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

Lei Li[1*] Siyu Zhu[2] Hongbo Fu[3†] Ping Tan[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 [60, 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, 64, 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

*L. Li was an intern at Alibaba A.I. Labs.
†H. Fu is the corresponding author. E-mail: hongbofu@cityu.edu.hk

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, 62] 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 [63] 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, 59, 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 Fusception, 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 view-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, 58, 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

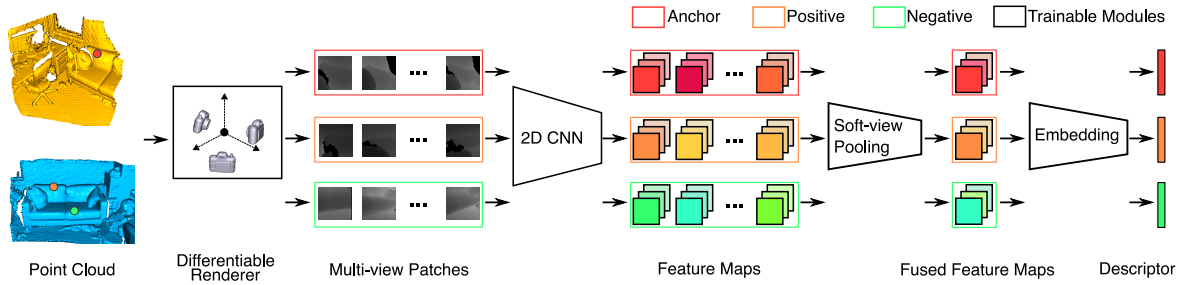

Figure 1: An end-to-end network that learns local multi-view descriptors for point clouds. The network takes point clouds as input and performs in-network multi-view rendering with a differentiable renderer for points of interest. Feature maps are extracted individually from each view and fused together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

for a point $\mathbf{p} \in \mathcal{P}$ in an end-to-end manner. To this end, we perform projective analysis on the local geometry of $\mathbf{p}$ 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 $\mathbf{p}$ as inputs and employs SoftRas [30] to render the local neighborhood of $\mathbf{p}$ 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 inputs 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 $\{\mathbf{c}_k = (\theta_k, \phi_k, \rho_k, \mathbf{u})\}_{k=1}^{n}$ using spherical coordinates, where n is the number of viewpoints. Each viewpoint $\mathbf{c}_k$ is represented by two angles θ_k and ϕ_k, the distance ρ_k from the local origin and a consistent upright orientation $\mathbf{u}$. Given the point of interest $\mathbf{p}$ as the origin, the local reference frame (LRF) for $\{\mathbf{c}_k\}$ is defined as follows (Fig. 2): the z-axis is collinear to the normal of $\mathbf{p}$; 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 $\{\mathbf{c}_k\}$ 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.

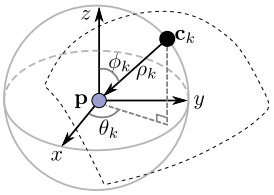

Figure 2: Local spherical coordinates (θ_k, ϕ_k, ρ_k) for a viewpoint $\mathbf{c}_k$.

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 $\{\mathbf{c}_k\}$, 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 $\mathbf{q}_j \in \mathcal{P}$ as a sphere [19], whose radius can be a fixed value [19] or derived from the average distance between $\mathbf{q}_j$ and its local neighbors. After perspective projection with a specific viewpoint $\mathbf{c}_k$, the point $\mathbf{q}_j$ produces a probability map $\mathcal{D}_j$ that describes the probability of each output pixel being covered by $\mathbf{q}_j$ [30]. The i-th pixel in the rendering output $\mathcal{I}$ (of size 64×64) is defined as

$$\mathcal{I}^i = \sum_j w(\mathcal{D}_j^i, z_j)\mathcal{C}_j + w_b\mathcal{C}_b, \tag{1}$$

where $\mathcal{C}_j$ is the rendered attribute (e.g., color or view-dependent depth) of $\mathbf{q}_j$, $\mathcal{C}_b$ is a default background value, and z_j is the depth of $\mathbf{q}_j$. The weighting function $w(\cdot)$ designed in [30] is biased to points that are closer to the camera and the i-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 $\mathcal{C}_j$ [8, 61], which is invariant to illumination changes. We refer the interested reader to [30] for detailed implementations and discussions of $\mathcal{D}_j$ and $w(\cdot)$.

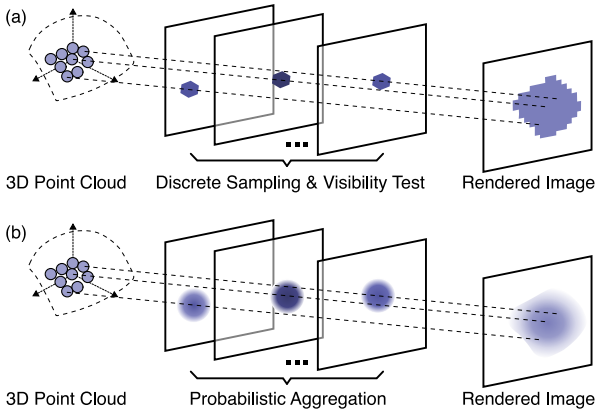

Figure 3: Rendering pipelines for point clouds: (a) Conventional 3D graphics rendering; (b) Soft Rasterizer [30] extended to 3D point cloud rendering.

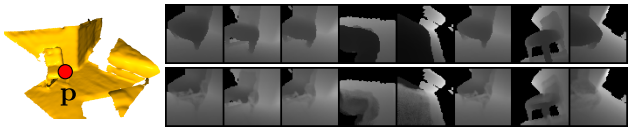

Figure 4: Multi-view rendering samples (depth, size = 64×64) for a point $\mathbf{p}$. Top: renderings of our hard-forward soft-backward scheme (Fig. 3-a); Bottom: renderings of Soft Rasterizer [30] (Fig. 3-b).

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 $\{\mathcal{I}_k\}_{k=1}^{4n}$ be the set of multi-view patches produced in the rendering stage for the point $\mathbf{p}$. 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 $\mathcal{I}_k$ to the network and obtain a corresponding feature map denoted as $\mathcal{F}_k$, which is of size 8×8 with 128 channels.

3.3. Multi-view Fusion

Given the set of feature maps $\{\mathcal{F}_k\}_{k=1}^{4n}$ as input, we perform feature fusion across views to obtain a more compact multi-view representation. Let $\tilde{\mathcal{F}}^i$ denote the feature value at location i of the fused output $\tilde{\mathcal{F}}$ (the same size as $\mathcal{F}_k$), 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

$$\tilde{\mathcal{F}}^i = \sum_k \alpha_k^i \mathcal{F}_k^i, \tag{2}$$

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

$$\frac{\partial \tilde{\mathcal{F}}^i}{\partial \mathcal{F}_k^i} = \begin{cases} 1 & \text{if } \mathcal{F}_k^i \text{ is the maximum value,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Thus, according to the chain rule, supervision signals from loss functions cannot flow into certain locations in $\mathcal{F}_k$ 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 $\alpha_k^i = \frac{1}{4n}$ 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 $\mathcal{F}_k$ 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 3×3 convolutional

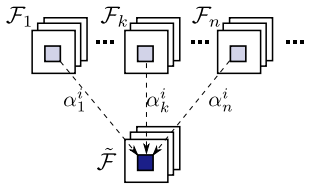

Figure 5: Multi-view fusion at location i that iterates over all spatial and channel-wise positions (top: feature maps of each view; bottom: fused feature maps).

layer and a 3×3 up-convolutional layer respectively, and a ReLU layer in-between. The output weight map is denoted as α_k (the same size as $\mathcal{F}_k$). 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 $\tilde{\mathcal{F}}$ 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 $\mathcal{B} = \{(\mathbf{p}_i, \mathbf{q}_i)\}$, we follow [12, 18] to adopt a batch-hard (BH) triplet loss

$$\mathcal{L}_{BH} = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \Big[m + \|f(\mathbf{p}_i) - f(\mathbf{q}_i)\|_2 - \min_{\substack{j=1\cdots|\mathcal{B}| \\ j \neq i}} \|f(\mathbf{p}_i) - f(\mathbf{q}_j)\|_2 \Big]_+, \tag{4}$$

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

$$\mathcal{L}_{OV} = \frac{1}{n} \sum_{k=1}^{n} \sum_{x \in \{\theta_k, \phi_k, \rho_k\}} [|x - \frac{x_a + x_b}{2}| - \frac{x_b - x_a}{2}]_+, \tag{5}$$

where $x_a = \{0,\ 0,\ 0.3\}$ and $x_b = \{2\pi,\ \pi/2,\ 1\}$ for θ_k, ϕ_k and ρ_k respectively. Thus, the total loss is $\mathcal{L} = \mathcal{L}_{BH} + \lambda \mathcal{L}_{OV}$, where λ 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 θ_k, ϕ_k, and ρ_k were initialized randomly within the range in Eq. 5, and $\mathbf{u}$ was initialized to $[0, -1, 0]^\top$. We use Adam [25] for stochastic gradient descent with $|\mathcal{B}| = 24$ 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, 60, 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:

$$\begin{aligned} \mathcal{M} = \{(\mathbf{p} \in \mathcal{P}, \mathbf{q} \in \mathcal{Q}) | g(\mathbf{p}) = \text{nn}(g(\mathbf{q}), g(\mathcal{P})) \wedge \\ g(\mathbf{q}) = \text{nn}(g(\mathbf{p}), g(\mathcal{Q}))\}, \end{aligned} \tag{6}$$

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

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

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 τ_1 for matching points is set to 10 cm. The inlier ratio τ_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 LMVCNN [19], a learned multi-view descriptor baseline using viewpoint clustering for offline rendering and max-view pooling for multi-view fusion. The original LMVCNN uses AlexNet [26] as its CNN backbone and outputs 128-d descriptors, but for fair comparisons, we reimplemented

τ_2	FPFH		SHOT		3DMatch		CGF		PPFNet		PPF-FoldNet		3DSmoothNet		LMVCNN		Ours	
	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2
Kitchen	50.2	8.7	74.3	26.1	58.1	9.7	61.3	12.3	89.7	-	78.7	-	97.4	62.8	98.8	76.5	**99.4**	**89.5**
Home 1	70.5	23.1	80.1	48.7	72.4	17.3	72.4	23.7	55.8	-	76.3	-	96.2	76.9	97.4	78.8	**98.7**	**85.9**
Home 2	60.1	24.0	70.7	37.5	61.5	17.8	58.2	23.1	59.1	-	61.5	-	90.9	66.3	90.9	68.3	**94.7**	**81.3**
Hotel 1	71.2	6.2	77.4	26.5	54.4	0.9	62.8	8.8	58.0	-	68.1	-	96.5	78.8	**99.6**	91.6	**99.6**	**95.1**
Hotel 2	57.7	5.8	72.1	18.3	48.1	6.7	56.7	5.8	57.7	-	71.2	-	93.3	72.1	99.0	90.4	**100.0**	**92.3**
Hotel 3	75.9	11.1	85.2	31.5	61.1	1.9	83.3	18.5	61.1	-	94.4	-	98.1	88.9	**100.0**	90.7	**100.0**	**94.4**
Study	46.9	0.3	64.0	6.2	51.7	2.4	44.9	2.4	53.4	-	62.0	-	94.5	72.3	95.2	77.4	**95.5**	**80.1**
MIT Lab	44.2	1.3	62.3	20.8	50.6	5.2	45.5	3.9	63.6	-	62.3	-	**93.5**	64.9	90.9	74.0	92.2	**76.6**
Average	59.6	10.1	73.3	26.9	57.3	7.7	60.6	12.3	62.3	-	71.8	-	95.0	72.9	96.5	81.0	**97.5**	**86.9**

Table 1: Average recall (%) of different methods on the 3DMatch benchmark with $\tau_1 = 10$cm and $\tau_2 = 0.05$ or 0.2.

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 τ_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 $\frac{1}{|\mathcal{G}|}\sum_{i=1}^{|\mathcal{G}|}\sum_{\mathbf{p},\mathbf{q}\in\mathcal{M}_i}\left[\|\mathbf{p}-\mathcal{T}_i(\mathbf{q})\|_2 < \tau_1\right]$, using the same notations as in Eq. 7. It is observed that our multi-view descriptor is about $1.5\times$ and $1.3\times$ 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-

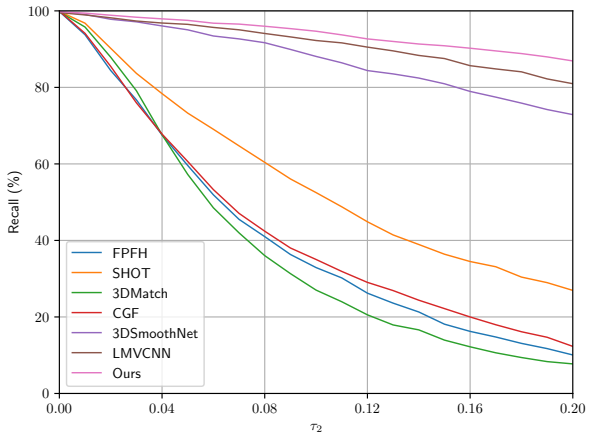

Figure 6: Average recall (%) w.r.t inlier ratio τ_2 on the 3DMatch benchmark.

	FPFH	SHOT	3DMatch	CGF	3DSmoothNet	LMVCNN	Ours
Kitchen	104	154	104	131	274	276	**380**
Home 1	158	207	134	168	325	344	**438**
Home 2	132	183	125	159	318	314	**395**
Hotel 1	103	131	74	95	272	347	**457**
Hotel 2	105	124	64	101	239	286	**407**
Hotel 3	131	160	65	134	277	301	**446**
Study	65	84	66	58	172	239	**299**
MIT Lab	84	122	84	84	247	301	**366**
Average	110	146	89	116	266	301	**398**

Table 2: Average number of correct correspondences on the 3DMatch benchmark.

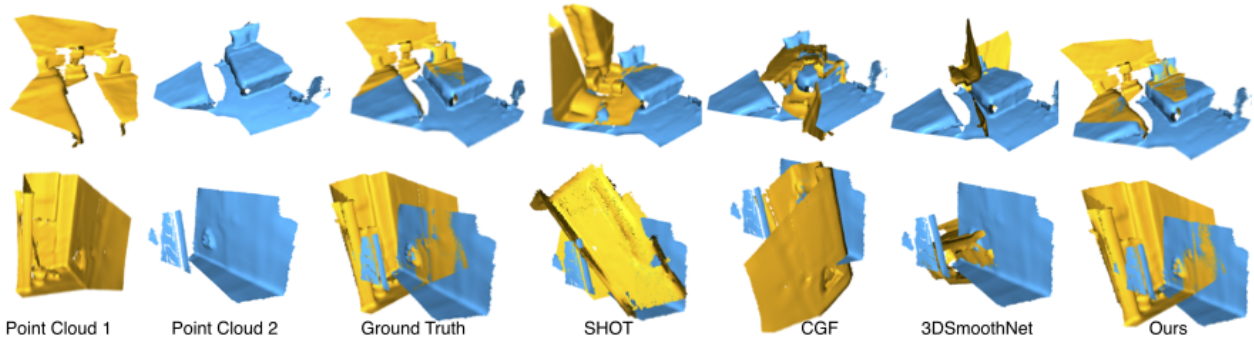

Figure 7: Geometric registration of point cloud 1 and point cloud 2 by different descriptors with RANSAC.

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.

	Rotated		Sparse (0.5)		Sparse (0.25)	
τ_2	0.05	0.2	0.05	0.2	0.05	0.2
FPFH	60.1	10.0	59.2	9.5	57.8	8.5
SHOT	73.3	26.9	72.3	25.5	70.7	23.1
3DMatch	11.6	1.4	73.1	15.8	73.3	15.9
CGF	60.7	12.5	52.6	7.8	41.7	3.8
PPFNet	0.3	-	-	-	-	-
PPF-FoldNet	73.1	-	-	-	-	-
3DSmoothNet	94.9	72.7	94.4	71.7	94.8	70.1
LMVCNN	95.7	76.7	96.2	81.3	95.9	81.5
Ours	**96.9**	**82.1**	**97.2**	**87.2**	**97.3**	**86.1**

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

	Input prep.	Inference	Total
3DMatch	0.1	2.0	2.1
CGF	10.6	0.1	10.7
3DSmoothNet	39.4	0.2	39.6
Ours	7.2	1.5	8.7

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

Running Time. 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 ρ_k in each viewpoint $\mathbf{c}_k$ 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.

	Gazebo		Wood		
	Sum.	Wint.	Sum.	Aut.	Avg.
FPFH	40.2	15.2	24.0	14.8	23.6
SHOT	73.9	45.7	64.0	60.9	61.1
3DMatch	22.8	8.7	22.4	13.9	16.9
CGF	38.6	15.2	19.2	12.2	21.3
3DSmoothNet	**91.3**	**84.1**	72.8	67.8	79.0
LMVCNN	53.3	31.8	42.4	31.3	39.7
Ours	85.3	72.0	**84.0**	**78.3**	**79.9**

Table 5: Average recall (%) on the ETH benchmark with $\tau_1 = 10$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 θ at a $\pi/4$ step (Sec. 3.1), similar to the strategy used in 3D shape recognition works [50, 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.

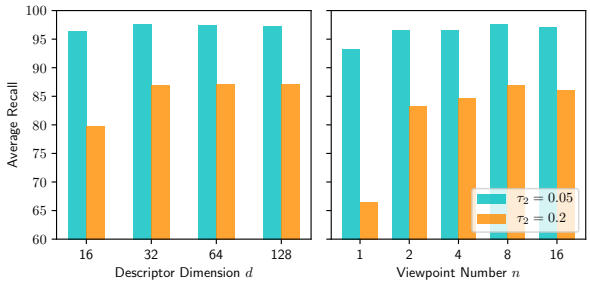

Figure 8: Average recall (%) w.r.t descriptor dimension d and viewpoint number n on the 3DMatch benchmark.

	3DMatch		ETH
τ_2	0.05	0.2	0.05
Random sampling	97.0	84.1	64.8
Viewpoint clustering	96.7	83.5	53.3
Orbited placement	92.5	55.2	42.2
Ours w/o rotation augment.	96.9	85.6	54.9
Ours	**97.5**	**86.9**	**79.9**
Max-view pooling	96.9	85.4	66.8
Fuseption	97.1	85.1	55.9
NetVLAD	95.9	77.4	58.7
Ours	**97.5**	**86.9**	**79.9**

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

[1] Edward Angel and Dave Shreiner. *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*. 6th edition, 2011. 3

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proc. IEEE CVPR*, June 2016. 2, 8, 11

[3] Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *CoRR*, abs/1908.01210, 2019. 2

[4] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proc. IEEE CVPR*, 2015. 1

[5] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proc. ECCV*, 2018. 2, 5, 6, 7

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3d point matching. In *Proc. IEEE CVPR*, 2018. 1, 2, 5

[7] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3D local features for direct pairwise registration. *CoRR*, abs/1904.04281, 2019. 2

[8] G. Elbaz, T. Avraham, and A. Fischer. 3D point cloud registration for localization using a deep neural network auto-encoder. In *Proc. IEEE CVPR*, pages 2472–2481, 2017. 1, 2, 4

[9] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. GVCNN: Group-view convolutional neural networks for 3d shape recognition. In *Proc. IEEE CVPR*, June 2018. 1, 3, 8

[10] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381395, June 1981. 5

[11] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *Proc. ECCV*, pages 224–237, 2004. 1, 2

[12] Zan Gojcic, Caifa Zhou, Jan D. Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proc. IEEE CVPR*, 2019. 1, 2, 5, 6, 7, 12

[13] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *IJCV*, 116(1):66–89, Jan 2016. 2

[14] Maciej Halber and Thomas A. Funkhouser. Structured global registration of rgb-d scans in indoor environments. *CoRR*, abs/1607.08539, 2016. 5

[15] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *Proc. IEEE CVPR*, pages 3279–3286, 2015. 1

[16] Z. Han, M. Shang, Z. Liu, C. Vong, Y. Liu, M. Zwicker, J. Han, and C. L. P. Chen. SeqViews2SeqLabels: Learning 3d global features via aggregating sequential views by rnn with attention. *IEEE TIP*, 28(2):658–672, Feb 2019. 2

[17] Xinwei He, Tengteng Huang, Song Bai, and Xiang Bai. View n-gram network for 3d object retrieval. In *Proc. IEEE ICCV*, 2019. 1

[18] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017. 5

[19] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM TOG*, 37(1):6:1–6:14, Nov. 2017. 1, 2, 3, 4, 5, 7, 8, 11

[20] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE TPAMI*, 21(5):433–449, May 1999. 1, 2

[21] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. IEEE CVPR*, 2018. 2, 4

[22] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. SGP*, pages 61–70, 2006. 3

[23] Michel Keller, Zetao Chen, Fabiola Maffra, Patrik Schmuck, and Margarita Chli. Learning deep descriptors with scale-aware triplet networks. In *Proc. IEEE CVPR*, 2018. 1

[24] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *Proc. IEEE ICCV*, 2017. 1, 2, 5, 7

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 5

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105. 2012. 2, 6

[27] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *Proc. ICRA*, pages 3050–3057, May 2014. 5

[28] Lei Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, and Chiew-Lan Tai. Sketch-R2CNN: An attentive network for vector sketch recognition. *CoRR*, abs/1811.08170, 2018. 2

[29] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. *ACM TOG*, 37(6):221:1–221:11, Dec. 2018. 2

[30] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A differentiable renderer for image-based 3d reasoning. *CoRR*, abs/1904.01786, 2019. 1, 2, 3, 4

[31] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Proc. ECCV*, 2014. 2

[32] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. GeoDesc: Learning local descriptors by integrating geometry constraints. In *Proc. ECCV*, September 2018. 1

[33] D. Maturana and S. Scherer. VoxNet: A 3d convolutional neural network for real-time object recognition. In *Proc. IROS*, pages 922–928, Sep. 2015. 2

[34] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NIPS*, pages 4826–4837. Curran Associates, Inc., 2017. 1, 2, 4

[35] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Repeatability is not enough: Learning affine regions via discriminability. In *Proc. ECCV*, 2018. 1

[36] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009. 7

[37] Naila Murray and Florent Perronnin. Generalized max pooling. In *Proc. IEEE CVPR*, 2014. 2

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NIPS*, pages 8026–8037. 2019. 5

[39] Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. Pix2Vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *CoRR*, abs/1903.11149, 2019. 2

[40] François Pomerleau, M. Liu, Francis Colas, and Roland Siegwart. Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14):1705–1711, Dec. 2012. 7

[41] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE CVPR*, 2017. 1, 2

[42] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. IEEE CVPR*, 2016. 1, 2, 3, 11

[43] Riccardo Roveri, Lukas Rahmann, Cengiz Oztireli, and Markus Gross. A network architecture for point cloud classification via automatic depth images generation. In *Proc. IEEE CVPR*, 2018. 2

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, Dec 2015. 2

[45] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. ICRA*, pages 3212–3217, May 2009. 1, 2, 5

[46] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Proc. IROS*, pages 3384–3391, 2008. 1, 2

[47] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proc. ICRA*, pages 1–4, May 2011. 5

[48] Samuele Salti, Federico Tombari, and Luigi di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *CVIU*, 125:251–264, 2014. 1

[49] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proc. IEEE CVPR*, pages 2930–2937, June 2013. 5

[50] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. IEEE ICCV*, 2015. 1, 2, 3, 8, 11

[51] Y. Tian, B. Fan, and F. Wu. L2-Net: Deep learning of discriminative patch descriptor in euclidean space. In *Proc. IEEE CVPR*, pages 6128–6136, 2017. 1, 2, 4, 11

[52] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proc. 3DOR*, pages 57–62, 2010. 1, 2

[53] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proc. ECCV*, pages 356–369, 2010. 1, 2, 5

[54] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. 4

[55] Julien P. C. Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip H. S. Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. *CoRR*, abs/1603.05772, 2016. 5

[56] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *Proc. BMVC*, 2017. 1, 2, 3, 8

[57] Hanyu Wang, Jianwei Guo, Dong-Ming Yan, Weize Quan, and Xiaopeng Zhang. Learning 3d keypoint descriptors for non-rigid shape matching. In *Proc. ECCV*, 2018. 1

[58] Yifan Wang, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *CoRR*, abs/1906.04173, 2019. 2

[59] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. IEEE CVPR*, pages 1912–1920, 2015. 2

[60] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *Proc. IEEE ICCV*, 2013. 1, 5

[61] Zhige Xie, Kai Xu, Wen Shan, Ligang Liu, Yueshan Xiong, and Hui Huang. Projective feature learning for 3d shapes with multi-view depth images. *CGF*, 34(7):1–11, 2015. 4

[62] X. Xing, Y. Cai, T. Lu, S. Cai, Y. Yang, and D. Wen. 3DTNet: Learning local features using 2d and 3d cues. In *Proc. 3DV*, pages 435–443, Sep. 2018. 2

[63] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly supervised local 3d features for point cloud registration. In *Proc. ECCV*, 2018. 2

[64] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *Proc. ECCV*, pages 467–483, 2016. 1

[65] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *Proc. ICLR*, 2013. 2

[66] Andy Zeng, Shuran Song, Matthias Niessner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proc. IEEE CVPR*, 2017. 1, 2, 5, 7

[67] Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. Learning and matching multi-view descriptors for registration of point clouds. In *Proc. ECCV*, 2018. 1, 2, 8, 11

6. Supplementary Material

6.1. CNN

In Sec. 3.2 of the main text, we adopt a CNN architecture similar to L2-Net [51] to extract feature maps for each view patch. The detailed configuration of the network is listed in Table 7. Note that the network input is of size 64×64 with a single depth channel, and the final output is of size 8×8 with 128 feature channels.

#	Layer	Kernel	Stride	Padding
1	Conv - Norm - ReLU	3×3×32	2	1
2	Conv - Norm - ReLU	3×3×32	1	1
3	Conv - Norm - ReLU	3×3×64	2	1
4	Conv - Norm - ReLU	3×3×64	1	1
5	Conv - Norm - ReLU	3×3×128	2	1
6	Conv - Norm - ReLU	3×3×128	1	1

Table 7: CNN backbone for feature extraction of each view patch. In the *Kernel* column, the first two numbers represent the kernel size, and the third number is the number of output feature channels.

6.2. Multi-view Rendering

In Fig. 9, we visualize the optimizable viewpoints after training. We also show the viewpoints obtained by a clustering scheme similar to the one in [19]. Specifically, 150 spherical coordinates (θ, ϕ) are randomly sampled on the hemisphere where point normals reside, and then the k-medoids clustering algorithm is applied to select three viewing directions. For each viewing direction, a virtual camera is placed at distances of 0.3m, 0.6m, 0.9m to the points of interest, and each rendered view patch is augmented with four in-plane rotations.

As shown in Fig. 9, there are mainly two differences between the hand-crafted rule and our method. First, the hand-crafted rule places some viewpoints far from points of interest, while the learnt viewpoints have more concentrated distance range, indicating the relatively low importance of broader global context. Second, the hand-crafted rule selects some dominant viewing directions through clustering, whereas the learnt viewpoints have more distributed viewing directions around the points of interests, which can help to capture more local geometry variance. In sum, the learnt viewpoints effectively balance the extent of context-awareness and local details in extracted descriptors, challenging the design wisdom of hand-crafted rules.

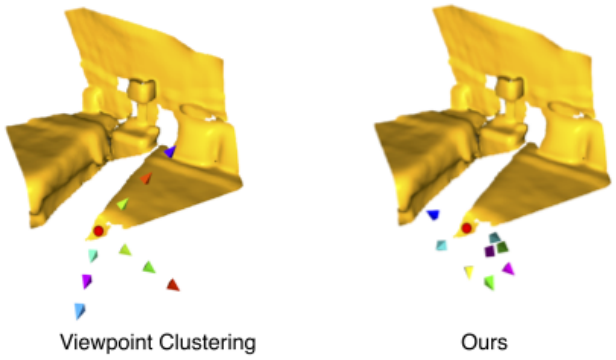

Figure 9: Visualization of viewpoints obtained by a clustering scheme and our method. The red spheres denote the points of interest, and the pyramids represent virtual cameras.

6.3. Multi-view Fusion

In Sec. 4.4 of the main text, we compared the proposed soft-view pooling with alternative fusion approaches including max-view pooling [19, 50, 42], Fuseption [67], and NetVLAD [2]. Fuseption has two branches: in the first branch, the feature maps of all the views are first channelwise concatenated together in a specific order and then fed into a convolutional block; in the second branch, max-pooling is applied to the inputs and the results are added to the output of the first branch, serving as a shortcut connection. NetVLAD is a descriptor pooling method that summarizes the residuals of each input w.r.t. several learnable cluster centers. The number of cluster centers is a hyper parameter, which is set to eight in our experiments. The network f is trained with the alternative fusion approaches, while the other stages are kept unchanged. The descriptor dimension d is set to 32, and the optimizable viewpoint number n is set to 8.

In Fig. 10, we visualize the rendered multi-view inputs to CNNs, extracted feature maps for each view, and fused feature maps across views. It is observed that the CNN is influenced by multi-view fusion for feature extraction. Before fusion, for soft-view pooling and NetVLAD, the feature maps of each view extracted by the CNN tend to have more response, compared to max-view pooling and Fuseption. After fusion, the feature maps produced by max-view pooling and NetVLAD tend to have more high response than soft-view pooling and Fuseption. Note that for each location in the fused feature maps, max-view pooling only selects the strongest input response across views and discards the rest, while our soft-view pooling collectively considers all the inputs in an attentive manner for integration.

6.4. Comparisons with 3DSmoothNet

In Fig. 11, we visualize the color-coded local descriptors for all the points in the point clouds. Specifically, we project the high dimensional descriptors with PCA and keep the first three components, which are color-coded. It is observed that the descriptors of 3DSmoothNet and our method are both geometry-aware. Particularly, our method is able to capture more geometric changes in the point clouds (see the highlighted wall, pillow and floor regions of the point clouds in Fig. 11). In Fig. 12, we show additional geometric registration results of point cloud pairs, which further demonstrate the above advantage of our method.

For the running time of 3DSmoothNet in Sec. 4.2 of the main text, we observed some gap between our experiment results (input prep: 39.4ms; inference: 0.2ms) and the performance reported by the authors (input prep: 4.2ms; inference: 0.3ms). We used the source code[1] of 3DSmoothNet released by the authors, and the running time gap of input preparation is likely due to the difference of hardware configurations. In [12], they used a PC with an Intel Xeon E5-1650, a 32GB RAM and an NVIDIA GeForce GTX 1080 GPU, while we used a PC with an Intel Core i7 @ 3.6GHz, a 32GB RAM and an NVIDIA GTX 1080Ti GPU. Their input preparation stage involving LRF computation and SDV voxelization runs on CPU, which may be accelerated with GPU for further improvement.

[1] https://github.com/zgojcic/3DSmoothNet

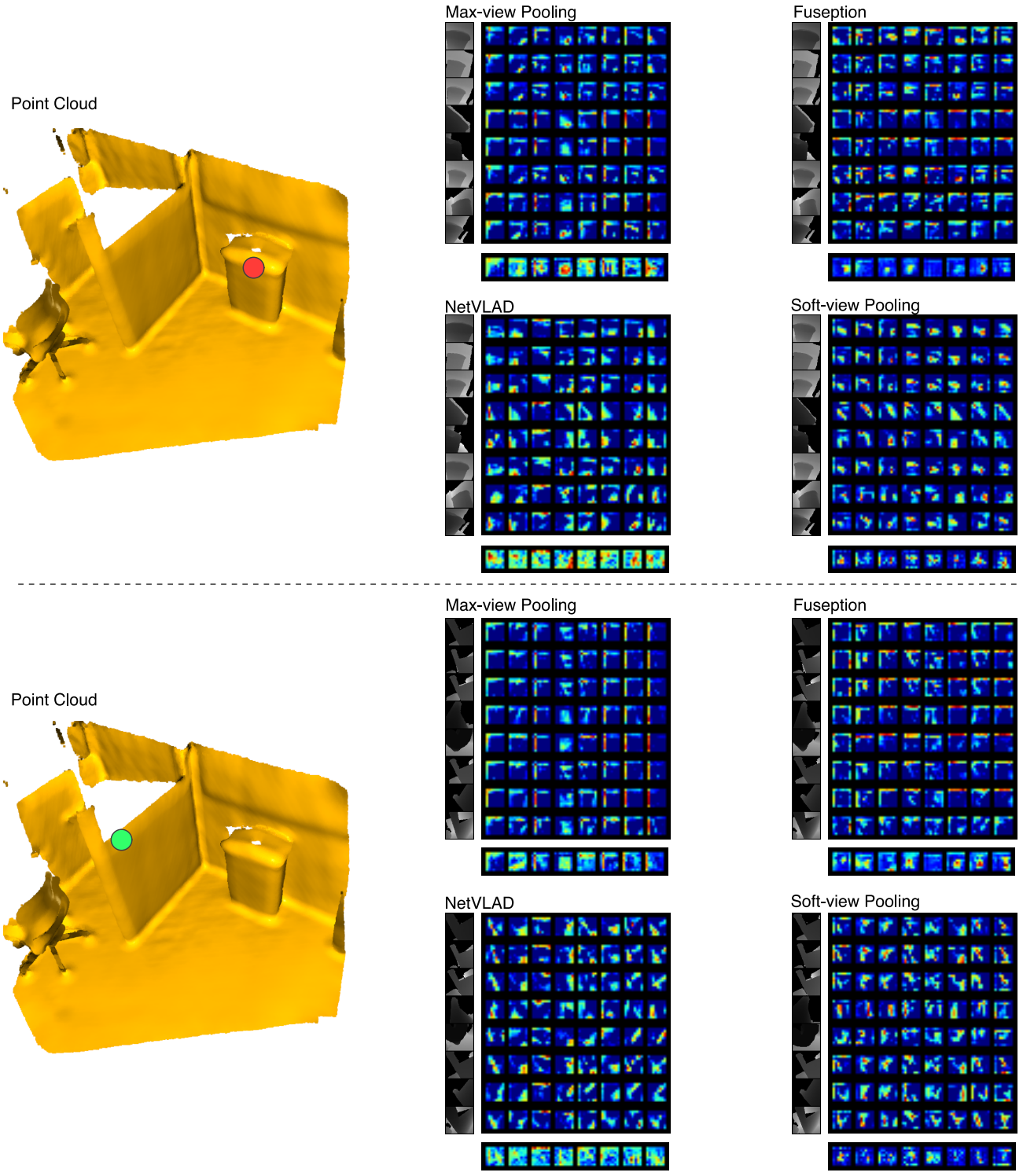

Figure 10: Visualizations for multi-view fusion by different methods. The top part is for the red keypoint while the bottom part is for the green keypoint. In each block, we visualize the view patches (depth) rendered with eight optimizable viewpoints on the left. On the right are the corresponding convolutional feature maps (with channel indices $\{1, 2, 4, 8, 16, 32, 64, 128\}$) before fusion, and each row is for a specific view. Fused feature maps across views are placed on the bottom.

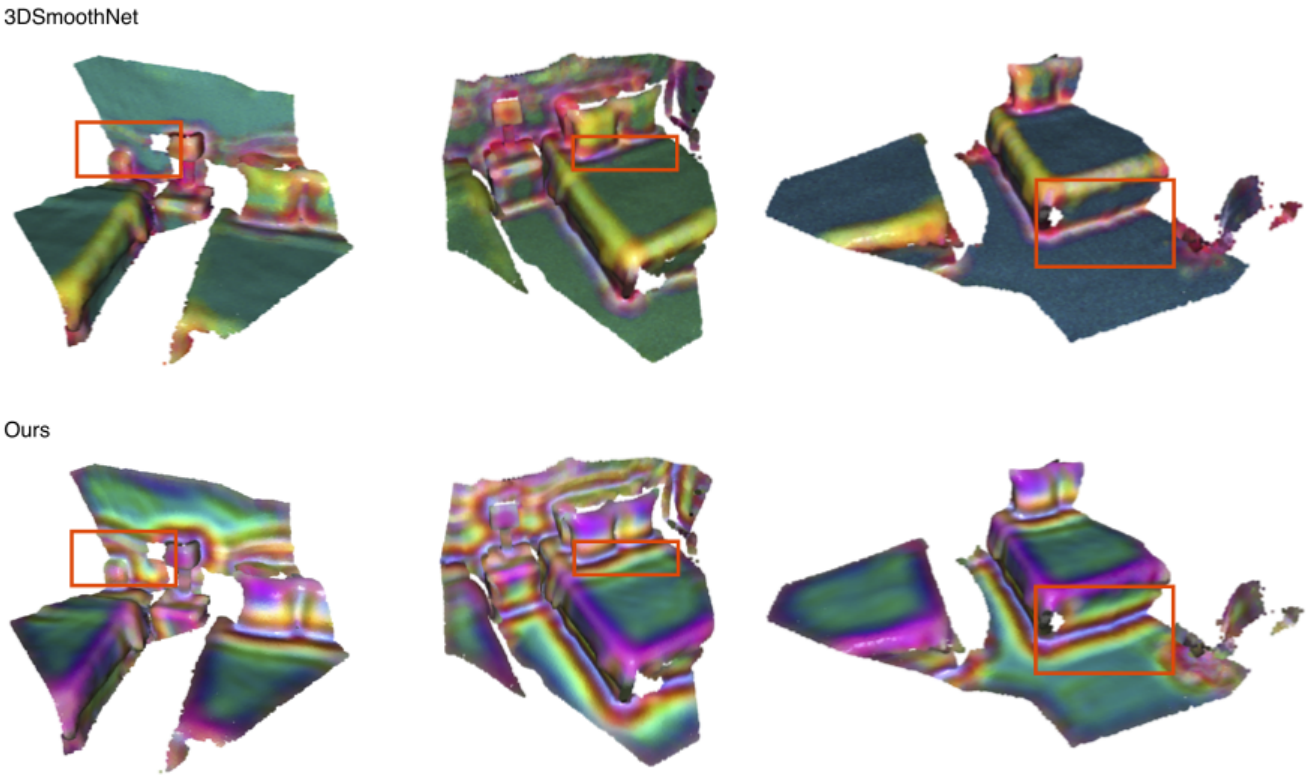

Figure 11: Visualization of local descriptors for 3DSmoothNet and our method. The high dimensional descriptors are projected with PCA to 3D space and color-coded. The highlighted regions show that our method can better capture geometric changes in the point clouds.

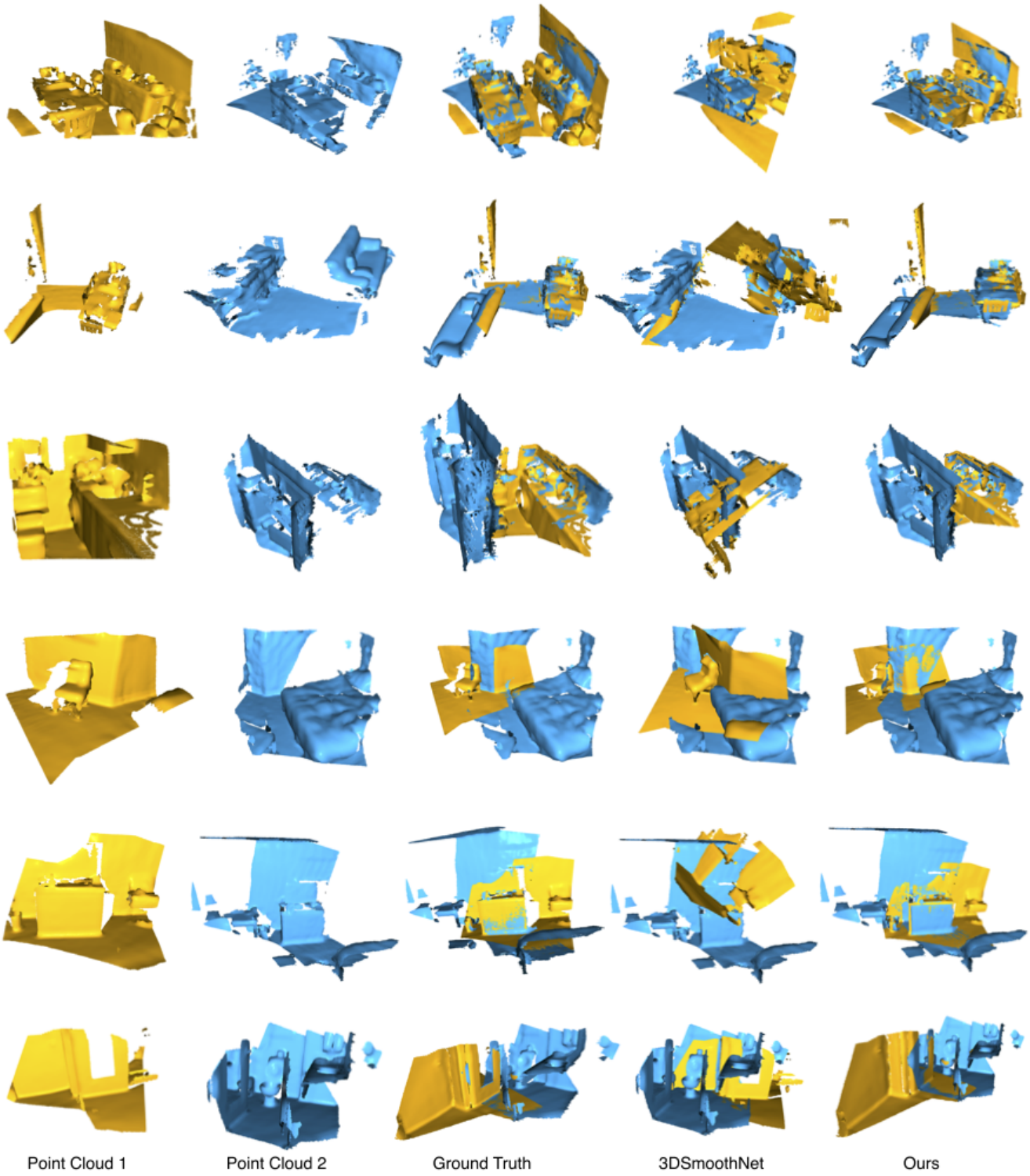

Figure 12: More geometric registration results with RANSAC for 3DSmoothNet and our method.