Sharp Feature Consolidation from Raw 3D Point Clouds via Displacement Learning

Tong Zhao, Mulin Yu, Pierre Alliez, Florent Lafarge

Sharp Feature Consolidation from Raw 3D Point Clouds via Displacement Learning

Mulin Yu* Tong Zhao* Pierre Alliez Florent Lafarge

Université Côte d'Azur, Inria, France

firstname.lastname@inria.fr

Abstract

Detecting sharp features in raw point clouds is an essential step in designing efficient priors in several 3D Vision applications. This paper presents a deep learning-based approach that learns to detect and consolidate sharp feature points on raw 3D point clouds. We devise a multi-task neural network architecture that identifies points near sharp features and predicts displacement vectors toward the local sharp features. The so-detected points are thus consolidated via relocation. Our approach is robust against noise by utilizing a dynamic labeling oracle during the training phase. The approach is also flexible and can be combined with several popular point-based network architectures. Our experiments demonstrate that our approach outperforms the previous work in terms of detection accuracy measured on the popular ABC dataset. We show the efficacy of the proposed approach by applying it to several 3D Vision tasks.

1. Introduction

For scanned or Computer-Aided Design (CAD) 3D models, a sharp feature usually refers to creases and corners where the surface is not smooth. Recognizing such sharp features from raw point cloud is a preliminary step for several point cloud processing tasks such as surface reconstruction [21, 9, 2, 11, 29], extraction of feature graphs [15, 20] or semantic segmentation [10, 18]. With the development of point cloud acquisition techniques and the release of well-annotated 3D datasets [14], many data-driven approaches have been proposed in recent years. Most previous approaches treat sharp feature detection as a classification problem, which leads to several issues: (1) Feature points are not located exactly on sharp features in general, which leads to incomplete features; (2) To deal with this issue, several approaches introduce a distance threshold that defines the maximum distance between a feature point to the nearest sharp feature. Using such a distance threshold in common classification problems highly impacts the final prediction and it is delicate to choose (see Figure 1); (3) Noise has a large impact over the classification results; and (4) The network can be easily overfitted given patches or clouds with ground truth labels.

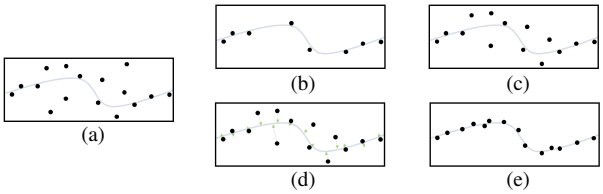

Figure 1. Illustrating the impact of distance thresholds and displacement vectors on sharp feature detection. (a) Noisy point cloud around a sharp feature. (b) Given a small distance threshold, existing classification-based methods detect few accurate sharp feature points, which do not capture well the entire sharp feature. (c) Given a large distance threshold, the detected sharp feature points become vague. (d) Our method can detect the sharp feature points within a large distance threshold, and the learned displacement vectors relocate the points closer to the sharp feature. (e) Resulting accurate detection.

Inspired by displacement-based point cloud denoising methods, we propose a deep learning framework for detecting and consolidating sharp features from a raw 3D point cloud, namely SFCNet. Our framework contains two main components: (1) A point-to-feature oracle that largely improves the data augmentation quality so as to prevent models from overfitting; (2) A multi-task neural network architecture that predicts altogether a binary label (sharp/smooth) for each input point and a displacement vector for each sharp feature point. Our approach outperforms popular unsupervised methods and supervised methods on the ABC dataset [14]. We also evaluate our method on noisy point clouds and real scanned models. Quantitative and qualitative results are provided in Section 4.

In summary, our insights and technical contributions are:

*Equal contribution listed in alphabetical order.

- A multi-task neural network architecture that performs accurate sharp feature detection and consolidation;
- Resilience to noise;
- A general learning framework that combines with popular point-based neural network architectures;
- A comprehensive analysis of the influence of displacement learning in sharp feature detection task.

2. Related work

In this section, we first review the popular neural network structures for extracting geometric features from raw point clouds. We then review related work on sharp feature detection and consolidation from raw point clouds, including unsupervised methods and supervised methods. We give last a short review of point cloud denoising, focusing on displacement-based methods that motivate our displacement learning approach.

2.1. Learning-based geometric feature extraction

While convolutional neural networks yield great results for many image-related tasks, point cloud processing remains a challenge due to the unstructured and irregular nature of point clouds. The PointNet approach proposed in 2017 by Qi et al. [24] quickly became a popular learning approach for point cloud processing, and was applied to many tasks. It extracts point features from point coordinates by applying input and feature transformations and then aggregates global point features by max pooling. Qi et al. then proposed PointNet++ [25], which improves over previous work by considering local geometries with sampling and grouping layers. Guerrero et al. proposed another variant of PointNet, referred to as PCPNet [12]. Instead of taking the whole point cloud as input, PCPNet deals with patches to learn specific local point features. More recently, Qian et al. proposed ASSANet [26], a novel separable set abstraction module that further improves PointNet++.

Another family of methods is spatial-based. Wang et al. introduced a graph neural network, referred to as DGCNN, which learns features on KNN neighbors using successive EdgeConv layers [32]. The So-Net [17] is devised to model the spatial distribution of the input points. The SPLATNet [30] computes hierarchical and spatially-aware features of input points with sparse and efficient lattice filters. All the above networks can be utilized as a backbone of our framework, as long as they produce pointwise descriptors.

2.2. Sharp feature detection and consolidation

Unsupervised methods. Several unsupervised methods rely upon Moving Least Squares (MLS) surface reconstruction [16]. Fleishman et al. [8] proposed a Moving Least Squares method for reconstructing a piecewise smooth surface from a point cloud. Sharp features are identified as points at the intersection of multiple iteratively fitted surfaces. Daniels et al. [6] improved over this method by using an adaptive threshold. They define a projection operator which takes into consideration edge points, followed by a smoothing filter. However, the performance of these methods highly depends on the quality of surface reconstruction results, which may fail in noisy cases. Moreover, most of these methods are computationally intensive and usually inefficient.

Huang et al. [13] contributed an edge-aware point cloud resampling technique built upon a normal estimation step followed by a robust projection operator. The final outcome highly depends on the quality of the normal estimation step.

Another line of methods detects sharp features via local geometric descriptors. Weber et al. [33] rely upon point-sampled geometries. They first eliminate planar points by performing a flatness test. Clustering over the Gauss map is then performed to find sharp feature points. Mérigot et al. [19] utilize convolved covariance matrices of Voronoi cells to characterize point properties and then filter out smooth points by a fixed threshold. Bazazian et al. [3] accelerated this approach by computing tensors directly from the nearest neighbors of points. The above methods are computationally efficient but defining relevant thresholds is a trial-and-error process.

Supervised methods. With the development of neural networks, more and more supervised methods are proposed for point cloud processing. Yu et el. [34] proposed EC-net in 2018, in which they extend PU-Net [35], a network designed to upsample and perform an edge-aware point cloud consolidation. Despite its success, its results are impacted by the way the patches are created. Wang et al. [31] contributed an end-to-end neural network, called PIE-Net, that is trained for parameter inference of feature edges over a 3D point cloud, where the output consists of one or more parametric curves. However, it is limited in terms of scalability. Loizou et al. [18] devised a convolutional neural network based on DGCNN [32], to detect the boundaries of parts in 3D point clouds. Recently, Himeur et al. [5] introduced PCEDNet, a lightweight neural network that takes as input a series of multi-scale differential geometric descriptors of points and predicts three-class labels for a sharp feature and its surroundings. In contrast, our framework learns displacement while detecting sharp features, which offers higher genericity and robustness to noise.

2.3. Displacement-based point cloud denoising

As the displacement property indicates the mismatches between the true model and the sampling, it can be utilized to reduce or even eliminate the intrinsic noise in point clouds. Rakotosaona et al. proposed PointcleanNet [28], which pioneered the idea of displacement learning for point

cloud denoising. They designed a two-step neural network based on PCPNet [12] that first eliminates outliers before estimating displacement vectors for all inliers. Instead, our method uses a shared backbone for both classification and regression, which significantly reduces the number of parameters. Pistilli et al. [23] introduced a fully convolutional network predicting displacement vectors. The success of the above methods shows the potential of displacement learning for sharp feature detection.

3. Our method

During training session, our method takes as input a ground truth mesh $\mathcal{M}$, a set of ground truth sharp features $\mathcal{F}$ (with parametric representations) and an initial point cloud $\mathbf{P} \in \mathbb{R}^{c\times 3}$ sampled near $\mathcal{M}$, where $\mathbf{n}$ is the number of points and 3 represents the $\{x, y, z\}$ coordinates of the points. During inference, our method only needs a 3D point cloud $\mathbf{P}$. It outputs a set of binary classification labels *Cla* $\in \{0, 1\}^{\mathbf{n}}$ and a set of displacement vectors *Dis* $\in \mathbb{R}^{\mathbf{n}\times 3}$, simultaneously.

Figure 2 provides an overview of our architecture with four modules: *point-to-feature oracle*, *point descriptor extractor*, *sharp feature detector* and *displacement predictor*.

3.1. Point-to-feature oracle

Most data-driven methods use pre-defined labels for point clouds during the training session. This operation would probably lead to overfitting, or designing complicated loss functions to compute gradients, which makes the network hard to adapt to other datasets or architectures. Instead, we propose a point-to-feature oracle. It updates dynamically the ground truth classification labels and the ground truth displacement vectors according to the current noisy point cloud and the ground truth mesh before forwarding inputs to the network. Our network can deal with, but is not limited to, four types of sharp creases: segments, circles, ellipsoids and B-splines.

Each time we first add a uniform random noise with a user-defined noise level (1% length of the bounding box diagonal by default) on the initial point cloud $\mathbf{P}$, which is sampled from a CAD mesh. We then use the oracle to find, for each point p, its nearest ground truth sharp feature, and then project p onto this feature. p is labeled as a sharp feature point if the distance between p and its projected point is smaller than a user-defined distance threshold, which is set to 0.03 times the length of the bounding box diagonal in our experiments.

For segments and (open or closed) circles, we compute the point's projection onto sharp features by algebra computation. To simplify the computation for (open or closed) ellipsoids and B-splines, we generate a dense point sample on the sharp features and find the nearest one to the query point p as its projection.

3.2. Point descriptor extractor

The so-called backbone is used to extract pointwise descriptions $\mathbf{f}$ for point clouds. We experimented with DGCNN [32] and PCPNet [12] backbones, but users can plug their own backbone as long as it produces pointwise descriptors.

DGCNN is a graph-based convolutional neural network that takes as input a point cloud of fixed cardinality $\mathbf{N}$. It yields a global descriptor of size 1,024 for the whole point cloud and a multi-scale local descriptor of size 448 for each point. By concatenating global and local descriptors, each point has a descriptor of size 1,472. Since $\mathbf{N}$ is fixed for DGCNN, the point cloud must have the same cardinality during inference, which is its main limitation.

PCPNet is a PointNet-based neural network that takes as input a center (query) point along with its k-nearest neighbors and produces a local descriptor of size 1,024. While the cardinality of point clouds is not restricted, the inference is substantially slower than that of the DGCNN backbone.

3.3. Network architecture

After generating the features, two multi-layer perceptrons take $\mathbf{f}$ as input and output *Dis* and *Cla*, respectively. Both $\mathbf{M_1}$ and $\mathbf{M_2}$ comprise three layers. *Cla* is a set of binary classification labels, indicating whether a point is near a sharp feature. *Dis* is a set of pointwise vectors that represent the relocation from sharp feature points to their closest sharp features. The displacement vectors for smooth points are ignored by applying masks to the loss function.

3.4. Loss function and training

Our loss function is composed of two parts, a Binary Cross Entropy (BCE) with logits loss for sharp feature classification and a masked Mean Squared Error (MSE) loss for displacement prediction. The two losses cooperate in order to yield both a more accurate classification and a more precise displacement regression.

BCE with logits loss. Given a point cloud with $\mathbf{n}$ points, the ground truth labels $y = \{y_1, \ldots, y_{\mathbf{n}}\}$, the predictions $x = \{x_1, \ldots, x_{\mathbf{n}}\}$ from $\mathbf{M_1}$ where $x_i \in [-\infty, +\infty]$, and the sigmoid function σ, the loss is computed as:

$$l_c(x, y) = -\frac{1}{\mathbf{n}} \sum_{i=1}^{\mathbf{n}} [y_i \cdot \log \sigma(x_i) + (1 - y_i) \cdot \log(1 - \sigma(x_i))] \tag{1}$$

By combining the Sigmoid layer with BCE with logits loss, the back-propagation is numerically more stable by taking advantage of the log-sum-exp formulation, since the computation converts to:

$$l_c(x_i, y_i) = \begin{cases} (1 - y_i) \cdot x_i + \log(1 + e^{-x_i}) & \text{if } x_i > 0 \\ -x_i \cdot y_i + \log(e^{x_i} + 1) & \text{otherwise} \end{cases} \tag{2}$$

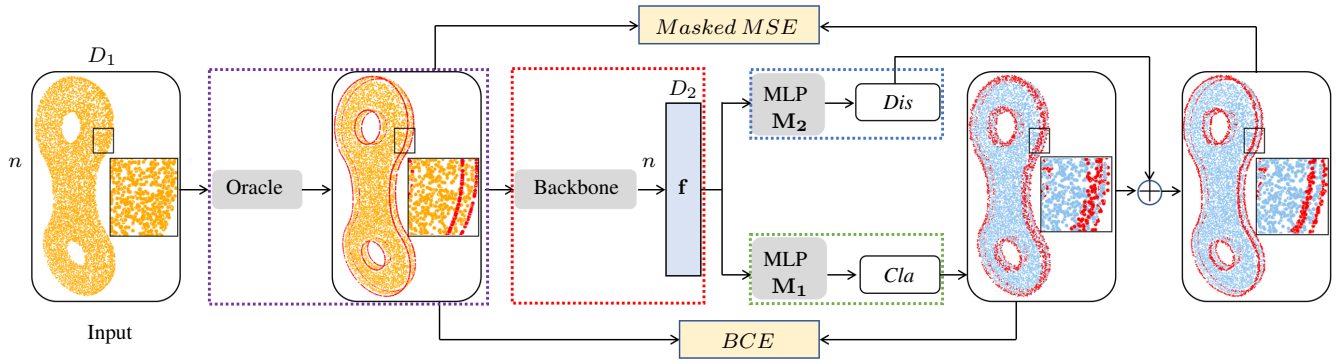

Figure 2. Overview of the proposed learning architecture. It comprises four modules: *point-to-feature oracle* (purple dash), *point descriptor extractor* (red dash), *sharp feature detector* (green dash) and *displacement predictor* (blue dash). *Point-to-feature oracle* module is only used for training. It introduces noise to the input, computes the ground truth classification labels and the ground truth displacement vectors. A backbone is used to extract high dimensional representations $\mathbf{f}$ (size: $n \times D_2$) in the *point descriptor extractor* module. **MLP** stands for multi-layer perceptron. *Cla* and *Dis* represent binary sharp feature labels and displacement vectors, respectively. $\bigoplus$ adds the corresponding predicted displacement vectors to detected sharp feature points to make them lie closer to the ground truth sharp features. BCE and $Masked\ MSE$ are two losses calculated base on *Cla*, *Dis* and the ground truth generated by the oracle.

Masked MSE loss. Given ground truth displacement vectors $u = \{u_1, \ldots, u_\mathbf{n}\}$ and the predicted vectors $v = \{v_1, \ldots, v_\mathbf{n}\}$ from $\mathbf{M_2}$, the masked MSE is:

$$l_d(y, u, v) = -\frac{1}{\mathbf{n}} \sum_{i=1}^{\mathbf{n}} \mathbb{1}_1(y_i) \cdot \|v_i - u_i\|^2 \qquad (3)$$

The mask makes the network concentrate on regressing displacement vectors for points around sharp features.

Training. The total loss energy is defined as:

$$l(x, y, u, v) = w_c \cdot l_c(x, y) + w_d \cdot l_d(y, u, v) \qquad (4)$$

where w_c and w_d denote user-defined coefficients to balance the two tasks. An appropriate balance between these two losses is recommended to obtain a good cooperation of two tasks. During training, we first set a large w_c and a small w_d ($w_c = 1$ and $w_d = 0.01$ for both backbones) during the first 10 epochs, because a good classification result is required for learning displacement vectors. We then set a smaller w_c and a larger w_d ($w_c = 0.01$ and $w_d = 100$ for DGCNN-backbone; $w_c = 1$ and $w_d = 20$ for PCPNet-backbone) until the network converges to ensure accurate displacement prediction.

4. Experiments

We implemented our approach using the *Geomdl* library [4] for B-spline distance computation and the PyTorch deep learning framework [22]. We first introduce the setup of our experiments. Then we study the robustness of our approach to the hyper-parameters, followed by an ablation study of each component of our approach to demonstrate their functionalities. We show both the quantitative and the qualitative results, as well as comparisons with unsupervised and supervised methods.

Dataset. The ABC dataset [14] is a collection of CAD models with parameterized curves and annotated surfaces. We selected 5,093 models which contain at least one sharp crease. We split them into three non-overlapping sets used for training (3,623 models), evaluating (471 models) and testing (999 models).

Metrics. We selected two lines of quantitative metrics to evaluate our approach: (1) Common metrics used for binary classification and (2) Distance metrics used for regression. Similarly to previous approaches [5, 18, 31], we adopt the following metrics for classification: Precision, Recall, F1-score, Accuracy and Intersection over Union score (IoU). Besides, we estimate two metrics based on the Euclidean distance. More specifically, we compute the distance between detected sharp feature points and continuous ground truth sharp features, by taking advantage of the annotation of the ABC dataset and our point-to-feature oracle, which evaluates the correctness of the sharp feature points. We refer to this distance with *PtF* distance. We then estimate the Chamfer distance between the detected sharp feature points and a set of uniformly-sampled points on the ground truth sharp features, which evaluates the completeness of the detected features.

Setups. We experimented with two backbones to validate our approach: PCPNet [12], an architecture based on PointNet; and DGCNN [32], an architecture based on graph neural networks. PCPNet takes a point cloud with arbitrary cardinality, while DGCNN takes a point cloud with a fixed cardinality (set by default to 10k in our experiments). We refer to our model with PCPNet backbone as SFCNet-P and the one with DGCNN backbone as SFCNet-D. The point cloud

is normalized inside the bounding box from $[-1, -1, -1]$ to $[1, 1, 1]$ and centered at the origin. By default, we add a random uniform noise with a noise level set to 0.01 during training. To compare different methods, we add a similar noise level in the test set. The default backbone is DGCNN and the default sharp feature threshold is 0.03. Figure 3 shows several results produced by our default model on the test set.

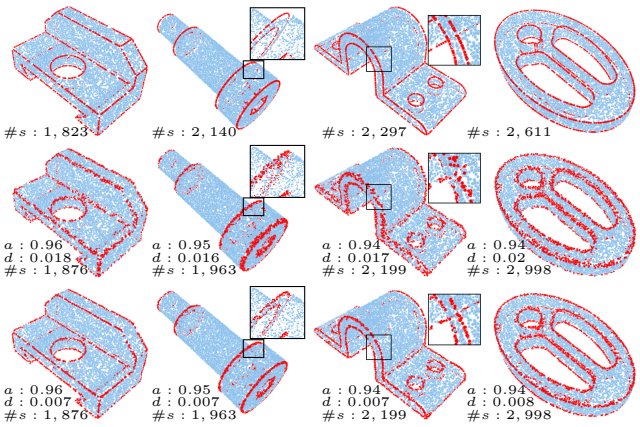

Figure 3. 3D models from the ABC dataset. Our method displaces the detected sharp feature points closer to the ground truth. Top row: ground truth. Middle row: detected raw sharp feature points. Bottom row: displaced sharp feature points. Sharp feature points are depicted in red, and smooth points are depicted in blue. a, d and $\#s$ denote accuracy, point-to-feature distance and number of sharp feature points, respectively.

4.1. Robustness study

We evaluate the sensitivity of our approach to the sharp distance threshold, defined as the maximum point-to-feature distance for sharp feature points, and to the noise level of the point clouds.

Distance threshold $d_{\max}$. A large sharp distance threshold helps the framework detect more sharp feature points, but displacement learning becomes more challenging, as shown in Figure 4. We trained four SFCNet-D networks using the same parameters, except for the distance threshold. Table 1 records the measured metrics for $d_{\max} \in \{0.02, 0.03, 0.05, 0.1\}$. We observe that the model with $d_{\max} = 0.1$ achieves good scores for most of the classification criteria while the distance criteria are much worse than $d_{\max} = 0.03$. Figure 4 shows the visualization results on one model. The right one detects more than 5k sharp feature points while we can see from the figure that some of the detected sharp feature points can not be well displaced to the sharp features.

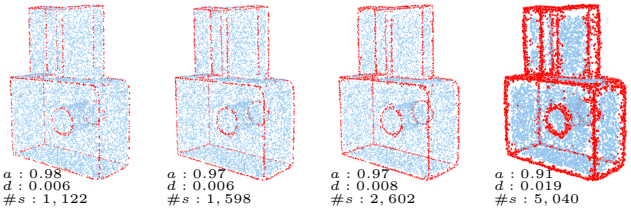

Figure 4. Visual comparison for different distance thresholds. From left to right: distance thresholds set to 0.02, 0.03, 0.05 and 0.1. The one with $d_{\max} = 0.1$ misclassified some smooth points as sharp feature points, while the ones with $d_{\max} \in \{0.03, 0.05\}$ keep a good trade-off between the classification accuracy and the distance criterion.

Noise level n. This parameter has a strong relationship with $d_{\max}$ thus we cannot fix $d_{\max}$ for all experiments. We choose to train a series of SFCNet-D networks with different pairs of $(\mathrm{n}, d_{\max})$ and then compare the distance metrics. Table 1 records the quantitative results, which indicates that our method is robust to noise when the distance threshold is larger than the noise level. When the distance threshold is smaller than the noise level, the sharp feature points can not be accurately detected, as shown in the eighth row of Table 1 and in the fourth model of Figure 5 where few sharp feature points are detected.

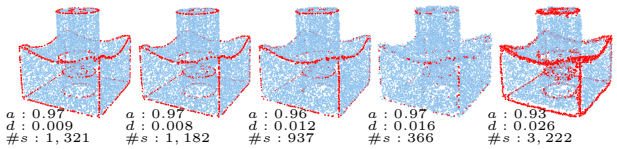

Figure 5. Visual comparison with different pairs of sharp distance thresholds and noise levels. From left to right, the pairs of sharp distance thresholds and noise levels have the same order as in Table 1. An incompatible choice of $d_{\max}$ and n (the 4th model) leads to a bad result.

4.2. Ablation study

We evaluate the role of the different components by removing one or several components from our approach and comparing the quantitative results on the test set.

Point-to-feature oracle. To show the importance of the dynamic labeling oracle, we consider four pairs of contrast experiments: D1 versus D2, D3 versus D4, P1 versus P2 and P3 versus P4 as shown in Table 1. The results show that the oracle can efficiently improve both the classification and the regression performance. In addition, the oracle helps the network converge faster and better. In our experiments, networks without oracle have much more chances to diverge (see P3) than the ones with oracle.

		Classification(%)					Distance($\times 10^{-2}$)	
		Recall ↑	Precison ↑	IoU ↑	F1 ↑	Accuracy ↑	Chamfer ↓	*PtF* ↓
Threshold	$(0.01, 0.02)$	76.20	66.31	54.74	68.29	**93.05**	5.43	2.77
	$(0.01, 0.03)$	79.65	82.68	67.65	78.98	92.65	**4.55**	**2.65**
	$(0.01, 0.05)$	80.58	84.53	69.88	80.77	90.87	5.16	3.26
	$(0.01, 0.1)$	**86.15**	**89.51**	**78.14**	**86.69**	89.96	5.84	4.17
Noise	$(0, 0.03)$	**87.27**	70.91	63.96	76.37	91.83	4.81	3.02
	$(0.01, 0.03)$	79.65	82.68	67.65	78.98	92.65	**4.55**	**2.65**
	$(0.03, 0.03)$	71.34	65.50	51.26	65.63	90.48	6.87	3.73
	$(0.06, 0.03)$	45.32	67.22	36.52	50.71	**93.63**	11.42	3.81
	$(0.06, 0.1)$	84.3	**83.67**	**71.63**	**81.89**	85.88	7.82	5.31
Components	D1: DGCNN+CLA	76.43	79.03	60.26	72.45	87.58	7.99	4.40
	D2: DGCNN+ORACLE+CLA	80.62	80.54	67.40	78.38	92.34	6.41	3.61
	D3: DGCNN+CLA+DIS	**87.32**	74.91	66.44	77.68	89.11	5.72	4.01
	D4: SFCNet-D	79.65	**82.68**	**67.65**	**78.98**	**92.65**	**4.55**	**2.65**
	P1: PCPNet+CLA	**96.01**	70.82	68.88	79.79	92.15	7.09	5.36
	P2: PCPNet+ORACLE+CLA	88.96	87.20	79.14	86.92	95.54	5.97	3.69
	P3: PCPNet+CLA+DIS	9.24	60.10	8.01	14.14	82.20	51.6	6.04
	P4: SFCNet-P	88.47	**88.58**	**79.86**	**87.38**	**95.74**	**4.31**	**2.63**

Table 1. Quantitative results for robustness study and ablation study. Quantitative results with different distance thresholds, noise levels and components are recorded. Multiple pairs of $(noise, threshold)$ are used to study the user-defined parameters. We also compared our SFCNet-P model and SFCNet-D model with different combinations of the proposed modules. D and P are abbreviations for DGCNN and PCPNet. 1 to 4 indicate the experiment numbers.

Displacement learning. The following four pairs of contrast experiments show the improvement by combining displacement learning with classical binary classification learning: D1 versus D3, D2 versus D4, P1 versus P3 and P2 versus P4 as shown in Table 1. Displacement learning boosts not only the distance metrics, but also the classification metrics in most cases. However, the comparison between P1 and P3 clearly shows that when the oracle is disabled during the training phase, the displacement learning introduces instability into the framework, since both MLPs share the same backbone. By combining our oracle with displacement learning, our network achieves the best scores in terms of most metrics.

Backbone. We conducted two groups of experiments: D1-D4 for DGCNN backbone and P1-P4 for PCPNet backbone. For both backbones, our proposed approach performs best within the group. Generally speaking, PCPNet backbone has a better performance than DGCNN backbone, while the price is a much longer testing time for a given point cloud.

4.3. Comparisons

We compare our approach with PB-DGCNN [18] (retrained from scratch on our dataset for 50 epochs, since there is no released pre-trained model), ECNet [34] (using the released pre-trained model), PIE-Net [31] (using the released pre-trained model), together with two unsupervised methods: Feature Edge Estimation (FEE) provided by the CGAL Library [19] and Covariance Analysis (CA) [3]. Table 2 records the error metrics on our test set. We observe that both of our networks have smaller Chamfer distances and smaller *PtF* distances compared to other methods. SFCNet-P performs best on the various metrics. Figure 6 offers a visual comparison of all methods. FEE and CA are sensitive to user-defined parameters: we find it hard to define through trial-and-error parameters that fit all point clouds in our test set. Compared to PB-DGCNN, our SFCNet-D achieves similar scores for classification metrics while performing better in terms of distance, owing to the displacement learning. EC-Net seems to have more missing parts than our models and both distance criteria are worse than the ours. The results show that PIE-Net is not noise-resilient and the detected sharp feature points may distribute over the whole point cloud. While the paper mentions that it also predicts 3D offset vectors to relocate points onto edges, such a prediction step is commented out in the code and no pre-trained model contains such a relocation operation.

We evaluate the average inference time on our test set. FEE [19], implemented in C++, takes around 1.877s for 1 model with 10k points. All the other methods are implemented in Python, and the approximate test times are: 0.134s for CA [3], 1.26s for PB-DGCNN [18], 2.4s for EC-Net [34], 0.177s for PIE-Net [31], 0.174s for SFCNet-D and

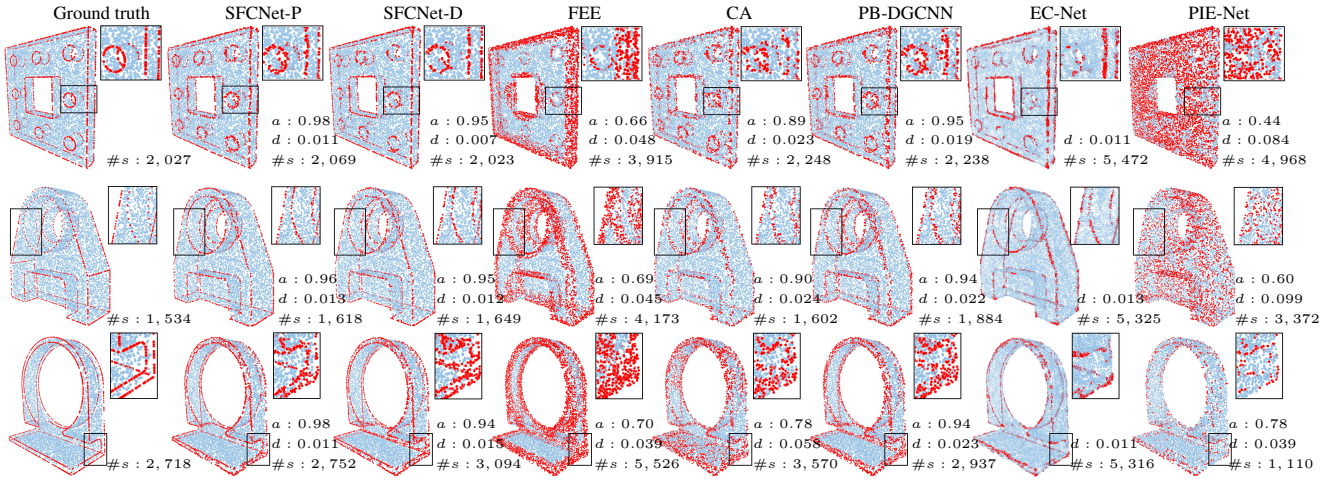

Figure 6. Visual comparisons with supervised and unsupervised methods on selected models from the test set. According to the closeups, our methods can detect more meaningful sharp feature points that are closed to the sharp features.

5s for SFCNet-P.

	Cla(%)					Dis($\times 10^{-2}$)	
	Rec	Pre	IoU	F1	Acc	Cha	*PtF*
CA [3]	67.18	63.06	43.50	57.58	78.98	11.80	8.23
FEE [19]	81.29	38.10	34.09	48.99	72.29	12.01	7.91
PB-D [18]	85.81	77.90	68.29	79.55	92.20	6.17	3.81
EC-N [34]	-	-	-	-	-	6.90	4.25
PIE-N [31]	48.87	45.11	20.32	32.47	62.86	16.65	14.07
SFCNet-D	79.65	82.68	67.65	78.98	92.65	4.55	2.65
SFCNet-P	**88.47**	**88.58**	**79.86**	**87.38**	**95.74**	**4.31**	**2.63**

Table 2. Quantitative comparison with unsupervised and deep learning sharp feature points detection methods on the ABC dataset. Note that we cannot compute classification metrics of EC-Net since the produced point clouds are upsampled.

4.4. Results on real scans

We evaluate our SFCNet-P model on real scanned 3D point clouds taken from the online Visionair repository [1]. These point clouds have never been seen during the training phase. Compared to the training dataset, we highlight three major differences in real scanned point clouds: (1) The cardinality of the point clouds (from 460k to 1 million) is substantially larger than that of our training dataset (10k); (2) The intrinsic noise in scanned point clouds originates from the laser, instead of being simulated randomly; (3) The models scanned by real-world scanners are often open, while all models of our training dataset are closed. Despite the above differences, our model predicts accurate sharp feature points from the real scanned point clouds with displacement learning, see Figure 7.

4.5. Limitations

Our current approach presents several limitations. First, the detected sharp feature points may not distribute uniformly over the sharp features. We intend to devise a loss function that favors the uniform distribution of inliers along sharp creases. However, the difficulty lies in the fact that we have no knowledge of continuous sharp feature graphs. Second, we classify points into two classes: sharp or smooth, instead of classifying them into instances. In future work, we wish to extend our framework to perform instance segmentation of sharp features, in order to cluster sharp feature points into several creases, possibly meeting at corners, darts, or cusps. To achieve this goal, the network needs to predict the number of sharp features, which is much more complicated than binary classification.

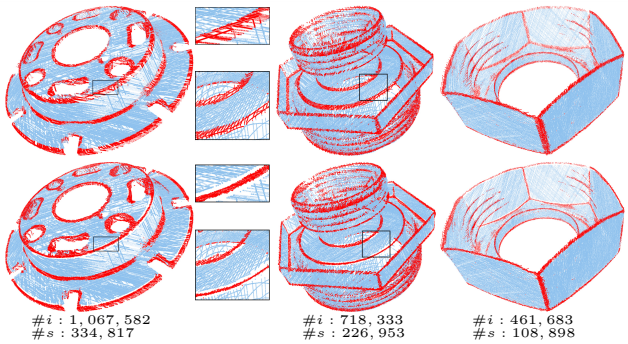

Figure 7. Scanned point clouds of Visionair repository. The first and the second rows show the detected predicted sharp feature points without and with displacement, respectively. $\#i$ and $\#s$ denote the number of input points and the number of detected sharp feature points, respectively.

5. Applications

We show two potential applications of our approach: 3D feature line extraction and 3D surface reconstruction. Nevertheless, our approach is not limited to these two applications. It can be further adapted to other applications such as

instance segmentation or feature graph extraction.

5.1. Feature line extraction

After consolidating the sharp feature points, we can apply primitive detection methods to extract parametric representations of sharp feature creases. We adapt RANSAC [7] algorithm to extract feature lines and the results are shown in Figure 8. The parameters are carefully tuned for all experiments. We observe that the extractions of consolidated sharp feature points are more robust and have a better quality than the one of the detected sharp feature points without displacement.

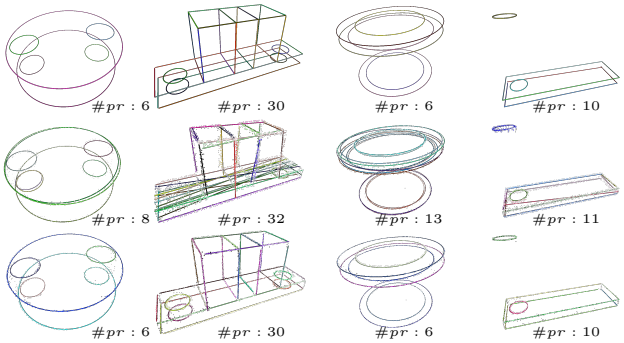

Figure 8. Feature line extraction results using RANSAC. From top to bottom: results of dense, uniformly sampled points on ground truth feature creases, results of the detected sharp feature points without displacement, results of the consolidated sharp feature points by our SFCNet-P model. $\#pr$ denotes the number of detected primitives. The ones without displacement detect many wrong primitives and there are more outliers (black points) left after the detection.

5.2. Surface reconstruction

Some surface reconstruction methods can benefit from consolidated point clouds. We utilize a recent learning-based surface reconstruction approach, referred to as DSE meshing [27]. DSE combines 3D Delaunay triangulations with learned local parametrizations to yield quality meshes, even if it may generate non-manifold edges. The advantage of this approach is that it does not require any normal information. We show the quality improvement of our approach by comparing the surface reconstruction results from the input 3D point clouds with the consolidated 3D point clouds. For generating a consolidated 3D point cloud, we first combine the input point cloud with the displaced sharp feature points, then 10k points are sampled using farthest point sampling to meet the input size requirement of the DSE.

Figure 9 depicts the reconstruction results before and after consolidation using our SFCNet-P model, in which we selected three 3D point clouds without noise from the test set in order to better compare the reconstruction quality around sharp creases.

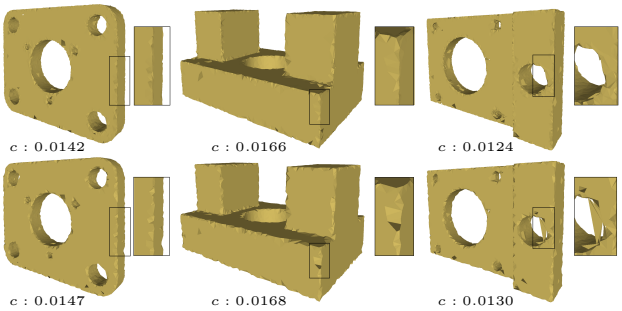

Figure 9. Reconstructed meshes using DSE meshing. First row: reconstruction from the consolidated 3D point clouds. Second row: reconstruction from the 3D input point clouds. c denotes the Chamfer distance between the reconstructed mesh and the ground truth mesh in the ABC dataset. DSE meshing is able to capture more details on sharp creases with consolidated point clouds and the Chamfer distance shows that the meshes are closer to ground truth compared to the original point clouds.

6. Conclusion

We contributed a novel deep learning based-framework devised to detect and consolidate sharp feature points from raw 3D point clouds. Compatible with existing backbones devised to extract features from 3D point clouds, our framework comprises two learnable modules: one module learns to predict binary smooth/sharp labels for all points, and the other module learns to regress the displacement vectors used for relocating sharp feature points onto sharp features. Our framework is capable of identifying and consolidating sharp features, altogether. Our approach is robust to intrinsic noises, thanks to the point-to-feature oracle, which performs data augmentation during the training phase, and displacement learning, which relocates sharp feature points onto their nearest sharp features. Our experiments demonstrate that our framework outperforms the state-of-the-art in terms of detection accuracy and distance criteria.

References

[1] Visionair advanced infrastructure for research. 7

[2] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, 1998. 1

[3] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *2015 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–8. IEEE, 2015. 2, 6, 7

[4] Onur Rauf Bingol and Adarsh Krishnamurthy. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9:85–94, 2019. 4

[5] Himeur Chems-Eddine, Lejemble Thibault, Pellegrini Thomas, Paulin Mathias, Barthe Loic, and Mellado Nicolas. Pcednet: A neural network for fast and efficient edge detection in 3d point clouds. *arXiv e-prints*, pages arXiv–2011, 2020. 2, 4

[6] Joel II Daniels, Linh K Ha, Tilo Ochotta, and Claudio T Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*, pages 123–136. IEEE, 2007. 2

[7] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 8

[8] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)*, 24(3):544–552, 2005. 2

[9] Ran Gal, Ariel Shamir, Tal Hassner, Mark Pauly, and Daniel Cohen-Or. Surface reconstruction using local shape priors. In *Eurographics Symposium on Geometry Processing*, pages 253–262, 2007. 1

[10] Jingyu Gong, Jiachen Xu, Xin Tan, Jie Zhou, Yanyun Qu, Yuan Xie, and Lizhuang Ma. Boundary-aware geometric encoding for semantic segmentation of point clouds. *arXiv preprint arXiv:2101.02381*, 2021. 1

[11] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM Siggraph 2007 papers*, pages 23–es. ACM, 2007. 1

[12] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018. 2, 3, 4

[13] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):1–12, 2013. 2

[14] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 1, 4

[15] Kai Wah Lee and Pengbo Bo. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering*, 3(2):102–111, 2016. 1

[16] David Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 37–49. Springer, 2004. 2

[17] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 2

[18] Marios Loizou, Melinos Averkiou, and Evangelos Kalogerakis. Learning part boundaries from 3d point clouds. *Computer Graphics Forum*, 39(5):183–195, 2020. 1, 2, 4, 6, 7

[19] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, 2010. 2, 6, 7

[20] Huan Ni, Xiangguo Lin, Xiaogang Ning, and Jixian Zhang. Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods. *Remote Sensing*, 8(9):710, 2016. 1

[21] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009. 1

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 4

[23] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *European conference on computer vision*, pages 103–118. Springer, 2020. 3

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[26] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet, and Bernard Ghanem. Assanet: An anisotropic separable set abstraction for efficient point cloud representation learning. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[27] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22–31, 2021. 8

[28] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum*, 39(1):185–203, 2020. 2

[29] Nader Salman, Mariette Yvinec, and Quentin Mérigot. Feature preserving mesh generation from 3d point clouds. *Computer graphics forum*, 29(5):1623–1632, 2010. 1

[30] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018. 2

[31] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems*, 33:20167–20178, 2020. 2, 4, 6, 7

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic

graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2, 3, 4

[33] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Sharp feature detection in point clouds. In *2010 Shape Modeling International Conference*, pages 175–186. IEEE, 2010. 2

[34] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 386–402, 2018. 2, 6, 7

[35] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 2