End-to-End Learning Local Multi-view Descriptors for 3D Point Clouds

Lei Li\(^1\)\footnote{L. Li was an intern at Alibaba A.I. Labs.} Siyu Zhu\(^2\) Hongbo Fu\(^3\)\footnote{H. Fu is the corresponding author. E-mail: hongbofu@cityu.edu.hk} Ping Tan\(^2,4\) Chiew-Lan Tai\(^1\)

\(^1\)HKUST \(^2\)Alibaba A.I. Labs \(^3\)City University of Hong Kong \(^4\)Simon Fraser University

Abstract

In this work, we propose an end-to-end framework to learn local multi-view descriptors for 3D point clouds. To adopt a similar multi-view representation, existing studies use hand-crafted viewpoints for rendering in a preprocessing stage, which is detached from the subsequent descriptor learning stage. In our framework, we integrate the multi-view rendering into neural networks by using a differentiable renderer, which allows the viewpoints to be optimizable parameters for capturing more informative local context of interest points. To obtain discriminative descriptors, we also design a soft-view pooling module to attentively fuse convolutional features across views. Extensive experiments on existing 3D registration benchmarks show that our method outperforms existing local descriptors both quantitatively and qualitatively.

1. Introduction

Local descriptors for 3D geometry are widely recognized as one of the cornerstones in many computer vision and graphics tasks, such as correspondence establishment, registration, segmentation, retrieval, etc. Particularly, with the prevalence of consumer-level RGB-D sensors, voluminous scanned data requires robust local descriptors for scene alignment and reconstruction [59, 4]. Such 3D data, however, is often noisy and incomplete, presenting challenges to the design of local descriptors.

Existing hand-engineered local descriptors [20, 11, 46, 45, 53, 52, 48], proposed in the past few decades, are mostly built upon histograms of low-level 3D geometric properties. Recent trends with deep neural networks have motivated researchers to develop learning-based local descriptors in a data-driven manner [66, 8, 24, 19, 6, 57, 12]. Several types of input representations for 3D local geometry have been explored, such as raw point cloud patches [24, 6], voxel grids [66, 12] and multi-view images [19, 67]. Currently, on the geometric registration benchmark of 3DMatch [66], most learning-based methods are built upon either PointNet [41] with point cloud patches or 3D CNNs with voxel grids, and 3DSmoothNet [12] achieves the state-of-the-art performance with smoothed density value voxelization. Despite the impressive progress made by the voxel representation, literature on 3D shape recognition and retrieval [50, 42, 56] indicates superior performance of multi-view images than voxel grids, and some initial attempts [19, 67] have been made to extend a similar idea to 3D local descriptors. Meanwhile, a line of recent studies has advanced 2D CNNs in learning local descriptors from a single image patch [15, 51, 34, 63, 23, 35, 32]. These motivate us to perform further investigation into a multi-view representation for 3D points and their local geometry.

The main challenges of adopting the multi-view representation in learning descriptors are as follows. First, to obtain multi-view images, a set of viewpoints (virtual cameras) are needed for 3D graphics rendering pipelines in a preprocessing stage [50, 19]. In existing studies [50, 42, 19, 56, 9, 17], the viewpoints are either randomly sampled or heuristically hand-picked. However, how to determine the viewpoints in a data-driven manner to produce more informative renderings for neural networks still remains a question. Second, an effective fusion operation is required to integrate features from multiple views into a single compact descriptor. Max-view pooling is a dominant fusion approach [50, 42, 19, 56], but this operation might overlook subtle details [67, 56], leading to sub-optimal performance.

In this work, we propose a novel network architecture that learns local multi-view descriptors for 3D point clouds in an end-to-end manner, as illustrated in Fig. 1. Our network consists of three main stages: (1) multi-view rendering for a 3D point of interest of a point cloud; (2) feature extraction in each rendered view; and (3) feature fusion across the views. Specifically, we first use an in-network differentiable renderer [30] to project the 3D local geometry of a specific point as multi-view patches. Viewpoints used by the renderer are optimizable parameters during training. The renderer can back-propagate supervision signals from rendered pixels to the viewpoints, enabling joint optimization of the rendering stage with the other two stages. Next, to extract features in each rendered view, we leverage existing CNNs that are well matured in the task of learning...
single patch descriptors [51, 34]. Lastly, to fuse the features across all the views, we examine the gradient flow problem of max-view pooling [50] and then design a novel soft-view pooling module. The former only considers the strongest response across views for each position in feature maps, while in contrast, our design adaptively aggregates all the responses with attentive weights estimated by a sub-network. In the backward pass, our design allows supervision signals to better flow into each input view for optimization. The experiments conducted on the 3DMatch benchmark [66] shows that our method outperforms existing hand-crafted and learned descriptors, and is robust against rotation and point density as well.

Our contributions in this work are summarized as: (1) we propose a novel end-to-end framework for learning local multi-view descriptors of 3D point clouds, with the state-of-the-art performance; (2) the viewpoints are optimizable via in-network differentiable rendering; (3) a soft-view pooling module fuses features across views attentively with a better gradient flow. We will make our code publicly available.

2. Related Work

Hand-crafted 3D Local Descriptors. Over the past few decades, a large body of literature has investigated descriptors for encoding geometric information of local neighborhoods of 3D points. A full review is beyond the scope of this paper. Classic descriptors include, to name a few, Spin Image [20], 3D Shape Contexts [11], PFH [46], FPFH [45], SHOT [53], and Unique Shape Context [52]. These hand-crafted descriptors are mostly constructed from histograms of low-level geometric properties. Despite the progress made by these descriptors, they may fail to handle well the nuisances commonly observed in real scanned data, like noise, incompleteness, and low resolution [13].

Learned 3D Local Descriptors. With the recent success of deep neural networks [44], more attention has been shifted to developing learning-based 3D local descriptors [6, 24, 66, 12, 19]. In general, these methods fall into three categories according to input representations, including point cloud patches, voxel grids and multi-view images.

Point cloud patches are the most straightforward representation for local neighborhoods of points. PointNet, a seminal work done by Qi et al. [41], is specifically designed to handle the unstructured nature of point clouds. Studies like [6, 5, 61] build upon PointNet to learn descriptors for point cloud patches. There also exist PointNet-based works that learn local descriptors jointly with other tasks, such as keypoint detection [62] and pose prediction [7].

Voxel grids, used in works like 3DMatch [66] and 3DSmoothNet [12], are a common structured representation for 3D point clouds [33, 58, 42]. To reduce noise and boundary effects, Gojcic et al. [12] proposed to use smoothed density value voxelization in 3DSmoothNet. Their method achieves the state-of-the-art performance on the 3DMatch benchmark [66], substantially outperforming the aforementioned PointNet-based approaches [6, 5, 7].

Multi-view images have demonstrated better performance than voxel grids in the task of 3D shape recognition and retrieval [50, 42, 43], owing to their ability of delivering rich information of 3D geometry. Motivated by the success in global shape analysis, researchers have extended the multi-view representation to 3D local descriptor learning [19, 67]. Huang et al. [19] re-purposed the CNN architecture from [50, 26] to extract local descriptors of 3D shapes (e.g., airplanes or chairs) from multi-view images, which are rendered offline with clustered viewpoints. There exist studies like [8, 43] that use 2D filtering for in-network image generation from point clouds. In contrast, our work considers the viewpoints as optimizable parameters and performs multi-view rendering with a differentiable renderer [30] in neural networks.

To fuse view features into a single compact representation, max-view pooling is widely used owing to its computational efficiency and view-order invariance [50, 42, 56, 19, 43, 67], but it tends to overlook subtle details as discussed in [56, 67, 34, 65, 37]. Zhou et al. [67] proposed FuseRcnn, a residual-learning module for feature fusion, but their module is not view-order invariant and its number of parameters grows with the number of input views. Alternative approaches, such as feature aggregation with NetVLAD [2] and RNN [16], have also been explored, but excessive computation or view ordering is required. Differently, by analyzing the gradient flow of max-view pooling, we propose soft-view pooling that adaptively aggregates features with attentive weights in a view-order invariant manner.

Differentiable Rendering. The conventional 3D graphics rendering pipeline involves rasterization and visibility test, which are non-differentiable discretization operations with respect to the projected point coordinates and viewpoint-dependent depths [30]. Thus supervision signals cannot flow from the 2D image space to the 3D shape space, preventing the integration of this pipeline into neural networks for end-to-end learning. Recently researchers have designed several differentiable rendering frameworks [31, 21, 29, 28, 39, 64, 3, 30] that incorporate approximated gradient formulations for the discretization operations. Among them, Soft Rasterizer (SoftRas), a state-of-the-art differentiable renderer developed by Liu et al. [30], treats mesh rendering as a process of probabilistic aggregation of triangles. In this work, we modify SoftRas to extend its application to point cloud rendering and adopt a hard-forward soft-backward scheme.

3. Methodology

Given a 3D point cloud \( \mathcal{P} \), we aim at training a neural network \( f \) that can extract a discriminative local descriptor
for a point \( p \in \mathcal{P} \) in an end-to-end manner. To this end, we perform projective analysis on the local geometry of \( p \) by using a multi-view representation. Compared to point cloud patches or voxel grids, the multi-view representation can capture different levels of local context more easily [19, 42].

Our network \( f \) is comprised of three stages as shown in Fig. 1. First, the network \( f \) directly takes the point cloud \( \mathcal{P} \) and the point of interest \( p \) as inputs and employs SoftRas [30] to render the local neighborhood of \( p \) as multi-view patches (Sec. 3.1). Second, we extract convolutional feature maps from each rendered view patch through a lightweight 2D CNN (Sec. 3.2). Lastly, all the extracted view features are compactly fused together by a novel soft-view pooling module to obtain the local descriptor (Sec. 3.3). The three stages of \( f \) are jointly trained in an end-to-end manner such that descriptors of corresponding points that are geometrically and semantically similar are close to each other, while descriptors of non-corresponding points are distant to each other (Sec. 3.4).

3.1. Multi-view Rendering

Optimizable Viewpoints. Existing multi-view approaches select a set of rendering viewpoints according to certain rules, e.g., by clustering [19] or circling around a viewing center at a fixed step [50, 56, 9]. However, this view selection process is detached from the subsequent multi-view fusion stage, and thus might produce less representative parameters for the latter. SoftRas allows the viewpoints to be optimizable parameters, which can be jointly trained with other network parameters in later stages. To set up virtual cameras in a look-at manner [1], we define the viewpoint parameters as \( \{ c_k = (\theta_k, \phi_k, \rho_k, \mathbf{u}) \}_{k=1}^n \) using spherical coordinates, where \( n \) is the number of viewpoints. Each viewpoint \( c_k \) is represented by two angles \( \theta_k \) and \( \phi_k \), the distance \( \rho_k \) from the local origin and a consistent upright orientation \( \mathbf{u} \). Given the point of interest \( p \) as the origin, the local reference frame (LRF) for \( \{ c_k \} \) is defined as follows (Fig. 2): the \( z \)-axis is collinear to the normal of \( p \); the \( x \)-axis is the cross product of \( \mathbf{u} \) and the \( z \)-axis (a small perturbation to \( \mathbf{u} \) if the normal is parallel to \( \mathbf{u} \)); and the \( y \)-axis is the cross product of the \( z \)-axis and \( x \)-axis. We constrain \( \{ c_k \} \) to be within the hemisphere where the point normal resides (Sec. 3.4). To augment rotation invariance in the learned descriptors, we rotate each rendered view patch at 90-degree intervals [19] (i.e., 4 in-plane rotations) within the network. Thus, a set of \( 4n \) view patches are obtained through rendering as detailed next.

![Figure 1: An end-to-end network that learns local multi-view descriptors for point clouds. The network takes point clouds as input and performs in-network multi-view rendering with a differentiable renderer for points of interest. Feature maps are extracted individually from each view and fused together via a soft-view pooling module to obtain the final descriptors.](image1.png)

![Figure 2: Local spherical coordinates \((\theta_k, \phi_k, \rho_k)\) for a viewpoint \(c_k\).](image2.png)

Differentiable Rendering. To address the non-differentiable issue of the conventional 3D graphics rendering pipeline (Fig. 3-a), SoftRas treats mesh rendering as a process of probabilistic aggregation of triangles in 2D. To render the point cloud \( \mathcal{P} \) as view patches with \( \{ c_k \} \), one approach is to firstly transform \( \mathcal{P} \) to a mesh via surface reconstruction [22], which, however, is challenging to integrate into our end-to-end framework and may not handle noise well (e.g., in laser scans of outdoor scenes). Instead, we modify SoftRas to make it amenable to point cloud rendering (Fig. 3-b). We consider each point \( q_j \in \mathcal{P} \) as a sphere [19], whose radius can be a fixed value [19] or derived from the average distance between \( q_j \) and its local neighbors. After perspective projection with a specific viewpoint \( c_k \), the point \( q_j \) produces a probability map \( D_j \) that describes the probability of each output pixel being covered by \( q_j \) [30]. The \( i \)-th pixel in the rendering output \( I \) (of size 64 \( \times \) 64) is defined as

\[
I_i = \sum_j w(D_j, z_j)C_j + w_bC_b, \tag{1}
\]
where \( C_j \) is the rendered attribute (e.g., color or view-dependent depth) of \( q_j \), \( C_0 \) is a default background value, and \( z_j \) is the depth of \( q_j \). The weighting function \( w(\cdot) \) designed in [30] is biased to points that are closer to the camera and the \( j \)-th pixel, and \( \sum_j w(\cdot) + w_b = 1 \). Such a linear formulation in Eq. 1 approximates the rasterization and visibility test in the conventional rendering pipeline (Fig. 3), and it is naturally differentiable. Since input point clouds may lack color information, we use view-dependent depth as \( C_j \) [8, 60], which is invariant to illumination changes. We refer the interested reader to [30] for detailed implementations and discussions of \( D_j \) and \( w(\cdot) \).

![Figure 3](image)

Figure 3: Rendering pipelines for point clouds: (a) Conventional 3D graphics rendering; (b) Soft Rasterizer [30] extended to 3D point cloud rendering.

Although the differentiability of Eq. 1 makes it possible for in-network rendering, we observed artifacts, such as blurry pixels at regions with large depth discontinuity, in the rendering outputs (see Fig. 4). To mitigate the influence of artifacts on the subsequent feature extraction, we instead adopt a hard-forward soft-backward scheme for rendering point clouds with SoftRas, sharing a similar idea to [21]. Specifically, in the forward pass, we perform rasterization and visibility test to obtain rendering results in the same way as the conventional rendering pipeline (Fig. 3-a). In the backward pass, we compute approximated gradients for the rendering using Eq. 1 of SoftRas. We found that this approximation scheme works well in our experiments.

### 3.2. Feature Extraction

Let \( \{Z_k\}_{k=1}^{2n} \) be the set of multi-view patches produced in the rendering stage for the point \( p \). This 2D representation can naturally lend itself to existing patch analysis networks. We adopt a lightweight CNN backbone similar to L2-Net [51, 34], a state-of-the-art network for learning local image descriptors. Concretely, the network is composed of six stacked convolutional layers, each followed by normalization [54] and ReLU layers. We feed each patch \( Z_k \) to the network and obtain a corresponding feature map denoted as \( F_k \), which is of size \( 8 \times 8 \) with 128 channels.

### 3.3. Multi-view Fusion

Given the set of feature maps \( \{F_k\}_{k=1}^{2n} \) as input, we perform feature fusion across views to obtain a more compact multi-view representation. Let \( \tilde{F} \) denote the feature value at location \( i \) of the fused output \( \tilde{F} \) (the same size as \( F_k \)), and \( i \) iterates over all spatial and channel-wise positions (Fig. 5). Max-view pooling is a widely adopted fusion approach for its simple computation and invariance to view ordering. However, this operation suffers from the following gradient flow problem in back-propagation. Mathematically, max-view pooling can be expressed as

\[
\tilde{F} = \sum_k \alpha_k^i F_k,
\]

where \( \sum_k \alpha_k^i = 1 \) and the weights \( \{\alpha_k^i\} \) are in a one-hot form for selecting the maximum value. In the backward pass, the gradient of Eq. 2 is

\[
\frac{\partial \tilde{F}^i}{\partial F_k} = \begin{cases} 
1 & \text{if } F_k^i \text{ is the maximum value,} \\
0 & \text{otherwise.}
\end{cases}
\]

Thus, according to the chain rule, supervision signals from loss functions cannot flow into certain locations in \( F_k \) if the locations do not have the maximum feature values, which may guide CNNs to overlook some details in feature extraction. An alternative approach is average-view pooling with \( \alpha_k^i = \frac{1}{2n} \) to alleviate the gradient flow problem. However, as shown in existing studies [19], this approach performs worse than max-view pooling, partially because treating features equally across views may reduce the contribution of useful features while increasing the effect of insignificant features, leading to less discriminative descriptors.

Based on the above analysis, we propose soft-view pooling that adaptively estimates attentive weights \( \{\alpha_k^i\} \) with a sub-network. Specifically, the sub-network takes each \( F_k \) as input and follows an encoder-decoder design to regress the corresponding weights. The sub-network performs downsampling and then upsampling by a factor of 2 for both spatial size and channel depth, using a \( 3 \times 3 \) convolutional
layer and a $3 \times 3$ up-convolutional layer respectively, and a ReLU layer in-between. The output weight map is denoted as $\alpha_k$ (the same size as $F_k$). Afterward, for each location $i$ as defined above, the softmax function is applied to $\{\alpha_k^i\}$ for normalization so that $\sum_k \alpha_k^i = 1$ holds. Note that the above computation is invariant to view orders.

At last, the network $f$ embeds the fused feature $\tilde{F}$ to a $d$-dimensional descriptor space with a fully-connected layer and a subsequent $l2$ normalization layer.

### 3.4. Training

To train the network $f$, we sample matching point pairs in the overlapped region of two point clouds (at least 30% overlap). Given a batch of matching point pairs $B = \{(p_i, q_i)\}$, we follow [12, 18] to adopt a batch-hard (BH) triplet loss

$$L_{BH} = \frac{1}{|B|} \sum_{i=1}^{|B|} \left[ m + \|f(p_i) - f(q_i)\|_2^2 - \min_{j \neq i} \|f(p_i) - f(q_j)\|_2^2 \right]_+, \quad (4)$$

where $[.]_+ = \max(\cdot, 0)$, and $m$ is a margin and set to 1. For a training triplet, $q_i$ is the positive sample of $p_i$, and $L_{BH}$ considers the hardest negative sample $q_i$ within the batch $B$ for $p_i$. As mentioned in Sec. 3.1, we also impose range constraints for the optimizable viewpoints as follows:

$$L_{OV} = \frac{1}{n} \sum_{k=1}^n \sum_{x \in \{\theta_k, \phi_k, \rho_k\}} \left[ x - \frac{x_a + x_b}{2} \right]_+^2, \quad (5)$$

where $x_a = \{0, 0.3\}$ and $x_b = \{2\pi, \pi/2, 1\}$ for $\theta_k$, $\phi_k$ and $\rho_k$ respectively. Thus, the total loss is $L = L_{BH} + \lambda L_{OV}$, where $\lambda$ is empirically set to 1.

We implemented the network with PyTorch [38]. We set the viewpoint number $n = 8$ and the descriptor dimension $d = 32$ (Sec. 4.4). The viewpoint parameters $\theta_k$, $\phi_k$, and $\rho_k$ were initialized randomly within the range in Eq. 5, and $u$ was initialized to $[0, -1, 0]^T$. We use Adam [25] for stochastic gradient descent with $|B| = 24$ and an initial learning rate of 0.001. The network is trained for 16 epochs, and the learning rate is decayed by 0.1 every 4 epochs.

### 4. Experiments

#### 4.1. 3DMatch Benchmark

**Dataset.** We evaluate the proposed method on the widely adopted geometric registration benchmark from 3DMatch [66]. The benchmark consists of RGB-D scans of 62 indoor scenes, an ensemble of several existing RGB-D datasets [55, 49, 59, 27, 14]. The data is split into 54 scenes for training and validation, and 8 scenes for testing. In each scene, point cloud fragments are obtained by fusing 50 consecutive depth frames. For each fragment in the testing set, a set of 5,000 randomly sampled points is provided as keypoints for descriptor extraction.

**Metric.** The recall metric is used for comparisons on the testing set by averaging the number of matched point cloud fragments [6, 5, 12]. Consider a set of point cloud fragment pairs $\mathcal{G} = \{(\mathcal{P}, \mathcal{Q})\}$, where point clouds $\mathcal{P}$ and $\mathcal{Q}$ have at least 30% overlap after alignment. For a specific descriptor extraction method $g(\cdot)$, the set of putative matching points between $\mathcal{P}$ and $\mathcal{Q}$ is computed in the descriptor space as follows:

$$\mathcal{M} = \{ (p \in \mathcal{P}, q \in \mathcal{Q}) | g(p) = \text{nn}(q, g(\mathcal{P})) \land g(q) = \text{nn}(p, g(\mathcal{Q})) \}, \quad (6)$$

where $p$ and $q$ are keypoints and $\text{nn}(\cdot)$ is the nearest neighbor search. The recall metric $\mathcal{R}$ is then defined as follows:

$$\mathcal{R} = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \left[ \frac{1}{|\mathcal{M}_i|} \sum_{p, q \in \mathcal{M}_i} \| p - \mathcal{T}_i(q) \|_2 < \tau_1 \right] > \tau_2, \quad (7)$$

where $[\cdot]$ is the Iverson bracket, and $\mathcal{T}_i(\cdot)$ is the ground-truth transformation for aligning the $i$-th fragment pair in $\mathcal{G}$. The distance threshold $\tau_1$ for matching points is set to 10 cm. The inlier ratio $\tau_2$ ranges from 0.05 to 0.2. To reliably find correct alignment parameters between two overlapping point clouds, the number of RANSAC [10] iterations is 55,000 for $\tau_2 = 0.05$ and 860 for $\tau_2 = 0.2$ [6, 12].

### 4.2. Evaluation Results

Following [6, 5, 12], we compare our method (32-d) with several existing 3D local descriptors on the benchmark. For hand-crafted descriptors, FPFH [45] (33-d) and SHOT [53] (352-d) are tested, and their implementations come from PCL [47]. For learned descriptors, 3DMatch [66] (512-d), CGF [24] (32-d), PPFNet [6] (64-d), PPF-FoldNet [5] (512-d) and the current state-of-the-art 3DSmoothNet [12] (32-d) are tested. Additionally, we also compare with LMCVNN [19], a learned multi-view descriptor baseline using viewpoint clustering for offline rendering and max-view pooling for multi-view fusion. The original LMCVNN uses AlexNet [26] as its CNN backbone and outputs 128-d descriptors, but for fair comparisons, we reimplemented
LMVCNN with the same CNN backbone and descriptor dimensionality (32-d) as our method. We use the implementations and trained weights from the authors for 3DMatch, CGF and 3DSmoothNet. Since the implementations of PPFNet and PPF-FoldNet are not publicly accessible, we include their reported performance for completeness.

Table 1 shows the comparison results on the benchmark. For $\tau_2 = 0.05$, our method achieves an average recall of 97.5%, outperforming all the competing descriptors. Nevertheless, $\tau_2 = 0.05$ is a relatively loose threshold on 3DMatch, since 3DSmoothNet (95.0%), LMVCNN (96.5%) and our method all have achieved almost saturated performance with relatively small difference. Even so, our method obtains higher recalls in most testing scenes than 3DSmoothNet and LMVCNN. More notably, for a stricter condition $\tau_2 = 0.2$, there is significant improvement of our method over the other competitors. Specifically, our method maintains a high average recall of 86.9%, while 3DSmoothNet and LMVCNN drop to 72.9% and 81.0%, respectively.

The performance of FPFH, SHOT, 3DMatch, and CGF falls below 30%.

In Fig. 6, we plot the average recalls with respect to a range of $\tau_2$, illustrating the consistency of improvement brought by our method over the compared descriptors under different inlier ratio conditions. Additionally, Table 2 lists the average number of correct correspondences found by each descriptor, which is computed as $\frac{1}{|G|} \sum_{i=1}^{|G|} \sum_{\mathbf{p} \in M_i} \|[\mathbf{p} - T_i(\mathbf{q})]\|_2 < \tau_i$, using the same notations as in Eq. 7. It is observed that our multi-view descriptor is about 1.5× and 1.3× the average number of correspondences of 3DSmoothNet and LMVCNN, respectively. This clearly accounts for the dominant robustness of our descriptor. Additionally, Fig. 7 visualizes some point cloud registration results obtained by different descriptors with RANSAC. Particularly, it is observed that our descriptor is robust in the registration of fragments with large flat regions (the second row).

**Rotated 3DMatch Benchmark.** To evaluate the robustness of the descriptors against rotations, we construct a rotated 3DMatch benchmark [5, 12] by rotating the testing fragments with randomly sampled axes and angles in $[0, 2\pi]$. The keypoint indices of each fragment are kept unchanged. Table 3 gives the average recalls for each descriptor in the Rotated column. Our method achieves average recalls of 96.9% and 82.1% for $\tau_2 = 0.05$ and 0.2 respectively, both surpassing the performance of 3DSmoothNet (94.9% and 72.7%), LMVCNN (95.7% and 76.7%) as well as the other descriptors. The evaluation results indicate that our method can handle rotation well.

**Sparse 3DMatch Benchmark.** To evaluate the ro-
Figure 7: Geometric registration of point cloud 1 and point cloud 2 by different descriptors with RANSAC.

bustness of the descriptors against point density, we follow [5, 12] to construct a sparse 3DMatch benchmark. Concretely, for each testing fragment, the keypoints are firstly retained and then 50% or 25% of the remaining points are randomly selected. The evaluation results are shown in Table 3 (the last two columns). It is found that owing to the sphere-based rendering, our method is able to handle different point densities, like LMVCNN and 3DSmoothNet, and maintains the superior performance.

Table 3: Average recall (%) on a rotated or sparse 3DMatch benchmark with \( \tau_1 = 10\text{cm} \) and \( \tau_2 = 0.05 \) or 0.2.

<table>
<thead>
<tr>
<th></th>
<th>Rotated</th>
<th>Sparse (0.5)</th>
<th>Sparse (0.25)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPFH</td>
<td>60.1 10.0</td>
<td>59.2 9.5</td>
<td>57.8 8.5</td>
</tr>
<tr>
<td>SHOT</td>
<td>73.3 26.9</td>
<td>72.3 25.5</td>
<td>70.7 23.1</td>
</tr>
<tr>
<td>3DMatch</td>
<td>11.6 1.4</td>
<td>73.3 15.9</td>
<td></td>
</tr>
<tr>
<td>CGF</td>
<td>60.7 12.5</td>
<td>52.6 7.8</td>
<td>41.7 3.8</td>
</tr>
<tr>
<td>PPFNet</td>
<td>0.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PPF-FoldNet</td>
<td>73.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3DSmoothNet</td>
<td>94.9 72.7</td>
<td>94.4 71.7</td>
<td>94.8 70.1</td>
</tr>
<tr>
<td>LMVCNN</td>
<td>95.7 76.7</td>
<td>96.2 81.3</td>
<td>95.9 81.5</td>
</tr>
<tr>
<td>Ours</td>
<td>96.9 82.1</td>
<td>97.2 87.2</td>
<td>97.3 86.1</td>
</tr>
</tbody>
</table>

Table 4: Average running time (ms) per point on the 3DMatch benchmark.

<table>
<thead>
<tr>
<th></th>
<th>Input prep.</th>
<th>Inference</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DMatch</td>
<td>0.1</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>CGF</td>
<td>10.6</td>
<td>0.1</td>
<td>10.7</td>
</tr>
<tr>
<td>3DSmoothNet</td>
<td>39.4</td>
<td>0.2</td>
<td>39.6</td>
</tr>
<tr>
<td>Ours</td>
<td>7.2</td>
<td>1.5</td>
<td>8.7</td>
</tr>
</tbody>
</table>

Table 4 summarizes the running time for the learned descriptors on the standard 3DMatch benchmark. All the experiments were performed on a PC with an Intel Core i7 @ 3.6GHz, a 32GB RAM and an NVIDIA GTX 1080Ti GPU. The input preparation in Table 4 refers to voxelization with TDF [66] for 3DMatch, spherical histogram computation [24] for CGF, LRF computation and SDV voxelization [12] for 3DSmoothNet, and multi-view rendering (Sec. 3.1) for our method. The inference in Table 4 refers to descriptor extraction from the prepared inputs with neural networks. The results show that the input preparation stage dominates the running time of our method. Additionally, for sphere-based rendering (Sec. 3.1), it takes 0.16ms to determine a point radius by neighborhood query with FLANN [36] (used in our implementation), while alternatively the computation can be eschewed by using a fixed radius as in [19]. Nevertheless, our method still demonstrates competitive running time performance.

4.3. Generalization to Outdoor Scenes

We further evaluate the generalization ability of the descriptors on an outdoor-scene benchmark constructed by Gojcic et al. [12] with point clouds from the ETH dataset [40]. This benchmark consists of four scenes, including Gazebo-Summer, Gazebo-Winter, Wood-Summer and Wood-Autumn. The point clouds were obtained by a laser scanner and mostly about outdoor vegetation. Thus, the point clouds are in a large spatial range with a low resolution and contain complex and noisy local geometry. Identical to the 3DMatch benchmark, 5,000 keypoints are randomly sampled in each point cloud for descriptor extraction. The evaluation metric is the same as that in Sec. 4.1. Following [12], no fine-tuning is performed for the descriptors trained on the 3DMatch benchmark. To accommodate the low resolution and large spatial range of the point clouds, the voxel grids for 3DMatch and 3DSmoothNet are enlarged with longer edges (3\( \times \) and 5\( \times \) respectively) than those in Sec. 4.2. The radius of spherical histogram in CGF is 3.3\( \times \) longer. For LMVCNN and our method, the distance \( \rho_k \) in each viewpoint \( c_k \) is multiplied by a factor of 3.

The average recall results are shown in Table 5. Our method (79.9%) achieves comparable performance to 3DSmoothNet (79.0%). Meanwhile, our method significantly outperforms LMVCNN (39.7%) and SHOT (61.1%), and the other descriptors (including CGF, 3DMatch and FPFH) fall below 25%. To account for the deteriorated
performance of LMVCNN, further experiments on its used view selection and multi-view fusion strategies are performed in Sec. 4.4. The above results show that our method trained on the 3DMatch benchmark can generalize well to outdoor scenes.

<table>
<thead>
<tr>
<th></th>
<th>Gazebo</th>
<th>Wood</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPFH</td>
<td>40.2</td>
<td>15.2</td>
</tr>
<tr>
<td>SHOT</td>
<td>73.9</td>
<td>45.7</td>
</tr>
<tr>
<td>3DMatch</td>
<td>22.8</td>
<td>8.7</td>
</tr>
<tr>
<td>CGF</td>
<td>38.6</td>
<td>15.2</td>
</tr>
<tr>
<td>3DSmoothNet</td>
<td>91.3</td>
<td>84.1</td>
</tr>
<tr>
<td>LMVCNN</td>
<td>53.3</td>
<td>31.8</td>
</tr>
<tr>
<td>Ours</td>
<td>85.3</td>
<td>72.0</td>
</tr>
</tbody>
</table>

Table 5: Average recall (%) on the ETH benchmark with \( \tau_1 = 10 \text{cm} \) and \( \tau_2 = 0.05 \).

4.4. Ablation Study

**Descriptor Dimension & Viewpoint Number.** In Fig. 8 we plot the average recalls of our method with different descriptor dimensions \( d \) and viewpoint numbers \( n \) (as defined in Sec. 3.3 and Sec. 3.1). It is found that increased descriptor dimensions \( (d \geq 32) \) and viewpoint numbers \( (n \geq 8) \) lead to saturated performance. Thus we adopt \( d = 32 \) and \( n = 8 \) for our method in the experiments.

**Viewpoints.** In Table 6 (top), we show the performance of our network \( f \) trained with different viewpoint selection rules in multi-view rendering. Concretely, the straightforward random sampling rule places the viewpoints randomly within the range in Eq. 5. The viewpoint clustering rule used in LMVCNN [19] selects three representative viewing directions via K-medoids clustering. The orbited placement rule sets the viewpoints with \( \rho = 0.3 \), \( \phi = \pi/6 \), and \( \theta \) at a \( \pi/4 \) step (Sec. 3.1), similar to the strategy used in 3D shape recognition works [30, 56, 9]. The performance of \( f \) without rotation augmentation to the rendered view patches is also provided. It is found that our optimizable viewpoints produce better performance than these alternative view selection rules, especially on the generalization ability to the ETH outdoor dataset.

**Multi-view Fusion.** We perform experiments to compare our soft-view pooling with several alternative multi-view fusion approaches, including max-view pooling [19], Fuseption [67], and NetVLAD [2]. We list the performance of the network \( f \) trained with the above fusion approaches in Table 6 (bottom). While on the 3DMatch dataset the improvement of soft-view pooling is small compared with max-view pooling, our method shows significantly better generalization on the ETH outdoor dataset. This is partially because the low-resolution scans of outdoor vegetation in ETH would produce relatively noisy renderings, presenting challenges to max-view pooling for selecting the strongest feature response. Differently, the response is adaptively gathered in our method with attention.

![Average Recall (%) w.r.t descriptor dimension \( d \) and viewpoint number \( n \) on the 3DMatch benchmark.](image)

<table>
<thead>
<tr>
<th>( \tau_2 )</th>
<th>3DMatch</th>
<th>ETH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random sampling</td>
<td>97.0</td>
<td>84.1</td>
</tr>
<tr>
<td>Viewpoint clustering</td>
<td>96.7</td>
<td>83.5</td>
</tr>
<tr>
<td>Orbited placement</td>
<td>92.5</td>
<td>55.2</td>
</tr>
<tr>
<td>Ours w/o rotation augment.</td>
<td>96.9</td>
<td>83.6</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>97.5</strong></td>
<td><strong>86.9</strong></td>
</tr>
</tbody>
</table>

Table 6: Ablation study of viewpoint selection and multi-view fusion on the 3DMatch and ETH benchmarks.

5. Conclusion

We have presented a novel end-to-end framework for learning local multi-view descriptors of 3D point clouds. Our framework performs in-network multi-view rendering with optimizable viewpoints that can be jointly trained with later stages, and integrates convolutional features across views attentively via soft-view pooling. We demonstrate the superior performance of our method and its generalization to outdoor scenes through experiments. For future work, it is worth investigating the acceleration of differentiable multi-view rendering of point clouds and the extension of our framework to other tasks such as 3D object detection and recognition in point clouds.

Acknowledgements

This work was supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11212119, HKUST 16206819, HKUST 16213520), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.
References


