Point Cloud Registration Based on Direct Deep Features With Applications in Intelligent Vehicles

Liang Li, *Member, IEEE*, and Ming Yang, *Member, IEEE*

***Abstract*— Point cloud registration is widely used in the research of intelligent vehicles, typical problems include map matching, visual odometer, pose estimation, *etc*. This paper proposes a deep learning-based registration method that can input point clouds directly, thereby preventing information loss of preprocessing needed by alternative deep-learning approaches. Our network, named DPFNet (Direct Point Feature Net), gradually downsamples the point cloud and aggregates points around determined reference points to formulate local features automatically. This is facilitated by a novel convolution-like operator and a novel loss function. The points in the point cloud are mapped to a high dimensional embedding through the designed deep neural network, where every embedding reflects the local feature of a specific spatial area. Based on the embedding features, correspondences between points can be estimated robustly and the registration between the point clouds can be obtained using an external geometric optimization algorithm. Experimental results on open benchmarks validate the proposed method and show that its performance is favourable over several baseline methods. Specifically, we test the proposed algorithm on KITTI benchmark, which shows its potential in tasks of intelligent vehicles, *e.g.*, map matching, visual or LiDAR odometer.**

***Index Terms*— Point set registration, deep learning, map matching, pose estimation.**

I. Introduction

POINT cloud registration is an important problem in a large number of intelligent vehicles related applications, *e.g.*, mapping [1], localization [2], 3D object semantic segmentation [3],. Given two point clouds $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$ and $\mathbf{Q} = \{q_1, q_2, \ldots, q_m\}$, where $p_i, q_i \in \mathbb{R}^d$ are d-dimensional points, the task of rigid point cloud registration is to find the optimal transformation $\mathcal{T}$ that minimizes the distance between the point clouds $\|\mathbf{P} - \mathcal{T}(\mathbf{Q})\|$. Estimating the transformation $\mathcal{T}$ requires (iteratively) establishing (soft) correspondences between $\mathbf{P}$ and $\mathbf{Q}$. Obtaining these correspondences can be facilitated by computing and matching features, which contain local geometric information of small patches around points in the point clouds. This local geometric information of small patches is more distinctive and robust for establishing point set correspondences than the single point coordinates.

Manuscript received December 6, 2020; revised May 31, 2021 and October 1, 2021; accepted October 21, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62173228/U1764264 and in part by the International Chair on Automated Driving of Ground Vehicle. The Associate Editor for this article was C. Wen. *(Corresponding author: Ming Yang.)*

The authors are with the Key Laboratory of System Control and Information Processing, Ministry of Education of China, Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: mingyang@sjtu.edu.cn).

Digital Object Identifier 10.1109/TITS.2021.3123619

An example is Fast Point Feature Histograms (FPFH) [4] that encodes a point's k-neighborhood geometrical properties to incorporate the geometric structure of local regions. Signature of Histograms of OrienTation (SHOT) [5] considers both signatures and histograms. Some other "hand-crafted" features can be found in [6]–[8], *etc*. In the current era of deep learning, the idea of learning features by deep neural networks from training data, is shown to be promising. Some pioneering work includes PPFNet [9], PPFFoldNet [10], 3DSmoothNet [11]. These methods first extract patches from input point clouds and then preprocess these patches. For example, PPFNet extracts Point Pair Features (PPF), 3DSmoothNet rasterizes the patches into 3D voxel and computes smoothed density values (SDV) with Gaussian kernels. The preprocessed patches are fed into the neural network to get the embedding features for every patch. Typically, to train the neural network, a triplet-like loss function is used that minimizes distances between corresponding features, and at the same time, maximizes that of non-corresponding features.

In this paper, we propose a framework that is based on deep neural networks and that extracts features from the point cloud directly without the need for preprocessing. Assuming that the dimension of the input point cloud is $N \times d$, where N is the number of points, and d is the dimension of every point, and the output of the network is $M \times d'$, where $M < N$ and $d' > d$, our network consists of two types of modules, one type is the *point selection module* that downsamples the points from N points to M points, while retaining the overall geometric structure of the point cloud. The other type of modules, is the *feature computation module* using deep convolution-like neural networks that, step-by-step, brings the computed features from a d to a d' dimensional embedding space. Conceptually, these feature computation modules incorporate the information of points that are abandoned by the selection modules into the embedding features. These two types of modules proceed one by another to abstract the $N \times d$ point cloud into a $M \times d'$ set of high-dimensional features. All features in the output can be seen as an embedding that incorporates information of a local region with the selected points at their centers. Herein, only downsampling is considered, which is inspired by the convolution and pooling operation on 2D images. The size of the image becomes smaller from layer to layer, while the dimension of every pixel becomes larger in this process. And the receptive field becomes bigger and bigger. Similarly in our case, the points become sparser, while dimension of every point increases. This process makes the feature region every point convey increases when the network going deep. Regarding minimizing the costs of sensors, and only sparse point cloud is available, upsampling [12]–[14] or inpainting [15], [16] is needed before feeding the point cloud into the proposed network. In some cases, upsampling is important to make the point cloud denser. While the quality of upsampling or point cloud completion can affect the following feature extraction and feature matching. In addition, for the 3D LiDAR such as Velodyne VLP-16 or above, we found the density of the point cloud is appropriate for the matching task (please see Section IV). Since upsampling the point cloud to get a better feature extraction is not the focus of this paper, we leave it our future work to test how different upsampling strategies affect the feature extraction. To learn the optimal features from training data, a Siamese network structure is used together with a structure regulated loss function that models the similarities of corresponding embeddings and the dissimilarities of non-corresponding embeddings. In short, the contributions of this paper are:

1. We propose a deep neural network approach that consumes point clouds directly with no need for preprocessing in terms of embedding calculations on local patches. This avoids information loss typically caused by rasterization, histogram statistics, *etc* as used in alternative methods.
2. A structure regulated loss function is designed for efficient and robust optimization. It assigns soft constraints on corresponding and non-corresponding embeddings, rather than using a dichotomous approach. Spatial correlations are also modeled through this loss function.
3. We propose a novel sorted convolution operation to capture geometric information of point clouds. Detailed analysis and experiments are carried out to compare our method with baseline algorithms.

The remaining of this paper is organized as follows. In Section II, we discuss the state-of-the-art in relation to our novel DPFNet approach. Section III, introduces the proposed network and the intuition behind its design. Section IV presents experimental results and cross validation on multiple benchmark datasets, together with comparisons and ablation studies. The conclusions of this study are provided in Section V.

II. Related Work

Point cloud registration algorithms can be classified into four categories: 1) point-wise correspondence based, 2) point distribution based, 3) hand-crafted feature based, and 4) learned feature based.

A. Point-Wise Correspondence Based Registration

Point-to-point correspondences are established based on the spatial distribution of the points (*e.g.*, nearest neighbor algorithm) first. Then, the sum of the distances between corresponding point pairs in terms of transformation is used as the cost function. Optimization based on Singular Value Decomposition (SVD) or quaternion computations obtain the optimal transformation under this correspondence relation. Usually these two procedures proceed in an iterative manner, *i.e.*, establishing correspondence, optimization, transforming the source point cloud with the optimal transformation, establishing new correspondence, and so forth. The classical ICP algorithm [17] and its variants [18], [19] follow this pipeline until a convergence condition is satisfied. The trimmed ICP [20] only considers a subset of the points that have the least point-to-point distances, during each iteration. GO-ICP [18] uses branch-and-bound to explore the optimal transformation over the complete parameter space, to prevent convergence to local minima. EM-ICP [21] models multiple matches that are weighted by normalized Gaussians and optimizes the problem with Expectation Maximization (EM), to make the registration more robust to (outlier) noise. In general, methods based on the ICP paradigm are known to be sensitive to initialization, noise, and outliers. The point cloud registration problem can be also modeled with filtering model, and use Kalman filter [22] or particle filter [23] to solve it.

B. Point Distribution Based Registration

Rather than establishing point-to-point correspondences, these types of methods try to fit a probabilistic distribution with the assumption that points in the point clouds are sampled from that distribution. Gaussian Mixture Model (GMM) is the most common model to approximate the distribution, where the mean of every Gaussian function is provided by the point positions and the number of components in the GMM is equal to the number of points. Coherent Point Drift (CPD) [24], [25] models one point cloud as a GMM, and its cost function is related to the probability that the other point cloud is sampled from this GMM. GMM-L2 [26] models both point clouds as GMMs and optimizes an L2 distance between them. Other than GMM, there are alternative models for distribution modeling, such as kernel correlation [27] and mixture of t-distributions [28]. Furthermore, Normal Distribution Transform (NDT) [29] is commonly used in the field of robotics, due to its robustness and computational efficiency, as it rasterizes point clouds into grids and computes a Gaussian distribution for every grid. In general, distribution based methods improve robustness to noise and outliers, as they incorporate probabilistic models that intrinsically allow for soft correspondences between point sets instead of hard point-to-point correspondences.

C. Hand-Crafted Feature Based Registration

An alternative technique for robust point cloud registration, is using features to establish point-to-point correspondences. Features are formed by local regions or small patches and reflect specific characters therein. FPFH [4] considers Euclidean distances and angles between a reference point and its k neighbors with their normals. TOLDI [6] concatenates three orthogonal view features in a local reference framework. RoPS [8] calculates central moments and entropy of neighboring points rotationally projected onto 2D planes. More types of point features can be found in the survey paper [30], [31].

D. Learned Feature Based Registration

With the exploration of applying deep learning to 3D data [32]–[35], registration based on deep learning methods are emerging rapidly. PPFNet [9] selects a number of local patches from the point cloud, converts them to Point Pair Features (PPF) and feeds PPF into PointNet, to get high-dimensional embeddings using a N-tuple loss function to model similarity. PPF-FoldNet [10] extends PPFNet using an encoder-decoder framework to introduce rotation invariance to the embedding. 3DFeat-Net [36] also uses PointNet as backbone network, and assigns weights which are learned from local patches. The very recent 3DSmoothNet [11] rasterizes every local patch into 3D voxels, computes point density of every voxel with a Gaussian kernel, and then it uses L2-Net [37] with 3D convolutions, to learn features based on a batch hard loss function. PointNetLK [38] does not select local patches, it deals with the entire point cloud, and represents the point cloud with a single feature vector. Then optimization of the transformation is by comparing the two feature vectors in an iterative manner like that of ICP. To the best of our knowledge, all current learning based methods need to select local patches or interest points in advance, thus they are not end-to-end methods. Besides, most of current methods need to convert point clouds into another type of representation before feeding them into the deep neural networks. The above can potentially lead to information loss and our approach aims to solve this.

III. Approach

This section details our network architecture, convolution module, and design of the loss function.

A. Basics of the Network

In this paper, neural networks for point cloud registration can be seen as a function $\mathcal{F}$ that converts a low-dimensional point with its position to a high-dimensional embedding given its surrounding points. We break this into several subroutines that have a structure that is similar to convolutional neural network layers used in image processing. Because of the natural grid shape of images, the output of convolutional calculation has also a strict grid shape. For point clouds, the situation is much different, structure of the data is unordered and not grid-based as that of 2D images, it is not possible to apply convolutions in the same manner as in image processing. Besides, the size of the point cloud will decrease very slowly with the standard convolution. If we rasterize the point cloud into 3D voxel, it may lose spatial and geometric information, which is undesired. Due to this, we propose splitting the neural network layers into two subroutines. This concept is shown in equation (1):

$$\mathcal{F}(\mathbf{P}) = \Xi_{i=1}^{L}\mathbf{FEN}_i(\mathbf{P}) = \Xi_{i=1}^{L}\mathbf{FN}_i(\mathbf{DN}_i(\mathbf{P})) \quad (1)$$

where the symbol $\Xi_{i=1}^{L}$ denotes applying an L-layered function, *i.e.*, $\Xi_{i=1}^{L} f_i(x) = f_L(f_{L-1}(\dots f_1(x)))$. $\mathbf{FEN}_i$ is the i-th layer in our network mainly consisting of a downsampling routine $\mathbf{DN}_i$ and a feature extraction and aggregation network $\mathbf{FN}_i$. The process is illustrated in Fig. 1.

The ideal output is that every embedding feature represents geometric information around specific points in the point cloud. Therefore, it is important that we need to keep relevant geometric information when downsampling in every $\mathbf{DN}_i$. Randomly downsampling may not result in a uniform point distribution over the entire space, which is not desirable. Some uniform downsampling algorithms such as voxel grid downsampling or Farthest Point Sampling (FPS) could be adopted here. We select FPS as the downsampling method in this paper, as in [33]–[35], because it can return a specific number of points, which is convenient for the following processing steps. Assume there are N_{i-1} points from the $i-1$-th layer as set A, and the expected number of points after downsampling is N_i with set B, $\mathbf{DN}_i$ first randomly selects a point from N_{i-1} as the seed point. Now A has $N_{i-1} - 1$ points, and B has one point. The second point is the point from A whose distance is the largest to the seed point in B. Then the distance of a point A to B is the minimal value of all the distances to points in B. The third point is still the farthest point in A to B. The sampling continues in this manner until B has N_i points. Note that downsampling $\mathbf{DN}$ based on FPS cannot identify where the features are rich and distinctive. The ideal sampling strategy is that sampling with a higher probability of the area that has higher potential to have good features compared with that is less likely to have effective features. Some advanced point selection algorithms [39] may be more promising for this task, and we leave it our future work to explore a better sampling strategy.

To deal with disorder of the input point cloud, PointNet [32], PointNet++ [33] and PointSIFT [34] use a symmetric function(*e.g.*, max-pooling), after which no matter what the input order is, only the maximal value across all the points of one feature layer will continue to the output. Different from that, PointCNN [35] learns a permutation matrix from the points' positions, multiplies it with the feature map and then

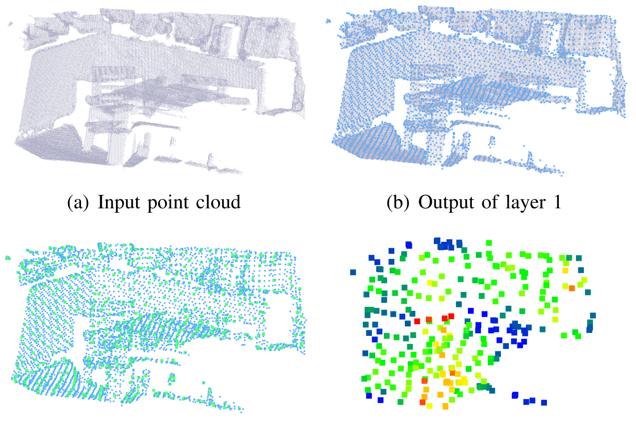

(a) Input point cloud (b) Output of layer 1 (c) Output of layer 2 (d) Output of layer L

Fig. 1. An illustration of the downsampling operation in neural network. Density of the points decreases layer to layer, while the spatial information every point conveys increase. The larger size of the point, the higher dimension (more information) the feature it is affiliated with. (a) Original point cloud input to the neural network, where the number of points is N_0 and the dimension of every point is 3. (b) Point cloud after layer 1, compared with (a), number of points is downsampled, *i.e.*, $N_1 < N_0$, while features affiliated to every point are enriched as they represent geometric features of their neighboring region, denoted as blue markers with larger size, the original points of (a) are still presented in gray. (c) Point cloud after layer 2, similar operation with (b), output denoted as green markers with larger size, input is from (b) denoted in blue. (d) The final output embeddings, every point aggregates neighboring features.

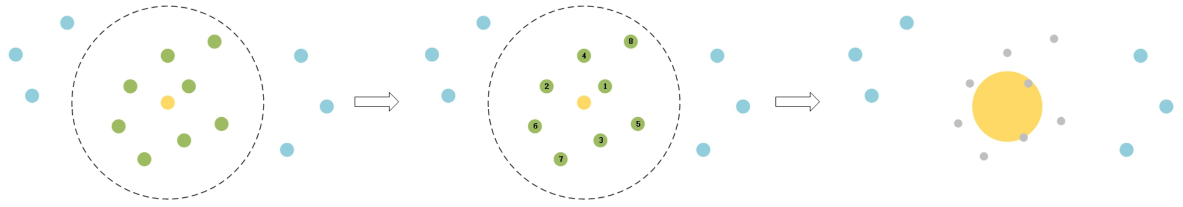


Fig. 2. An illustration of the novel convolutional operation using a sort function. The yellow point denotes the reference point, green points are the selected neighbors within a threshold radius, and blue points are beyond that threshold. The neighboring points are sorted according to their own distance to the reference point. Then convolution is conducted with all the neighboring points to obtain a new feature vector, denoted as a bigger yellow point that represents information for the circled region. Note that blue points may become reference points or may be selected neighbors of other reference points. For clarifying, we only focus on one specific reference point.

applies convolutional computation to the permutated feature map. While for our task, max-pooling loses lots of spatial information and learned permutation is a fuzzy or heuristic manner which may be inefficient. Therefore, in this paper, we utilize the spatial distribution of the points by sorting the feature vectors according to their point's Euclidean distance to the reference point. Because spatial positions of points play an important role when comparing similarity of two patches, a weighted combination of features from near to far in a neighboring domain reflect both distribution and shape characteristics of the given patch.

The process of layer $\mathbf{FN}_i$ in the network is mainly made up of our novel sorted convolution operator $\mathcal{F}_i^S$ that is provided by:

$$\begin{aligned}&\mathcal{F}_i^S\\&= \mathbf{FN}_i^S(\mathbf{P}_i, \mathbf{P}_{i-1}, \mathcal{F}_{i-1})\\&= Conv(\mathbf{K}_i, Sort(FC(Group(\mathcal{F}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i-1})), \mathbf{P}_i, \mathbf{P}_{i-1}))\end{aligned} \tag{2}$$

where $Conv$ and FC denote convolutional and fully-connected layers respectively, and $\mathbf{K}_i$ is the convolution kernel. The function $Group$ groups the feature vectors within a given radius, as done in PointNet++ [33], the function $Sort$ sorts the feature map based on the Euclidean distance between the points and their reference points. Assume that the dimension of $\mathcal{F}_i^S$ is $N_i \times C_i$, where N_i denotes the number of points of the i-th layer, and C_i is channel of every point's feature. The output of function $Group(\mathcal{F}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i-1})$ is $N_i \times M_i \times C_{i-1}$, which means we find M_i neighboring points for every downsampled point. Given the spatial coordinate of a query point in $\mathbf{P}_i$, the function $Group$ finds all points (in $\mathbf{P}_{i-1}$) that are within a radius to the query point. And we set M_i the same value for every point in $\mathbf{P}_i$. If there are more than M_i points, just select the first M_i points with smallest distance. If there are less than M_i points, duplicate the query point itself to until getting M_i points. Then the FC lifts each neighboring points to C'_{i-1} channels with kernel size 1×1, which yields the output of FC with $N_i \times M_i \times C'_{i-1}$. So, the FC operation only operates every point's own feature. Function $Sort$ perturbs orders of the neighboring points for every reference point, the output dimension remains $N_i \times M_i \times C'_{i-1}$. Finally we adopt the convolution to the feature map, the kernel size of the convolution is $M_i \times C'_{i-1}$, the channel number is C_i, which means the output dimension of i-th layer is $N_i \times C_i$. The $Conv$ is the general convolutional operation same as 2D image processing. An illustration of this concept is shown in Fig. 2. Besides the output of the sorted convolution $\mathcal{F}_i^S$, we also apply a symmetric function to the input feature map. Rather than max-pooling, we choose weighted average as the function. The weight are the normalized distances between the point and its reference point, resulting in $\mathcal{F}_i^W$. To generate $\mathcal{F}_i^W$, the function group is first adopted to select M_i points in $\mathbf{P}_{i-1}$ for every point in $\mathbf{P}_i$. Then, using the FC to lift the feature channels. And every point is assigned a weight which is based on the normalized distance. Finally, the FC function lift the feature channels to C_i to yield $\mathcal{F}_i^W$. The final output is again a weighted average of the output of the sorted convolution $\mathcal{F}_i^S$ and the output of the symmetric function $\mathcal{F}_i^W$, i.e. the final output of layer i is $\mathcal{F}_i = \alpha\mathcal{F}_i^S + \beta\mathcal{F}_i^W$. The complete computation of the network's layers is provided in Algorithm 1, where in line 3, $W(\mathbf{P}_i, \mathbf{P}_{i-1})$ is the normalized weight function, $W_{jk} = \frac{\exp(-\|p_i^{jk} - p_i^j\|)}{\sum_{k=1}^{M}\exp(-\|p_i^{jk} - p_i^j\|)}$. We sum the feature map in the sampling axis, *i.e.*, dimension of $FC(Group(\mathcal{F}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i-1}))$ is $N_i \times M_i \times C'_{i-1}$, dimension of $\mathcal{F}_i^W$ is $N_i \times C_i$. Finally, we use PointSIFT [34] to incorporate neighboring features further, it could also be seen as a part of the feature aggregation module. Given the input point set with size $N_i \times C_i$, the output size of PointSIFT is still the same as $N_i \times C_i$, but assigns a new C_i-dimensional feature to every point. Inspired by the 2D descriptor SIFT [40], PointSIFT models various orientations and is invariant to scale. Please refer to [34] for a detailed description of the method.

B. Network Architecture

Two parallel networks with sharing parameters (*i.e.*, Siamese network) are used to extract features of the two input point cloud respectively. Every network consists of the same downsampling and feature aggregation modules, and they proceed one by another. In this way, the point clouds become gradually sparser, layer after layer, while the dimension of features affiliated to each point becomes higher and higher, as the space they represent becomes larger. An illustration of the entire network is shown in Fig. 3. The structure is also presented in Algorithm 2.

The number of points in every layer N_i and the number of channels C_i are hyper-parameters that could be tuned by the user. Setting N_i too small may make the final embeddings

Algorithm 1: Feature Extraction Network (FEN)

```
  Input: $\mathbf{P}_0$, $\mathbf{F}_0$                        ▷ Input point cloud and initial feature (if have)
  Output: $\mathbf{P}_L$, $\mathcal{F}_L$                       ▷ Downsampled point cloud and its feature vectors
1 while $i$ in $L$ do
2     $\mathbf{P}_i = \mathbf{DN}_i(\mathbf{P}_{i-1}, N_i)$       ▷ Downsample point cloud using FPS to $N_i$ points
3     $\mathcal{F}_i^S = \mathbf{FN}_i^S(\mathbf{P}_i, \mathbf{P}_{i-1}, \mathcal{F}_{i-1})$       ▷ Extract feature as equation (2)
4     $\mathcal{F}_i^W = FC(\sum(W(\mathbf{P}_i, \mathbf{P}_{i-1}) \times FC(Group(\mathcal{F}_{i-1}, \mathbf{P}_i, \mathbf{P}_{i-1}))))$   ▷ Weighted average feature
5     $\mathcal{F}_i = \alpha\mathcal{F}_i^S + \beta\mathcal{F}_i^W$       ▷ Combine two features
6     $\mathcal{F}_i = PointSIFT(\mathcal{F}_i)$       ▷ Use PointSIFT to merge features further
7 end
```

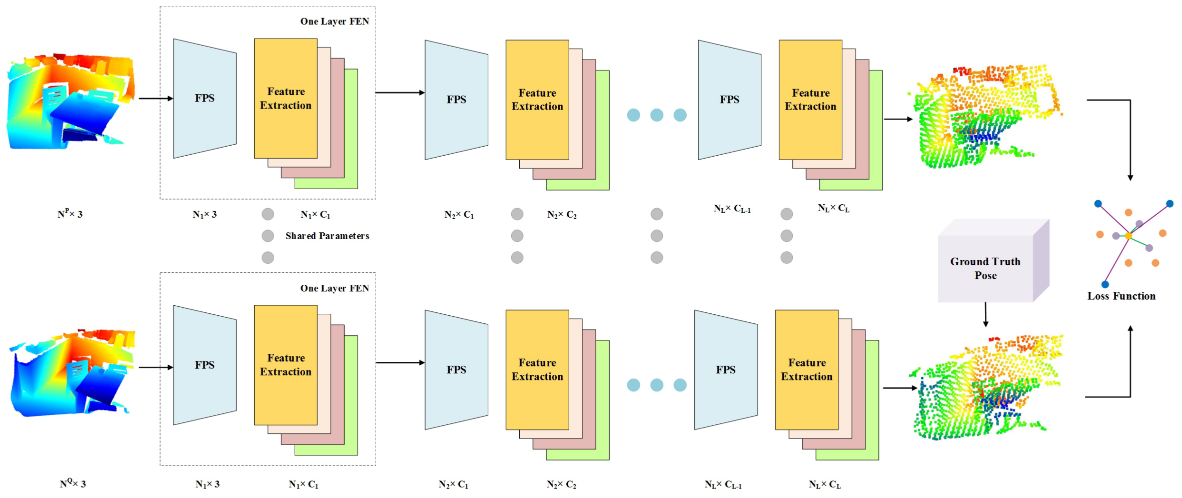


Fig. 3. The proposed network architecture of DPFNet. Every FEN layer mainly consists of two modules: 1) $\mathbf{DN}_i$ for downsampling by FPS and 2) $\mathbf{FN}_i$ for feature extraction by lifting feature channels incorporating surrounding points. There are four parts in feature extraction: a Fully-connected layer (FCN) denoted by yellow, a sort function denoted by pink, a Convolutional layer (CNN), a weighted average combination denoted by dark pink, and PointSIFT denoted by green. The layers also contain the typical final ReLU activation function and a batch normalization layer, which are not shown here. The training and testing utilize the same network architecture except that there is ground truth of the transformation between the two point clouds for supervising in the training phase.

Algorithm 2: Direct Point Feature Network (DPFNet)

```
  Input: $\mathbf{P}_0$, $\mathbf{F}_0^P$, $\mathbf{Q}_0$, $\mathbf{F}_0^Q$         ▷ Input point clouds and their initial features
  Output: $\mathbf{P}_L$, $\mathcal{F}_L^P$, $\mathbf{Q}_L$, $\mathcal{F}_L^Q$, $loss$   ▷ Downsampled point clouds, their features, loss
1   $\mathbf{P}_L, \mathcal{F}_L^P = FEN(\mathbf{P}_0, \mathbf{F}_0^P)$         ▷ Feature extraction using Algorithm 1
2   $\mathbf{Q}_L, \mathcal{F}_L^Q = FEN(\mathbf{Q}_0, \mathbf{F}_0^Q)$         ▷ Feature extraction with sharing parameters
3   $loss = LOSS(\mathbf{P}_L, \mathcal{F}_L^P, \mathbf{Q}_L, \mathcal{F}_L^Q)$     ▷ Compute loss
4   Return: $\mathbf{P}_L$, $\mathcal{F}_L^P$, $\mathbf{Q}_L$, $\mathcal{F}_L^Q$
```

lose structure information. Setting C_i too small may make the embedding not capture information in local regions precisely. While setting both too large, would generate redundant information and may cause overfitting. In this paper, we use 5 layers ($L = 5$) and the number of points in every layer is $[N_1, N_2, N_3, N_4, N_5] = [50000, 30000, 10000, 4096, 1024]$ and the number of channels in every layer is $[C_1, C_2, C_3, C_4, C_5] = [16, 32, 64, 128, 256]$.

C. Loss Function

The aim is to draw the corresponding feature vectors to each other as near as possible while drive the non-corresponding feature vectors as far as possible from each other. PPFNet [9] and 3DSmoothNet [11] *etc.* use a triplet-based loss function. They divide the patches into anchor, positive, and negative groups, penalize distance between anchors and positive samples while maximizing distance between anchors and negative samples. As they sample patches from the overlapping parts of two point clouds, one patch could find its corresponding patch in the other point cloud according to the Euclidean distance, and all the rest are negative patches. While for our case, we sample the whole point cloud, which means patches from the independent parts may not have its positive sample in the other point cloud. Besides, one point in the output downsampled point may not have the exact corresponding point in the other output point cloud, but it may have corresponding point near it. Thus, a hard constraints on the patch pairs is not suitable for our network.

We therefore design a novel structure-aware loss function to deal with the incompatibility of current loss functions. According to the spatial distribution, we classify points into three categories: potential positive, intervening, and negative. For a given point p_L^i in $\mathbf{P}_L$, we divide the space of $\mathbf{Q}_L$ into three regions based on their distances to p_L^i. Points within threshold po are selected to be potential positive points and are denoted as $\mathbf{Q}_L^{po}$, points beyond threshold ne are deemed to be negative points and are denoted as $\mathbf{Q}_L^{ne}$, points between po and ne maybe relevant to the reference points p_L^i with some probability or maybe negative and we define this region as intervening, denoted as $\mathbf{Q}_L^{in}$. Thus we have $\mathbf{Q}_L = \mathbf{Q}_L^{po} \bigcup \mathbf{Q}_L^{in} \bigcup \mathbf{Q}_L^{ne}$, which means every point has one label. Based on the model above, we define loss function as equation (3):

$$loss = \sum_{i=1}^{N} \max((\sum_{j=1}^{N_{po}} W_{ij}^{po} \parallel \mathcal{F}_{L(ij)}^{Q(po)} - \mathcal{F}_{L(i)}^{P} \parallel - \sum_{k=1}^{N_{ne}} W_{ik}^{ne} \parallel \mathcal{F}_{L(ik)}^{Q(ne)} - \mathcal{F}_{L(i)}^{P} \parallel) + margin, 0)) \quad (3)$$

where the weights $W_{ij}^{po} = \frac{\exp(-\|q_{L(j)}^{po} - p_L^i\|)}{\sum_{j=1}^{N_{po}} \exp(-\|q_{L(j)}^{po} - p_L^i\|)}$ and $W_{ik}^{ne} = \frac{1}{N_{ne}}$, $q_{L(j)}^{po}$ and $q_{L(k)}^{ne}$ are the j-th positive point and k-th negative point for p_L^i respectively, the same symbolism also holds for the feature vector $\mathcal{F}_{L(j)}^{Q(po)}$ and $\mathcal{F}_{L(k)}^{Q(ne)}$, $margin$ is the margin value and we set it to 0.5 in this paper. We minimize feature differences between reference points and their potential positive points with weights based on distance. Simultaneously, we also maximize it between reference points and their negative points also with weights. We leave the intervening points as a kind of uncertainty margin and do not penalize on it. For sparse point cloud, determining the value of po and ne is tricky, and it can affect the performance of the network. However, in this paper, we mainly consider dense point cloud, which is available in many cases of intelligent vehicles. Then, setting them in a reasonable range will not influence the output of the network obviously based on both empirical analysis and experimental results. For point cloud obtained by 3D LiDAR, the points are sparse at long ranges, then according to equation (3), they do not have positive patches, which means they are just excluded from the corresponding calculation. For different types of datasets, *e.g.*, indoor and outdoor, the densities are different. Fine-tuning the parameters can affect the registration results. For the indoor dataset we set po as 0.1m, and ne as 0.5m. For the outdoor dataset, we set po as 0.3m and ne as 1.0m. In training, since the ground truth of point cloud transformation is available, we can apply it to one point cloud. And then the corresponding positive and negative patches are known. Thus, the loss can be calculated based on it.

D. RANSAC Based Registration

Assume that outputs of the network are $\mathbf{P}_L = \{p_L^1, p_L^2, \ldots, p_L^n\}$ with its embedding $\mathcal{F}_L^P = \{f_{L(1)}^P, f_{L(2)}^P, \ldots, f_{L(n)}^P\}$, and $\mathbf{Q}_L = \{q_L^1, q_L^2, \ldots, q_L^n\}$ with its embedding $\mathcal{F}_L^Q = \{f_{L(1)}^Q, f_{L(2)}^Q, \ldots, f_{L(n)}^Q\}$, we could get correspondence of $\mathbf{P}_L$ in $\mathbf{Q}_L$ by finding the nearest embedding in $\mathcal{F}_L^P$ of every embedding in $\mathcal{F}_L^Q$. For convenience, we assume correspondence of q_L^i is p_L^i, and the points are placed in terms of the embedding distance in an ascending manner, *i.e.*, $\parallel f_{L(1)}^P - f_{L(1)}^Q \parallel \leq \parallel f_{L(2)}^P - f_{L(2)}^Q \parallel \leq \cdots \leq \parallel f_{L(n)}^P - f_{L(n)}^Q \parallel$. Then we abandon the pairs that are not correspondent obviously, *i.e.*, their embedding distances are larger than a threshold. Then we could get two smaller sets $\mathbf{P}'_L = \{p_L^1, p_L^2, \ldots, p_L^m\}$ and $\mathbf{Q}'_L = \{q_L^1, q_L^2, \ldots, q_L^m\}$, where $m < n$. We select four points in each sets in order, *i.e.*, the selection of first iteration would be $\{p_L^1, p_L^2, p_L^3, p_L^4\}$ and $\{q_L^1, q_L^2, q_L^3, q_L^4\}$. For each of the pair, we have the equation:

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} p_{L1}^i \\ p_{L2}^i \\ p_{L3}^i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} q_{L1}^i \\ q_{L2}^i \\ q_{L3}^i \end{bmatrix} \quad (4)$$

where $i = 1, 2, 3, 4$, $\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} = \mathbf{R}$ is the rotation matrix, $\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \mathbf{t}$ is the translation vector. We could get solution by solving these four equations combinedly. Next step is to check if the obtained rotation matrix is close to a strict rotation matrix. If so, applying the obtained transformation to all the correspondence pairs. If distance between two corresponding points after transformation is with a threshold, it is treated as an inlier, otherwise it is an outlier. Then new four points, *e.g.*, $\{p_L^1, p_L^2, p_L^3, p_L^5\}$ and $\{q_L^1, q_L^2, q_L^3, q_L^5\}$ are selected for next iteration. This procedure iterates until the convergence condition is reached, *e.g.*, number of inliers is larger than a presented value.

IV. Experiments

In the experiments, we compare our DPFNet to the very recent state-of-the-art 3DSMoothNet, which has shown to outperform many other learning-based approaches, and also the very recent end-to-end deep learning-based method PointNetLK. And we also compare the proposed method with other common non-learning baselines methods. For all feature-based methods, we use default Random Sampling Consensus (RANSAC), to obtain the transformation from the estimated correspondences. All our experiments run on an Intel Core i7 PC with 16GB RAM and a TITAN XP GPU.

A. Results of Point Cloud Registration

1) Datasets: We perform the experiments on two datasets: 3DMatch[1] [41] and KITTI[2] [42]. The former is an indoor

[1] http://3dmatch.cs.princeton.edu/

[2] http://www.cvlibs.net/datasets/kitti/

TABLE I
REGISTRATION RECALL AND ERROR OF DIFFERENT METHODS ON 3DMATCH DATASET

Method	ICP		CPD		FPFH		3DSmoothNet		PointNetLK		DPFNet(ours)	
	recall	error	recall	error	recall	error	recall	error	recall	error	recall	error
Kitchen	0.014	0.034	0.010	0.039	0.315	**0.024**	0.356	0.041	0.012	0.041	**0.402**	0.078
Home	0.027	0.027	0.010	0.038	0.393	**0.025**	0.352	0.038	0.017	0.043	**0.426**	0.078
Hotel	0.020	**0.029**	0.003	0.054	0.362	**0.029**	0.315	0.040	0.015	0.038	**0.376**	0.081
Studyroom	0.020	0.043	0.007	0.058	0.233	**0.029**	0.372	0.041	0.012	0.055	**0.383**	0.079
Lab	0.026	**0.017**	0	n/a	0.247	0.022	0.337	0.035	0.020	0.036	**0.396**	0.072
All	0.020	0.031	0.007	0.043	0.325	**0.026**	0.340	0.040	0.016	0.043	**0.398**	0.078

dataset and the latter is an outdoor dataset. To facilitate fair comparison between all methods (and to facilitate training with mini-batch), all input point clouds are normalized to contain 50,000 points, by random upsampling or downsampling. For real applications, the upsampling or downsampling is not a necessary procedure. Moreover, for the matching task, it is not the denser the point cloud is, the better the result will be. For many SLAM or localization algorithm, they downsample the point cloud first before matching or feature extraction [43], [44].

2) *Metrics:* In previous work considering learning based methods for matching point clouds [10], [11], the authors presented matching recall as a metric for the network's performance. However, the standard used seems relatively loose, *i.e.*, if 5% of the patch pairs of a point cloud are correctly matched, it counts as a success to the recall. We focus more on the point cloud registration task but not only on point feature matching, the precision and robustness of the registration are reported in this study, as they are important metrics for registration. The registration recall is defined as:

$$R = \frac{1}{M}\sum_{i=1}^{M}\mathbb{1}\Big(\prod_{j=1}^{6}\mathbb{1}(\| \mathbf{T}_{ij} - \mathbf{T}^{*}_{ij} \|) \leq \tau_j\Big) \tag{5}$$

where M is the total number of point cloud pairs that could be matched, $\mathbf{T}$ and $\mathbf{T}^*$ are the estimated and ground truth transformation parameters respectively, each of them has six values, *i.e.*, rotation and translation along x, y and z axis, τ_j is the j-th threshold. The thresholds are set according to $\tau = [0.10, 0.10, 0.10, 0.20, 0.20, 0.20]$. Registration precision is defined as the average error over all six degrees of freedom of the estimated registrations over all point cloud pairs that pass the threshold τ (lower is better).

3) *Comparison to State-of-the-Art:* Quantitative results on different 3DMatch scenes are provided in Table I. In this paper, we select four types of methods to compare with, each of which is a reference method in its own category, *i.e.*, ICP [17] in point-wise registration, CPD [25] in distribution based registration, FPFH [4] in handcrafted feature based registration, and the state-of-the-art method 3DSmoothNet [11], PointNetLK [38] in learning based registration. For fairness of comparison, we assume that we do not have prior knowledge about the registration, thus randomly sampling 1024 points for the 3DSmoothNet. From Table I, we can see that our proposed DPFNet outperforms all the other four methods in terms of recall. For precision though, the proposed method is not the best, but the differences are marginal. This indicates that if matching is successful, the precision of the estimated transformation is also successful. Recall curves of all methods under various thresholds are shown in Fig. 4, from which we can see that DPFNet performs best under all thresholds. The PointNetLK performs much worse than the other two deep learning-based methods, this is because the generalization of PointNetLK is not good. If the training data and testing data are not the same model, its performance degenerates largely. Moreover, since we do not have classification labels for the 3DMatch dataset, a pre-training is not adopted here. An illustration of typical registration results by various methods are shown in Fig. 5.

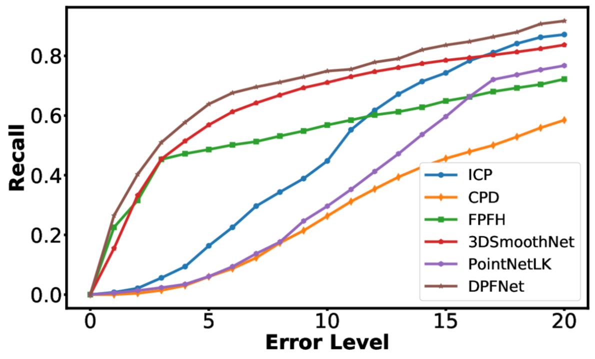


Fig. 4. Recall curve of different methods, x-axis is error threshold with $X \times [0.05, 0.05, 0.05, 0.10, 0.10, 0.10]$, where $X = 0, 1, 2, \ldots, 20$.

To demonstrate the generalizability, we also test on the KITTI dataset, while the networks are trained on the 3DMatch dataset. These two datasets are very different, as the 3DMatch dataset is collected indoor and the KITTI dataset is collected outdoor for autonomous driving applications. The goal is to verify the network's invariance to the specifics of the scene on which it has been trained. Results are provided in Table II and it can be observed that the performance of our approach degrades but is still comparable to that of some non-learning based methods, which are invariant to scene specifics by design. However, the performance of 3DSmoothNet, the state-of-the-art in learning-based approaches, collapses completely. The result on 3DMatch and KITTI show that our DPFNet is an important step towards accurate, robust, and generally applicable network-based point cloud registration. To further verify the potential application of the proposed algorithm for the self-driving task, we plan to build a new dataset which has the point cloud generated by the Velodyne VLP-16 LiDAR and ground-truth pose by RTK-GPS. We leave it our future work

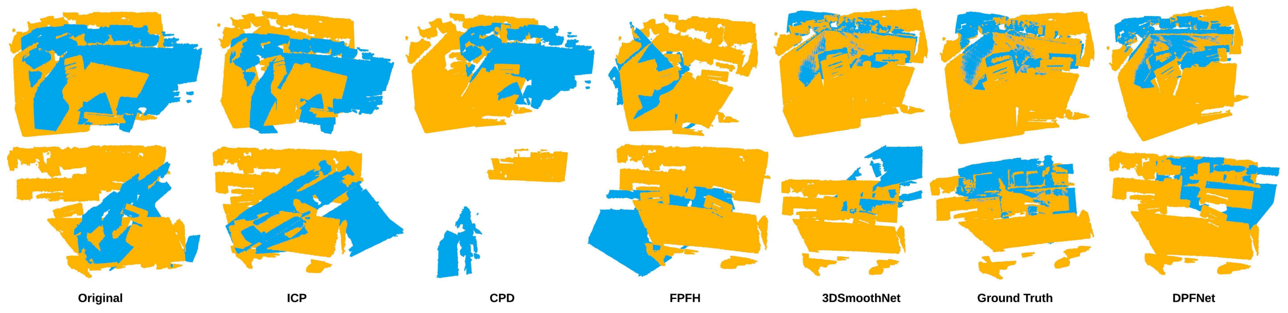


Fig. 5. An illustration of registration results by different methods. The first row is an example for which both 3DSmoothNet and DPFNet match the point clouds successfully. In the second row, only DFPNet matches the point clouds successfully.

TABLE II
REGISTRATION RECALL AND ERROR OF DIFFERENT METHODS ON KITTI DATASET

Method	ICP		CPD		FPFH		3DSmoothNet		PointNetLK		DPFNet(ours)	
	recall	error	recall	error	recall	error	recall	error	recall	error	recall	error
Sequence 0	**0.44**	**0.029**	0.20	0.041	0.20	0.033	0	n/a	0	n/a	0.21	0.102
Sequence 1	0	n/a	0.05	0.052	0.01	**0.013**	0	n/a	0	n/a	**0.16**	0.082
Sequence 2	0.11	0.039	0.13	0.040	0.02	**0.028**	0	n/a	0	n/a	**0.20**	0.143
Sequence 3	**0.40**	**0.031**	0.35	0.043	0.07	0.042	0	n/a	0	n/a	0.23	0.098
Sequence 4	**0.46**	0.036	0.37	0.040	**0.11**	**0.023**	0	n/a	0	n/a	0.18	0.052
Sequence 5	**0.42**	**0.027**	0.28	0.036	0.05	0.030	0	n/a	0	n/a	0.26	0.112
All	**0.29**	**0.030**	0.19	0.040	0.09	0.033	0	n/a	0	n/a	0.21	0.116

TABLE III
REGISTRATION RESULTS THAT ARE TRAINED AND TESTED BOTH ON KITTI DATASET

Method	3DSmoothNet		PointNetLK		DPFNet	
	recall	error	recall	error	recall	error
Sequence 0	0.36	0.05	0.06	**0.03**	**0.49**	0.06
Sequence 1	0.34	0.03	0.05	**0.03**	**0.51**	0.08
Sequence 2	0.35	0.06	0.11	**0.05**	**0.44**	0.07
Sequence 3	0.40	**0.03**	0.08	0.04	**0.53**	0.05
Sequence 4	0.30	0.07	0.04	**0.02**	**0.41**	0.10
Sequence 5	0.35	**0.05**	0.08	0.06	**0.43**	0.06

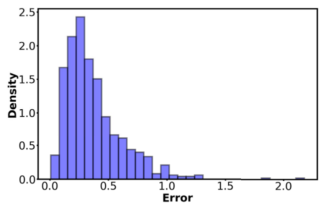
TABLE IV
REGISTRATION RESULTS WITH RESPECT TO INITIAL GUESS

Method	original	15°	30°	60°	90°	120°	180°
ICP	**52%**	45%	34%	5%	0	0	0
CPD	38%	38%	26%	17%	3%	0	0
FPFH	27%	28%	27%	25%	22%	20%	18%
3DSmoothNet	39%	39%	39%	36%	37%	35%	34%
PointNetLK	15%	15%	15%	15%	14%	14%	14%
DPFNet	46%	**46%**	**46%**	**46%**	**44%**	**45%**	**43%**

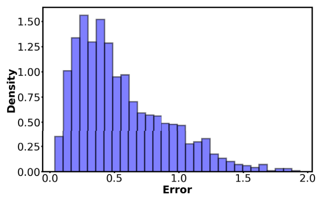
to give more results that the proposed method used as the data association algorithm for the LiDAR-based SLAM.

To further investigate how the proposed method and baselines perform on the outdoor dataset, which is especially important for intelligent vehicles related applications, we also train the networks using KITTI dataset. We split 80% of every sequence for training, and the remaining 20% for testing. The results are presented in Table III. As the training only affects deep learning related methods, we only report 3DSmoothNet, PointNetLK and DPFNet in Table III. For results of the non-learning methods, please see Table II. Compared with the results in Table II, there is an significant improvement for the learning-based methods when training and testing using the same type of data. Moreover, the proposed method DPFNet performs the best for all the sequences in terms of recall. The results here is also consistent with that tested on the indoor dataset presented in Table I.

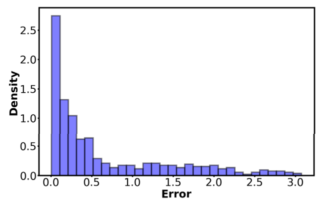
Another important metric for point cloud registration, especially for autonomous driving, is the robustness to initial guess. One of the benefits of the proposed method is that it is not affected by initial guess. Even if the point cloud rotates or translates a lot, the feature of every patch does not change as long as the shape does not deform. As indicated in [25], the translational error in initial guess is relatively easy to solve by aligning the center, *etc*. However, the robustness to poor initial guess in rotation is still an open problem. Due to this, we only investigate the performance of various methods with respect to rotational initial guess in this paper. We test all the methods on KITTI dataset which is more relevant to autonomous driving. We select 100 point cloud pairs from the testing set randomly, and rotate one of the point cloud (source) from 0°, 15°, until to 180°. Then, match the rotated source to the unchanged target point cloud. The results are presented in Table IV. Here we show the percentage of successful registration, and the define of successful registration is the same as equation (5). From the results, we can see that the learning-based methods are barely affected by the rotation. While the successful rate of ICP decreases gradually until no error within the given bound. CPD is more robust than ICP but is still significantly affected after the error of initial guess is beyond 60°. The hand-crafted



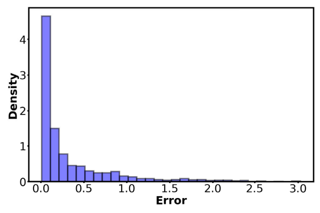
(a) ICP



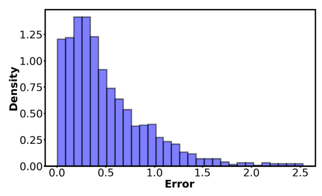
(b) CPD



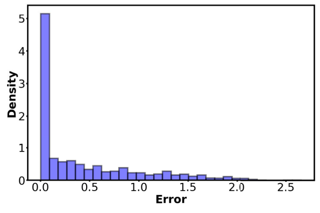
(c) FPFH



(d) 3DSmoothNet



(e) PointNetLK



(f) DPFNet

Fig. 6. Error histogram of different methods.

feature FPFH is relatively stable but the successful rate of it is not as good as 3DSmoothNet or DPFNet. Moreover, we highlight the proposed method DPFNet performs the best except the original case among all the methods.

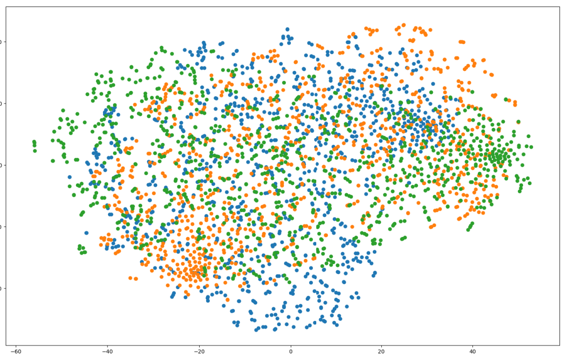
B. Registration Error Distribution

We present registration error distributions of different methods on 3DMatch dataset with the form of histogram shown in Fig. 6. From the figure, we can see that FPFH, 3DSmoothNet and DPFNet perform better than ICP and CPD. Especially DPFNet, most of its errors concentrate in small domain, which indicates its robustness for point set registration.

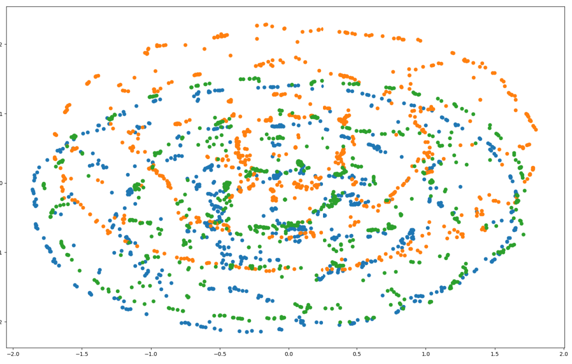
We present more registration results on 3DMatch dataset (Fig. 8) and KITTI dataset (Fig. 9). In Fig. 8, the first four rows display examples of successful registration, from which we can see that good performance of the proposed method even in some complex situations. The last two rows of Fig. 8 show some failure examples. For failure cases, the overlapping part of input point clouds are too small or features in the overlapping part are not remarkable (*e.g.*, plane) and it is hard for the network to extract distinguished features. In Fig. 9, the first five rows also show examples of successful registration. We can see that the proposed method is able to register point clouds of common roads, intersections, *etc*. The last two rows show failure examples on KITTI dataset. One reason of failure is that there is no good feature in these point clouds, of which most parts are road surface without buildings, facilities around it. Another reason is that these data are coarse and noisy.

C. Feature Distribution

We visualize embeddings learned by 3DSmoothNet and DPFNet using t-SNE [45] in Fig. 7. Here, we randomly pick



(a) 3DSmoothNet



(b) DPFNet

Fig. 7. T-SNE visualization of features extracted by (a) 3DSmoothNet and (b) DPFNet. Features of matched point clouds are denoted in blue and green, features of non-matched point cloud are denoted in orange color.

two point clouds that can be matched and one point cloud that does not match them in the kitchen scene of 3DMatch dataset. In the figure, embeddings of matched point clouds are denoted in blue and green, embeddings of non-matched point cloud are denoted in orange. The ideal representation would be that most parts of blue and green dots are near to each other as they share common parts of the scene and they do have their own unique parts. Most of the orange dots do not match the blue and green as they represent different scenes while some of the embeddings may represent similar spatial structures. From Fig. 7 (b), we can see that most blue dots have their own corresponding dots in the green set in terms of Euclidean distance, and the orange dots which move far from the two are not close to both of them. Besides, embeddings in one point cloud are also distinguished, which could reflect different spatial structures. From Fig. 7 (a), we can see that embeddings extracted by 3DSmoothNet are cluttered. By comparing Fig. 7 (a) and Fig. 7 (b), we can also see that embeddings extracted by 3DSmoothNet are not as distinguished as the embeddings extracted by DPFNet. Distributions of them are disordered in the space. Thus, in terms of Fig. 7, feature distribution of DPFNet outperforms that of 3DSmoothNet.

Here we also report the recall in terms of correspondence as that in 3DSmoothNet [11]. Recall of correspondence is defined as following equation:

$$R_c = \frac{1}{|\mathcal{P}|}\sum_{p=1}^{|\mathcal{P}|}\mathbb{1}\left(\left[\frac{1}{m_p}\sum_{i=1}^{m_p}\mathbb{1}\left(\| \; p_i - T\left(q_i\right) \|_2 < \tau_1\right)\right] > \tau_2\right) \tag{6}$$

TABLE V

ABLATION STUDY OF DPFNET ON 3DMATCH IN TERMS OF LOSS FUNCTION AND STRUCTURE CONVOLUTION

	Loss function		Convolution				Ours (all together)
	Triplet	BH	W/o sorted CNN	W/o weighted pooling	W/o both	W/o PointSIFT	
Recall	0.32	0.31	0.04	0.27	0	0.18	**0.40**
Error	0.074	0.079	**0.051**	0.083	n/a	0.092	0.078

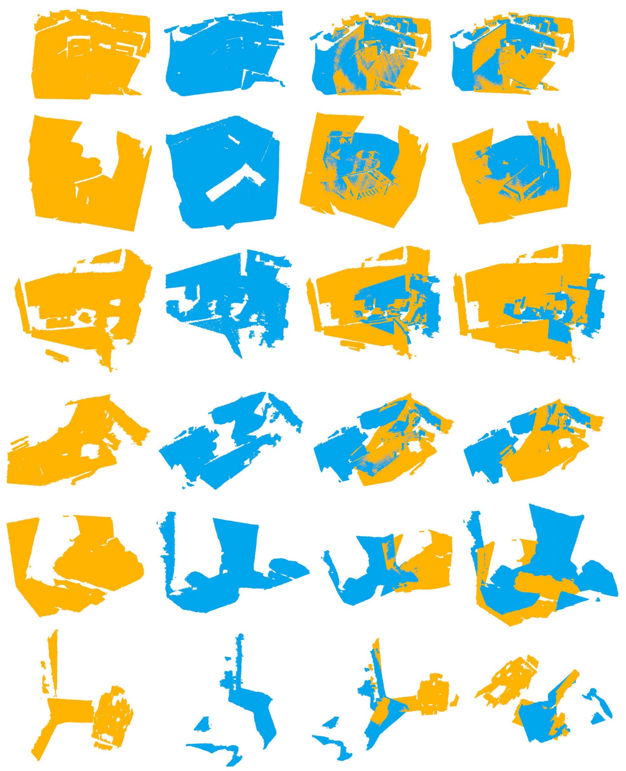


Fig. 8. Some registration results by DPFNet on 3DMatch dataset.

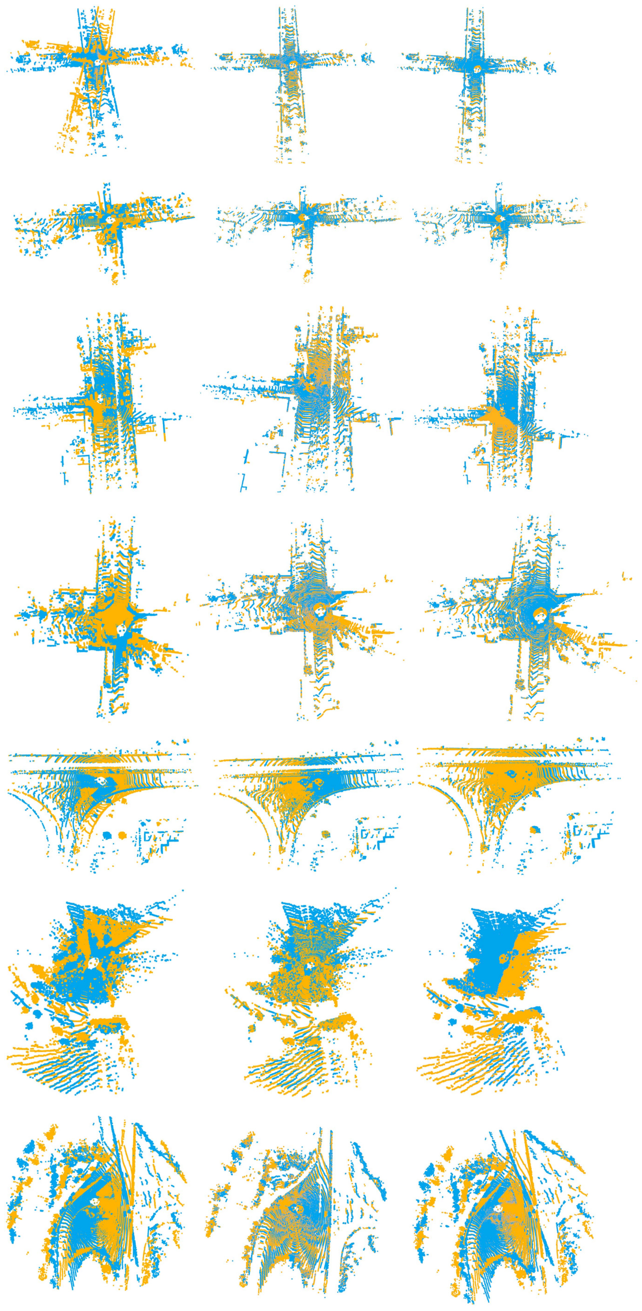


Fig. 9. Some registration results by DPFNet on KITTI dataset. The first to third column are the input point clouds, the matching by ground-truth pose, and matching by the proposed method respectively.

where $|\mathcal{P}|$ is the number of point set pairs to be matched, m_p is the number of corresponding point pairs after trimming of p-th matched point set pair as that defined in Section III-D. Following [10], [11], the threshold in this paper is set as $\tau_1 = 0.1m$, $\tau_2 = 0.05$. Results are shown in Table VI, where we just report the results for FPFH [4], SHOT [5], 3DMatch [41], CGF [46], PPFNet [9], PPF-FoldNet [10], 3DsmoothNet [11] as that in [11] and add the results of the proposed DPFNet. Though the result of our method is not as good as 3DMatchNet but is better than other methods, the gap between DPFNet and 3DSmoothNet is not that large.

D. Computation Time

The average computation time of different methods on 3DMatch dataset is presented in Table VII. ICP is the fastest among all the five methods with $0.33s$ on average for one registration. CPD is the slowest with $2386.25s$ for one registration averagely, which is much slower than other methods. This is partly due to its high runtime complexity of GMM modeling and coherent optimization. The computational time of PointNetLK is much bigger than that of ICP, but much smaller than that of CPD. It is similar to the FPFH method, both are several seconds. For the other two learning-based

TABLE VI

COMPARISON OF RECALL OF CORRESPONDENCE WITH THE POINT CLOUD FEATURE EXTRACTION METHODS ON 3DMATCH DATASET. HERE, 'O' DENOTES ORIGINAL DATASET, 'R' DENOTES ROTATED DATASET

		FPFH	SHOT	3DMatch	CGF	PPFNet	PPF-FoldNet	3DSmoothNet	DPFNet
O	AVG	54.3	73.3	57.3	58.2	62.3	71.8	**94.7**	78.6
	STD	11.8	7.7	7.8	14.2	11.5	9.9	**2.7**	9.3
R	AVG	54.8	73.3	3.6	58.5	0.3	73.1	**94.9**	76.5
	STD	12.1	7.6	**1.7**	14.0	0.5	11.1	2.5	7.5

TABLE VII

AVERAGE COMPUTATION TIME OF DIFFERENT METHODS ON 3DMATCH DATASET. FE DENOTES FEATURE EXTRACTION

Method	ICP	CPD	FPFH	PointNetLK	3DSmoothNet			DPFNet (Ours)	
					LRF+SDV	FE	RANSAC	FE	RANSAC
AVG. time	**0.33s**	2386.25s	3.45s	4.76s	5.96s	0.36s	1.82s	1.20s	0.03s

methods, we present computation time in different procedures for a better clarification. 3DSmoothNet-based registration consists of three parts: local reference frame transformation and smoothed density value voxelization (we denote them by LRF + SDV) which is pre-processing procedure for learning, feature extraction (FE) procedure and finally RANSAC based registration. From the table we can see that pre-processing consumes much of time (72.8%) of one registration. DPFNet-based registration consists of two parts: feature extraction and RANSAC based registration. Our method do not need pre-process, one registration needs $1.23s$ in total compared with that of $8.04s$ of 3DSmoothNet. The computational time of the proposed method (1.23s) cannot fulfill demand for tasks that need to be in real time. However, it still makes sense to improve efficiency in lots of offline tasks such as HD-map generation for autonomous driving, or tasks do not need to be real-time such as loop closure detection in SLAM. We use the proposed method as map matching algorithm, where the map is pre-built based on RTK-GPS. It is processed offline to better evaluate accuracy of the proposed method. Error of localization is within 0.36m in a 3km run.

E. Ablation Study

For a better understanding of the effects that the proposed network modules have on the overall network performance, we test performance with or without including a specific module. First, we test the effect of our novel structure-aware loss function. We present registration recall in three circumstances: 1) a triplet-loss with one corresponding example in the batch and all the other non-corresponding, 2) batch hard (BH) loss function [11] with one corresponding example and one hardest non-corresponding example, and 3) our proposed structure-based loss function. The results are presented in Table V and it can be observed that the proposed structure-aware loss function plays an important role, as not using it degrades recall by nearly 10%. We also test effects of our novel sorted CNN and the structure weighted pooling. We test performance of the network under four situations: 1) only with sorted CNN, 2) only with structure weighted pooling, 3) with both of them, and 4) only with max-pooling. The results in Table V clearly show that the combination of sorted CNN and weighted pooling has significant impact on registration, especially the sorted CNN, without which registration almost fails totally. These ablation experiments demonstrate the relevance of the proposed novel point cloud processing methods that form the backbone of our DPFNet approach.

V. CONCLUSION

We presented a direct deep learning-based point cloud registration method: DPFNet. Experimental results on indoor and outdoor benchmark datasets demonstrate that DPFNet outperforms existing learning-based methods in terms of recall and generalizability to unseen data. Ablation studies show that the key to the success of DPFNet are its structure-aware loss function, to deal with the loose point pairs brought by FPS point sampling, and its sorted convolution operator, that prevents having to sample patches in the point cloud before feeding it into the network. Both allow putting the point cloud directly into the network and thereby lower the risk of loosing important information in pre-processing steps, as done by alternative methods. For future work, we aim to improve on the FPS point sampling method by automatically learning to select salient points that are stable across different point clouds.

REFERENCES

[1] B. Li, L. Yang, J. Xiao, R. Valde, M. Wrenn, and J. Leflar, "Collaborative mapping and autonomous parking for multi-story parking garage," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1629–1639, May 2018.

[2] H. Lategahn and C. Stiller, "Vision-only localization," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1246–1257, Jun. 2014.

[3] H. Luo *et al.*, "Patch-based semantic labeling of road scene using colorized mobile LiDAR point clouds," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1286–1297, May 2015.

[4] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3212–3217.

[5] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 356–369.

[6] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao, "TOLDI: An effective and robust approach for 3D local shape description," *Pattern Recognit.*, vol. 65, pp. 175–187, May 2017.

[7] Y. Sun and M. A. Abidi, "Surface matching by 3D point's fingerprint," in *Proc. 8th IEEE Int. Conf. Comput. Vis.*, Jul. 2001, pp. 263–269.

[8] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *Int. J. Comput. Vis.*, vol. 105, no. 1, pp. 63–86, 2013.

[9] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 195–205.

[10] H. Deng, T. Birdal, and S. Ilic, "PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors," in *Proc. Eur. Conf. Comp. Vis.*, 2018, pp. 602–618.

[11] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3D point cloud matching with smoothed densities," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5545–5554.

[12] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2790–2799.

[13] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7203–7212.

[14] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3D point set upsampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5958–5967.

[15] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4087–4100, Aug. 2019.

[16] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "PF-Net: Point fractal network for 3D point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7662–7670.

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

[18] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, Nov. 2016.

[19] G. C. Sharp, S. W. Lee, and D. K. Wehe, "ICP registration using invariant features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 90–102, Jan. 2002.

[20] D. Svirko, P. Krsek, D. Stepanov, and D. Chetverikov, "The trimmed iterative closest point algorithm," in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2002, pp. 545–548.

[21] S. Granger and X. Pennec, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2002, pp. 418–432.

[22] L. Li, M. Yang, C. Wang, and B. Wang, "Rigid point set registration based on cubature Kalman filter and its application in intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1754–1765, Jun. 2018.

[23] R. Sandhu, S. Dambreville, and A. Tannenbaum, "Point set registration via particle filtering and stochastic dynamics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1459–1473, Aug. 2010.

[24] A. Myronenko, X. Song, and M. A. Carreira-Perpinán, "Non-rigid point set registration: Coherent point drift," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1009–1016.

[25] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.

[26] B. Jian and B. C. Vemuri, "Robust point set registration using Gaussian mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1633–1645, Aug. 2011.

[27] Y. T. Sin and T. Kanade, "A correlation-based approach to robust point set registration," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2004, pp. 558–569.

[28] D. Gerogiannis, C. Nikou, and A. Likas, "Robust image registration using mixtures of t-distributions," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[29] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1627–1644, Dec. 2013.

[30] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 66–89, 2016.

[31] L. Kiforenko, B. Drost, F. Tombari, N. Krüger, and A. G. Buch, "A performance evaluation of point pair features," *Comput. Vis. Image Understand.*, vol. 166, pp. 66–80, Jan. 2017.

[32] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660.

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[34] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation," 2018, *arXiv:1807.00652*.

[35] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on x-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.

[36] Z. J. Yew and G. H. Lee, "3DFeat-Net: Weakly supervised local 3D features for point cloud registration," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2018, pp. 630–646.

[37] Y. Tian, B. Fan, and F. Wu, "L2-Net: Deep learning of discriminative patch descriptor in Euclidean space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 661–669.

[38] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK: Robust & efficient point cloud registration using PointNet," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7163–7172.

[39] S. Wu, H. Huang, and M. Gong, "Deep points consolidation," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–13, 2015.

[40] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1150–1157.

[41] A. Zeng, S. Song, M. Niessner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1802–1811.

[42] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[43] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4758–4765.

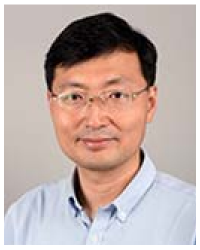
[44] L. Li, M. Yang, L. Guo, C. Wang, and B. Wang, "Hierarchical neighborhood based precise localization for intelligent vehicles in urban environments," *IEEE Trans. Intell. Veh.*, vol. 1, no. 3, pp. 220–229, Sep. 2016.

[45] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[46] M. Khoury, Q.-Y. Zhou, and V. Koltun, "Learning compact geometric features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 153–161.

Liang Li (Member, IEEE) received the bachelor's degree in automation from the Harbin Institute of Technology, Harbin, China, in 2013, and the Ph.D. degree in control science from Shanghai Jiao Tong University, Shanghai, China, in 2018.

His current research interests include high-precision localization for intelligent vehicles, computer vision, and pattern recognition. He is especially interested in outdoor high-precision localization for intelligent vehicles based on laser scanner and computer vision methods.



Ming Yang (Member, IEEE) received the master's and Ph.D. degrees from Tsinghua University, Beijing, China, in 1999 and 2003, respectively.

He is currently the Director of the Research Institute of Intelligent Vehicles Technology and a Full Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. He has been working in the field of intelligent vehicles for more than 20 years and participated in several related research projects, such as the THMR-V Project (first intelligent vehicle in China), European CyberCars and CyberMove projects, CyberC3 project, CyberCars-2 project, ITER transfer cask project, and AGV.