected layers instead of two branches of fully connected layers in DeepPCO. Our purpose is to check whether the design of two branches for predicting translation and orientation separately is better than a single branch to jointly train and predict transformation. We note that all variants of our ablation architectures are trained from scratch.

a) **Individual Sub-Network**: Since we trained our sub-networks using full 6-DOF ground truth, we want to investigate whether any single sub-network was good enough to predict pose. Hence, we segmented our parallel architecture to two individual networks but kept all the parameters and hyperparameter settings unchanged. In order to ensure the fairness of our experiments, we randomly run 5 times for each sub-network to evaluate translation and orientation on sequence 04 and 10 respectively. Meanwhile, we calculated the average of RMSE for the 5 experiments. All the results are shown in Table II. Results suggest that all translation
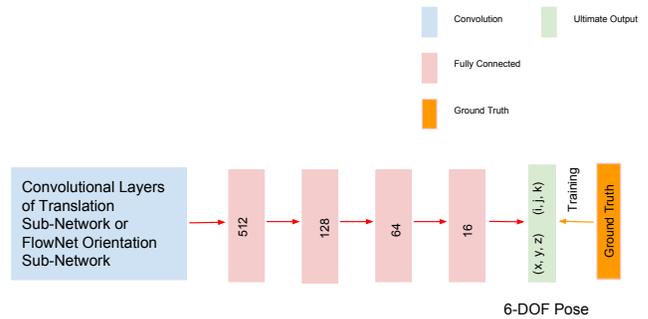

Fig. 7: Ablation experiment of single branch fully connected layers. All the parameter configurations of convolutional layers and fully connected layers are the same as DeepPCO. Different from DeepPCO in which transformation vector is trained using two branches, 3-D translation (x, y, z) and orientation (i, j, k) are jointly trained and inferred by just one branch here.

predictions on Sequence 04 and 10 using Translation Sub-Network are better than FlowNet Orientation Sub-Network, while all orientation predictions based on FlowNet Orientation Sub-Network are better than those generating from Translation Sub-Network. In other words, this ablation experiment demonstrated that Translation Sub-Network performs better for inferring translation whereas achieving worse orientation prediction than FlowNet Orientation Sub-Network. Exactly the opposite, FlowNet Orientation Sub-Network outperforms Translation Sub-Network for orientation inference. Therefore, it is a desirable choice for designing dual sub-networks to predict translation and orientation separately.

b) **Single Branch Fully Connected Layers** In our proposed architecture, we utilized two branches for predicting translation and orientation. However, we consider whether only using a single branch of fully connected layers if each sub-network can produce a better result. Thus, we design the ablation study for this situation as shown in Fig. 7. All the parameter configurations of layers remain the same. The only difference is that we use one branch to jointly learn transformation. The results are presented in Table III. While single branch FC layers are capable of attaining reasonable estimations, they are less accurate than our parallel architecture. Results indicate that adopting the design of two branches of FC layers in DeepPCO tends to have better transformation inference for the PCO task.

VI. CONCLUSION

In this paper, we have presented an end-to-end deep parallel neural network named DeepPCO for the point cloud odometry task. Two consecutive point clouds are processed into panoramic depth images, which are stacked and sent simultaneously to dual sub-networks for transformation estimation. Our approach shows good performance for the

TABLE II: Test results of individual sub-network on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement unit is meter (**m**). Each sub-network was trained using 6-DOF poses, and we conducted 5 times independently for each one.

	Translation Sub-Network Only				FlowNet Orientation Sub-Network Only			
	Sequence 04		Sequence 10		Sequence 04		Sequence 10	
Times	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel
1	**0.0288**	0.0421	**0.0243**	0.0735	0.0474	**0.0315**	0.0352	**0.0648**
2	**0.0282**	0.0392	**0.0235**	0.0731	0.0471	**0.0312**	0.0373	**0.0675**
3	**0.0262**	0.0429	**0.0248**	0.0737	0.0469	**0.0297**	0.0377	**0.0682**
4	**0.0307**	0.0417	**0.0234**	0.0755	0.0469	**0.0302**	0.0342	**0.0676**
5	**0.0281**	0.0426	**0.0247**	0.0740	0.0580	**0.0309**	0.0447	**0.0669**
Mean	**0.0284**	0.0417	**0.0241**	0.0740	0.0493	**0.0307**	0.0378	**0.0670**

TABLE III: Test results of various branches of fully connected (FC) layers on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement unit is meter (**m**). For single branch FC layers, each sub-network was trained using full 6-DOF pose while for DeepVO, each sub-network was learned by two branches, one for translation and the other for orientation.

	Sequence 04		Sequence 10	
Model	t_rel	r_rel	t_rel	r_rel
Single Branch FC	0.0559	0.0843	0.0327	0.0729
Two Branches FC	**0.0284**	**0.0307**	**0.0241**	**0.0670**

odometry task in the 3-D real-world environment. Several interesting extensions could be considered from our work, such as integrating our system to a full learning-based SLAM system, evaluating our approach on more challenging environments like severe weather, or developing learning-based sensor fusion methods for odometry tasks. In the future, we plan to add geometric optimization to our architecture, which may further improve our performance. We hope our design choices and experiments on 2-D and 3-D encoding approaches and corresponded architectures can inspire further research for point cloud-related tasks and system developments. We believe that this work is an important step towards developing robust point cloud-based odometry.

ACKNOWLEDGMENT

REFERENCES

[1] P. J. Besl and D. N. McKay. A Method for Registration of 3-D Shapes. *TPAMI*, 1992.
[2] C. Cadena, L. Carlone, C. Henry, L. Yasir, S. Davide, N. Jose, R. Ian, and J. L. John. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 2016.
[3] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni. Selective Sensor Fusion for Neural Visual-Inertial Odometry. *CVPR*, 2019.
[4] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. *AAAI*, 2017.
[5] G. Costante and T. A. Ciarfuglia. LS-VO: Learning Dense Optical Subspace for Robust Visual Odometry Estimation. *ICRA*, 2018.
[6] P. Fischer, E. Ilg, H. Philip, C. Hazrbas, P. V. D. Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *ICCV*, 2015.
[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *CVPR*, 2012.
[8] R. Gomez-Ojeda, Z. Zhang, J. Gonzalez-Jimenez, and D. Scaramuzza. Learning-based Image Enhancement for Visual Odometry in Challenging HDR Environments. *ICRA*, 2018.
[9] M. Grimes and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. *ICCV*, 2015.
[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.
[11] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.
[12] T. Le and Y. Duan. PointGrid: A Deep Network for 3D Shape Understanding. *CVPR*, 2018.
[13] B. Li, T. Zhang, and T. Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *RSS 2016*.
[14] R. Li, S. Wang, Z. Long, and D. Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. *ICRA*, 2018.
[15] P. Muller and A. Savakis. Flowdometry: An Optical Flow and Deep Learning Based Approach to Visual Odometry. *WACV*, 2017.
[16] A. Nicolai, R. Skeele, C. Eriksen, and G. A. Hollinger. Deep Learning for Laser Based Odometry Estimation. *RSS Workshop on Limits and Potentials of Deep Learning in Robotics*, 2016.
[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017.
[18] M. R. U. Saputra, A. Markham, and N. Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. *ACM Computing Surveys (CSUR)*, 2018.
[19] A. V. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. *RSS*, 2009.
[20] S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. *ICRA*, 2017.
[21] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. *arXiv*, 1906.01140, 2019.
[22] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry. *ECCV*, 2018.
[23] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. *RSS*, 2014.
[24] J. Zhang and S. Singh. Visual-lidar Odometry and Mapping: Low-rift, Robust, and Fast Localization. *ICRA*, pages 393–398, 2015.