# Deep Point Cloud Normal Estimation via Triplet Learning

Weijia Wang, Xuequan Lu, Dasith de Silva Edirimuni, Xiao Liu, Antonio Robles-Kelly
Deakin University
{wangweijia, xuequan.lu, dtdesilva, xiao.liu, antonio.robles-kelly}@deakin.edu.au

## Abstract

*Normal estimation on 3D point clouds is a fundamental problem in 3D vision and graphics. Current methods often show limited accuracy in predicting normals at sharp features (e.g., edges and corners) and less robustness to noise. In this paper, we propose a novel normal estimation method for point clouds. It consists of two phases: (a) feature encoding which learns representations of local patches, and (b) normal estimation that takes the learned representation as input and regresses the normal vector. We are motivated that local patches on isotropic and anisotropic surfaces have similar or distinct normals, and that separable features or representations can be learned to facilitate normal estimation. To realise this, we first construct triplets of local patches on 3D point cloud data, and design a triplet network with a triplet loss for feature encoding. We then design a simple network with several MLPs and a loss function to regress the normal vector. Despite having a smaller network size compared to most other methods, experimental results show that our method preserves sharp features and achieves better normal estimation results on CAD-like shapes.*

## 1. Introduction

Point cloud data is a representation of 3D geometry which is applied in a wide range of fields, including autonomous driving, robotics, archaeology and architecture [34, 31]. Raw point clouds, typically captured by 3D sensors [31], consist of points that are unordered [24, 11] and lack connectivity information among them [10, 4]. In addition, the captured point clouds are usually corrupted by noise due to limitations on the scanning devices' precision [20, 33], and also lack normal information. This makes it challenging to directly use raw point clouds for visual computing tasks such as surface reconstruction, shape smoothing and segmentation [34, 4]. Estimating reliable normal information on the input point clouds has proven to be significant in achieving satisfactory results in such tasks [33, 13, 15, 22, 28].

Conventional principal component analysis (PCA) or Voronoi-based normal estimation methods for point clouds [11, 2, 1, 9] have limited accuracy in estimating normals at sharp features and are sensitive to noise. In recent years, learning-based methods have been proposed to improve performance in this area [5, 10, 4, 30, 13, 20]. Such methods typically learn geometrical features based on local structures, and involve complicated pre-processing tasks in the testing phase [4, 20]. Learning-based networks such as [5, 10, 30] are examples of networks that perform poorly in the presence of noise and in predicting normals at sharp features (e.g., edges). The method proposed by [20] shows more robust performance in the presence of noise yet consists of a two-step testing phase that requires a large amount of time to produce normal estimation results. Nesti-Net [4], while being a promising candidate, has a large network size and lengthy inference time.

In this paper, we introduce a novel normal estimation method for point clouds, to address those limitations. Our idea is to realise the normal estimation through representation (or feature) learning. To achieve this, we design a triplet network with a triplet loss to push the anchor sample to be far away from the negative sample and to be close to the positive sample. Inspired by PointNet [24], we construct a similar backbone capable of consuming unordered point sets directly based on the principle of a symmetric function (i.e., max-pooling). However, additional PointNet submodules such as the input and feature transformations, implemented using Spatial Transformer Networks (STNs), are not employed in our work. Also, we treat the local neighborhood of each point (i.e., central point) as a local patch or sample which becomes the input for the network. To define a triplet, we first select a local patch as the anchor, and regard another local patch with a small angle between their central point normals as the positive sample, and a third patch with large angle as the negative sample. We derive this based on the fact that two local patches on the same isotropic surface should have similar normals (i.e., normals of the corresponding central points) and representations. This phase encodes features which will be used as input for the normal estimation phase. We simply design multilayer

perceptrons (MLPs) as the normal estimation network and use it to regress the normal vector from the encoded features of a local patch. We also design a cosine similarity loss function to facilitate the training.

We respectively collect two datasets for training and testing, containing both CAD and non-CAD point clouds. Also, we test the effectiveness and scalability of our method on scanned CAD-like point clouds by simply using the same trained network. Experiments show that our method achieves better or comparable results to other normal estimation techniques on noisy CAD shapes, while less obvious performance is gained on smoother non-CAD shapes. Our method has several characteristics: (a) it performs very well on noisy CAD-like shapes (both synthetic and scanned) in terms of preserving sharp features, (b) it is robust to irregular sampling and less points, (c) it has a small network size of 10.42 MB and it completes normal estimation within a short amount of time (i.e., 55.6 seconds per 100,000 points), and (d) the single trained network can estimate normals for both CAD and non-CAD shapes.

## 2. Related Work

### 2.1. Normal Estimation for Point Clouds

As a commonly adopted traditional normal estimation approach, PCA [11] derives each point's normal by calculating the eigenvalues and eigenvectors of the covariance matrix of its neighbourhood. The eigenvector corresponding to the smallest eigenvalue is determined to be the normal at that point. Based on [11], several other approaches [14, 32, 23] have been proposed, considering the input point clouds' local features such as noise, curvature and sampling density. Nevertheless, such methods tend to smooth out sharp edges and corners that exist in the input shapes. To overcome these issues, other approaches such as Voronoi-based methods [2, 1, 9], minimising $L_0$ or $L_1$ norms [27, 3], adopting statistical-based methods [18, 33] and utilising low-rank matrices [35, 19, 21] have been proposed, which can better preserve sharp features for point clouds in the presence of noise.

In recent years, learning-based normal estimation methods have started to emerge. Only a few of these methods showcase the ability to preserve sharp features, handle noise robustly and generalise among different shapes. As the pioneer of such approaches, Boulch and Marlet [5] presented HoughCNN, in which 3D point clouds are mapped to a Hough space and, thereafter, a convolutional neural network (CNN) performs normal estimation on this representation. As 3D points need to be transformed to a 2D Hough space before being fed into CNNs, this method may discard important geometrical details. PointNet [24] which allows neural networks to directly consume 3D point clouds, was proposed for classification and segmentation. They utilised a max-pooling function, such that features are extracted without being affected by the ordering of input points. Thereby, 3D point clouds can be directly fed into the network for normal estimation. For example, Guerrero et al. introduced PCPNet [10] which adopts PointNet [24] as the backbone to encode local neighbourhoods on point clouds and regresses normals from them. Ben-Shabat et al. [4] introduced a mixture-of-experts network, known as Nesti-Net, that checks points' neighbourhoods in varying scales and selects the optimal one. Finally, it estimates normals using the corresponding specialised subnetwork. Similarly, Hyeon et al. [13] proposed a multi-scale k-nearest neighbour (kNN) network to robustly extract local features in preparation for normal estimation. Wang and Prisacariu [30] adopted an attentional module that softly selects neighbouring points for normal estimation. This network, known as NINormal, can thereby decide which neighbouring points are useful for preserving features in the shape. To further preserve feature normals, Lu et al. [20] classified points into two classes (feature and non-feature), and performed denoising on noisy point clouds in conjunction with normal estimation in their Deep Feature Preserving (DFP) method. Recently, Lenssen et al. [17] proposed Deep Iterative (DI), a graph neural network based approach that performs a weighted least squares plane-fitting of point neighbourhoods where the weights are iteratively refined by an adaptive kernel parameterised by the network.

### 2.2. Triplet Loss

The triplet loss function, which focuses on optimizing the relative distances of an anchor to a positive and negative sample, has been applied to a broad range of contexts. Triplet loss was mainly applied to 2D image processing tasks, such as image learning [29, 12, 16], face identification [25] and person matching [36]. Extending from 2D, triplet loss has also been employed for 3D geometric data processing in recent years. For example, [31] and [8] used triplet loss for point cloud matching, and [6] employed unsupervised triplet loss for point cloud classification. While triplet loss has proven to be effective for feature-learning tasks on 3D data, it has never been exploited and extended to normal estimation on 3D point clouds.

## 3. Method

Our normal estimation approach consists of two phases: (a) feature encoding that learns geometric features from input point clouds, and (b) normal estimation that regresses normals from the encoded features. In phase (a), we first define the local patch as a point (i.e., central point) with its neighboring points, and conduct pre-processing to mitigate pose and point number inconsistencies. We then construct triplets of patches and feed them to the feature encoding network (i.e., a triplet network). In phase (b), the latent

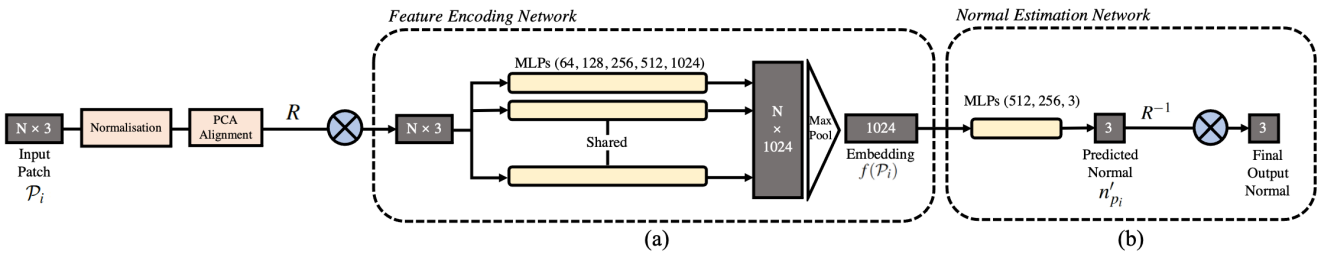


Figure 1. The overall architecture of our method, where the patch size N is empirically set to 500.

representations learned by phase (a) are consumed by the normal estimation network which outputs a predicted normal for the given patch (i.e., normal for the central point of that patch). Note that a rotation matrix is used in phase (a) for patch alignment, and its inverse matrix is further used in phase (b) for recovering back to the original space. Fig. 1 shows the overall architecture of our method, which is trained in a supervised manner.

### 3.1. Feature Learning

**Patch Pre-processing.** Before conducting the training process, we pre-process each point cloud into patches as inputs for our model. For any point cloud $\mathbf{P} = \{p_1, \ldots, p_n \mid p_i \in \mathbb{R}^3,\ i = 1, ..., n\}$, we define a local patch $\mathcal{P}_i$ centered at a point $p_i$ as:

$$\mathcal{P}_i = \{p_j \mid \|p_j - p_i\| < r\}, \tag{1}$$

where $p_j$ is any point within the patch $\mathcal{P}_i$ and $r$ is the radius of the ball centered at point $p_i$.

There are two issues associated with raw patches that make them unsuitable for effective learning: (a) they contain unnecessary degrees of freedom and rigid transformations, and (b) each patch's number of points might vary, which prohibits them from being grouped into input batches during the training phase [10, 34]. To address (a), we firstly translate the patch to the origin and normalise its size, such that $\mathcal{P}_i = (\mathcal{P}_i - p_i)/r$. Then, we align each patch's PCA vectors with the global space, which was inspired by [24, 34]. We conduct PCA alignment on the patch by rotating it using a rotation matrix $R$, which is computed using the principal components of the patch's covariance matrix. Its inverse matrix $R^{-1}$ is required later, during testing. Applying $R$ to $\mathcal{P}_i$ aligns the patch's last principal axis with the global $z$ axis, and the second principal axis with the global $x$ axis. As a result, the unnecessary degrees of freedom have been removed, leaving the patch invariant under rigid transformations. To solve (b), we select $k$ points from each patch to ensure the input patches are of a consistent size. For raw patches with more points than $k$, we randomly screen $k$ points; otherwise, we pad the input using existing points within the patch to make up $k$ points. We empirically set $k$ to be 500 and the ball radius $r$ to be 5% of the point cloud's bounding box diagonal.

**Triplet Generation.** As defined in [25], each data sample is treated as an anchor and needs to be paired with a positive and a negative sample in order to construct a triplet. We treat each input patch as an anchor patch $\mathcal{P}_i$, which is paired with a positive and negative patch. We denote the ground truth normal of the anchor patch's central point, $p_i$, by $n_{p_i}$. Thereafter, we define a positive patch $\mathcal{S}_i$ and negative patch $\mathcal{T}_i$ according to the anchor, as:

$$\mathcal{S}_i = \{s_j \mid \|s_j - s_i\| < r,\ \theta(n_{p_i}, n_{s_i}) \leq \theta_{th}\}, \tag{2}$$
$$\mathcal{T}_i = \{t_j \mid \|t_j - t_i\| < r,\ \theta(n_{p_i}, n_{t_i}) > \theta_{th}\}, \tag{3}$$

where $s_i$ and $t_i$ are respectively the central points of $\mathcal{S}_i$ and $\mathcal{T}_i$, and $n_{s_i}$ and $n_{t_i}$ are their corresponding ground-truth normals. $\theta_{th}$ is the angle threshold used for identifying positive and negative patches, which is empirically set to 20 degrees.

To ensure the feature encoder network can effectively learn sharp features such as edges on the input point clouds, we choose $s_i$ and $t_i$ as close to $p_i$ as possible. To do so, we start by trying to locate $s_i$ and $t_i$ within $r$ of $p_i$, and then gradually increase the search radius if either target cannot be found. By doing so, $s_i$ is more likely to lie on the same isotropic surface as $p_i$, while $t_i$ is more likely to lie on an adjacent surface without being too far from the edge. To ensure the consistency of each triplet, the anchor, positive and negative patches within each triplet are chosen from the same point cloud. The rotation matrix $R$ of $\mathcal{P}_i$ is then applied to $\mathcal{S}_i$ and $\mathcal{T}_i$ as well, to ensure the network learns consistently within each triplet.

**Triplet Loss**. Since input patch points are not organised in a specific order, the learning result may be affected by different permutations of the same set of points [10]. To address this issue, we utilise an architecture which comprises of several MLPs and a max-pooling layer, inspired by PointNet [24], as the backbone of our encoder. Within each triplet, we perform feature extraction by feeding each patch (a 500 $\times$ 3 vector) into the network, which aggregates the features as a 1024-dimensional latent vector using the max-pooling function. We perform the same operation for $\mathcal{P}_i$, $\mathcal{S}_i$ and $\mathcal{T}_i$. The loss for the encoded triplet is given

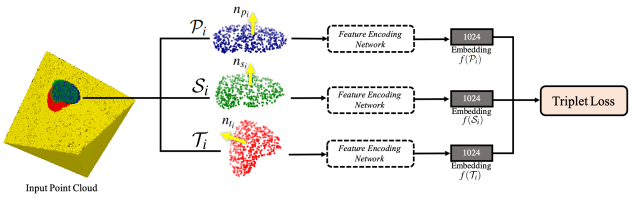


Figure 2. Illustration of triplet-based learning. The anchor, positive and negative patches are marked in dark blue, green and red respectively.

by:

$$L_E = \max\{\|f(\mathcal{P}_i) - f(\mathcal{S}_i)\|_2 - \|f(\mathcal{P}_i) - f(\mathcal{T}_i)\|_2 + m, 0\}, \tag{4}$$

where $f(\mathcal{P}_i)$, $f(\mathcal{S}_i)$ and $f(\mathcal{T}_i)$ are the latent representations of patch $\mathcal{P}_i$, $\mathcal{S}_i$ and $\mathcal{T}_i$, respectively. Here, $f(\cdot)$ is the encoder. We empirically set the margin $m = 0$. For pairwise distances of latent vectors, we use the $L_2$-norm regularisation during our training procedure. Intuitively, patches lying on isotropic and anisotropic surfaces should have small and large angles between their central point normals, respectively. Therefore, the representations of patches on an isotropic surface should be similar while patches of an anisotropic surface should be distinct. Triplet learning accomplishes this by bringing representations of patches on an isotropic surface closer together while pushing apart those from an anisotropic surface. Fig. 2 demonstrates the process of feature learning.

### 3.2. Normal Estimation

Next we aim to estimate the patch normal (i.e., the normal of the central point) from each patch's latent representation. This is accomplished by the normal estimation network which simply involves several MLPs. To train the normal estimation network, we firstly sample a patch and preprocess it in the same way as Sec. 3.1. Then, we feed the patch into the feature encoder to obtain its $1 \times 1024$ latent representation which then becomes the input for the normal estimation network. The output is a $3 \times 1$ normal vector.

As for the loss function for normal estimation, we define the cosine similarity between the estimated normal, $n'_{p_i}$, and ground truth normals within the patch, $\{n_{p_j} | p_j \in \mathcal{P}_i\}$ to facilitate the training. We set the exponent of the cosine of the angle to 8, for the sake of preserving sharp features. Inspired by [34], we also add a weighting scheme that takes into consideration the cosine similarity of the ground truth patch's central point normal and the neighbouring ground-truth normals. Therefore, the weighted loss function $L_N$ for the normal estimation network and its weight function $\theta(n_{p_i}, n_{p_j})$ are given by:

$$L_N = \frac{\sum_{p_j \in \mathcal{P}_i} (1 - (n'_{p_i} \cdot n_{p_j})^8) \theta(n_{p_i}, n_{p_j})}{\sum_{p_j \in \mathcal{P}_i} \theta(n_{p_i}, n_{p_j})}, \tag{5}$$

$$\theta(n_{p_i}, n_{p_j}) = \exp\left(-\frac{1 - n_{p_i} \cdot n_{p_j}}{1 - \cos(\sigma_s)}\right), \tag{6}$$

Here, $p_j$ are points in $\mathcal{P}_i$, $p_i$ is the patch's central point, $\theta(n_{p_i}, n_{p_j})$ is the weighting function based on the ground truth patch's central point normal and the normals of neighbouring points. $\sigma_s$ is the support angle which is set to 15 degrees by default. Since the input patch $\mathcal{P}_i$ is rotated by the rotation matrix $R$, we multiply the predicted normal by the inverse matrix $R^{-1}$ to get the normal for the patch's central point in the original space.

## 4. Experimental Results

### 4.1. Dataset

Our training dataset consists of 22 clean shapes: 11 CAD shapes and 11 non-CAD shapes (Fig. 3). Each clean shape contains 100,000 points that are sampled from the shape's ground-truth surfaces. The most distinct difference between CAD and non-CAD shapes is that CAD shapes contain sharp features (e.g., edges and corners) while non-CAD shapes are generally smoother. To ensure the network is trained equally on both types of shapes, we provide the same amount of training data from both CAD and non-CAD shapes. In addition, we also add noisy variants of each clean shape to the training dataset to improve the robustness of our network to noise. Each clean shape has 5 variants with different levels of Gaussian noise (0.25%, 0.5%, 1%, 1.5% and 2.5% of each clean shape's bounding box diagonal length, respectively). Therefore, we have a total of 132 ($22 \times 6$) shapes in our training dataset.

In both the feature encoding and normal estimation training, we sample 8,000 patches from each shape to ensure the network generalises well on sufficient data. We validate our network model on 3 point clouds (i.e., validation set in Fig. 4). Later, we tested on 9 synthetic shapes with various noise levels (the clean shapes are shown in Fig. 4) as well as 4 scanned point clouds, which are separately discussed in Sec. 4.3.

### 4.2. Implementation Details

We implemented our networks using PyTorch 1.8.0 and ran both training and testing on a NVIDIA GeForce RTX 3080 GPU with CUDA 11.3. Both feature encoding and normal estimation networks were trained using an SGD optimiser, with momentum of 0.9 and initial learning rate of 0.01. We train the feature encoding network for 5 epochs since it converges quickly, and train the normal estimation network for 50 epochs. During training, the learning rate is multiplied by a 0.1 factor if no improvement happens within 3 consecutive epochs.

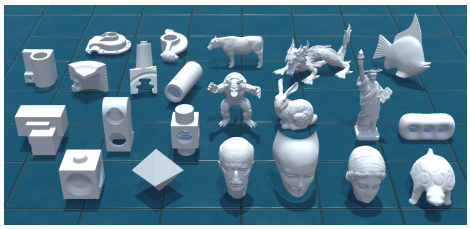

Figure 3. Training shapes.

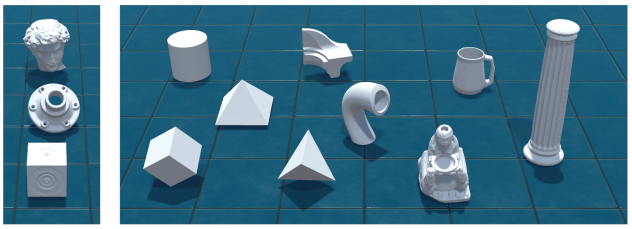

Figure 4. Validation shapes (3 on the left) and testing shapes (9 on the right).

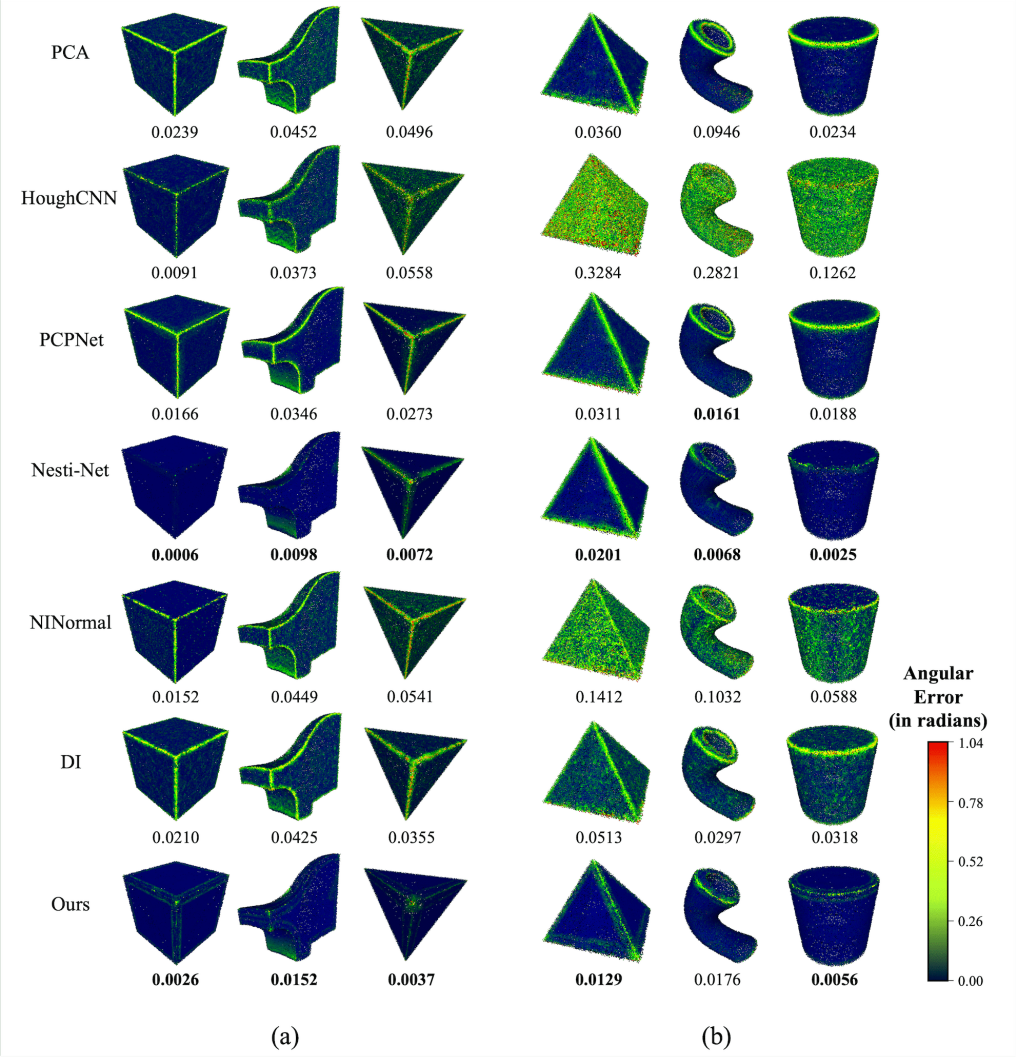


Figure 5. MSAE comparisons on 6 noisy CAD shapes with (a) 0.5% noise, and (b) 1% noise, with the two best results in bold in each column.

### 4.3. Comparisons

We compare our normal estimation results on the test dataset with PCA [11], HoughCNN [5], PCPNet [10], Nesti-Net [4], DI [17] and NINormal [30]. We also compare our performance with one of the most recently proposed methods, DFP [20], which is designed to perform normal estimation and point filtering in an iterative manner for noisy point clouds. During the comparison study, we retrained all methods, except for HoughCNN and DFP, on our training dataset and used recommended values of parameters for each respective method during testing, to ensure a fair comparison. HoughCNN creates synthetic point set

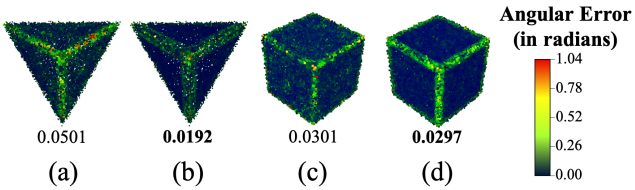


Figure 6. MSAE comparisons with DFP on Cube and Tetrahedron with 1% noise, where (a) and (c) are results from DFP, and (b) and (d) are our results.

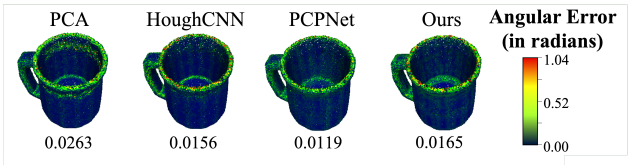


Figure 7. MSAE results on a non-CAD shape.

data and computes corresponding accumulators while DFP has a point classification step requiring labelled ground-truth points. Therefore, we were not able to adapt our training dataset to their training procedures. Also, methods such as [5, 4, 30] sometimes produce normals with wrong orientations, and sometimes the results are not normalised. Therefore, we flip and normalise the affected normals to guarantee fair comparison. We use mean squared angular error (MSAE) [20, 21] for evaluating normals (i.e., predicted normals versus ground-truth ones). Table 1 shows average MSAE values for the different methods over the testing shapes shown in Fig. 4 as well as 4 scanned CAD-like shapes with known ground truth normals. The average MSAE for CAD shapes includes the 6 shapes shown in Fig. 5 and the 4 scanned shapes in Fig. 8 while the average MSAE for non-CAD shapes is calculated using the results for the 3 non-CAD shapes shown in Fig. 4.

**Synthetic Point Clouds.** Fig. 5 demonstrates results on 6 CAD shapes with 0.5% and 1% random vertex displacement noise, added using MeshLab [7], respectively. Our method outperforms all other methods on Tetrahedron and Pyramid (the third and fourth columns in Fig. 5), and ranks the second on Cube, Fandisk and Cylinder (the first, second and sixth columns in Fig. 5). Our method generates outputs with less obvious gains on non-CAD shapes since it tends to over-sharpen features, as shown in Fig. 7.

We also demonstrate comparison results with DFP [20] which has two separately trained models for feature points and non-feature points. For fair comparisons, we use normal estimation results from the first iteration of DFP since it optimises point positions in further iterations. As can be seen from Fig. 6, our method outperforms DFP on noisy CAD shapes in terms of MSAE and inference time: DFP takes 1.69 hours to process every 10,000 points, while ours only requires 5.56 seconds. Since the two shapes in Fig. 6 have a much smaller number of points in them (10,171 points for Tetrahedron and 16,747 points for Cube), it also demonstrates the robustness of our method on less dense point clouds.

**Scanned Point Clouds**. We also test our method on 4 raw scanned point clouds with noise, where the ground truth normals are known, and compare our results to others. As shown in Fig. 8, our result outperforms others on the raw scanned point clouds with sharp features.

**Network Size and Inference Time.** It is worth noting that our method achieves the state-of-the-art performance with a relatively small network size and estimates normals at a relatively faster speed. Table 2 lists the comparison of all network sizes and inference time for all methods. Although Nesti-Net slightly outperforms our results on 4 CAD shapes listed in Fig. 5, it utilises the largest network among all methods. Despite DI having the most lightweight network, it cannot handle noise and sharp features well. Regarding the inference time, although NINormal and DI have shorter inference times, they can hardly preserve sharp features on noisy point clouds.

In addition, our method is more robust to irregular sampling and less points. Please refer to our supplementary document for additional results.

### 4.4. Ablation Study

**Normal Estimation.** During the normal estimation phase, we introduced a cosine similarity exponent of 8 in Eq. (5) as it effectively preserves sharp features on noisy shapes compared to others. We evaluate this over our validation set in Fig. 4 (0.5% Gaussian noise) using different exponents. As shown in Fig. 9, the mean MSAE value is minimum when the exponent is set to 8.

**Feature Encoding Network.** Alternatively, we can directly regress the normal vector using a PointNet-like architecture (i.e., without the feature encoding network). As illustrated in Fig. 10, utilising triplet loss to train our feature encoding network is effective in maintaining sharp edges and thus achieves a smaller MSAE value.

## 5. Conclusion

In this paper, we presented a novel deep learning normal estimation method for 3D point cloud data. Our method consists of two phases. In phase (a) the feature encoder creates latent representations of local point cloud patches through optimising relative distances within triplets. In phase (b) these representations are consumed by the normal estimation network which regresses the normals for the central points in the patches using MLPs. Our method is shown to effectively estimate normals in the presence of noise, especially for sharp features (e.g., edges), achieving state-of-the-art results.

| Method | PCA | HoughCNN | PCPNet | Nesti-Net | NINormal | DI | Ours |
|---|---|---|---|---|---|---|---|
| CAD | 0.0400 | 0.1043 | 0.0297 | 0.0249 | 0.0696* | 0.0341 | **0.0165** |
| Non-CAD | **0.2553** | 0.3425 | 0.3034 | 0.3101 | 0.2587 | 0.2631 | 0.3182 |

Table 1. Average MSAE of each method on shapes in the testing set (Fig. 4) and the scanned shapes (Fig. 8). *For NINormal, the average value does not include scanned shapes in Fig. 8 as it requires a fixed input of 100,000 points.

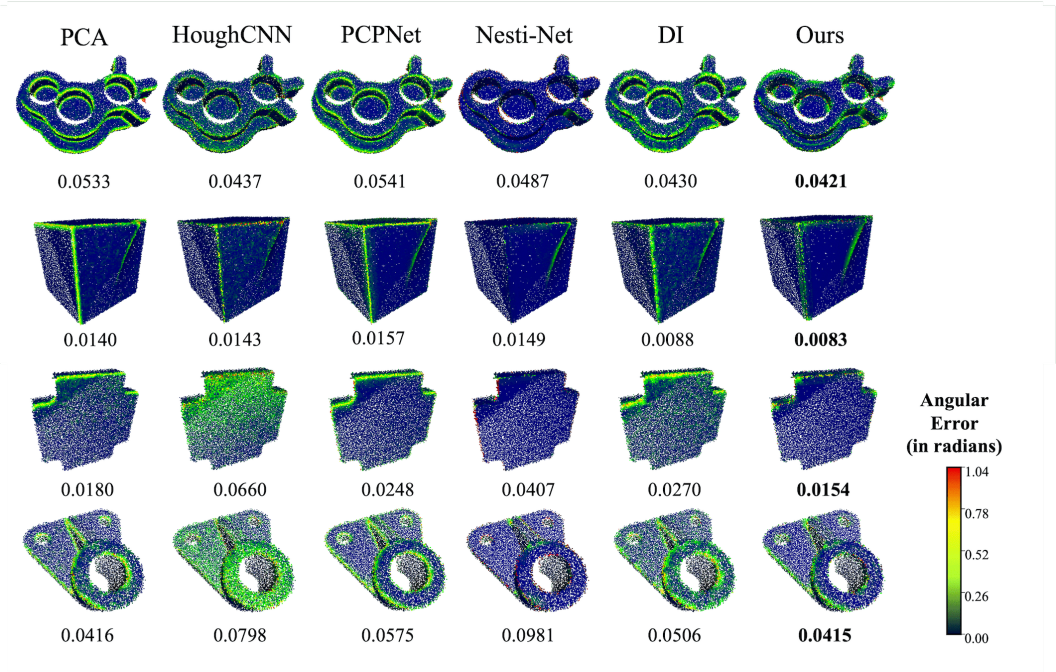


Figure 8. MSAE comparisons on 4 raw scanned point clouds (with ground truth normals).

| Method Name | HoughCNN | PCPNet | Nesti-Net | NINormal | DI | DFP | Ours |
|---|---|---|---|---|---|---|---|
| Network Size (in MB) | 111.6 | 85.4 | 2020 | 39.5 | **0.037** | 268.4 | 10.42 |
| Inference time (s) | 74.67 | 227.33 | 1234.5 | **2.7** | 3.7 | 60840 | 55.6 |

Table 2. Network size comparison among different approaches and inference time comparison (seconds per 100K points).

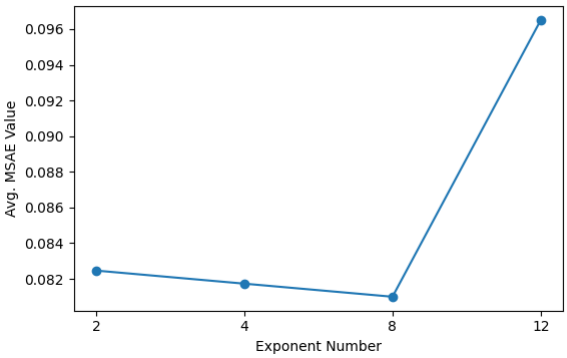


Figure 9. Ablation study on the choice of exponent in Eq. (5) over the validation dataset (0.5% Gaussian noise).

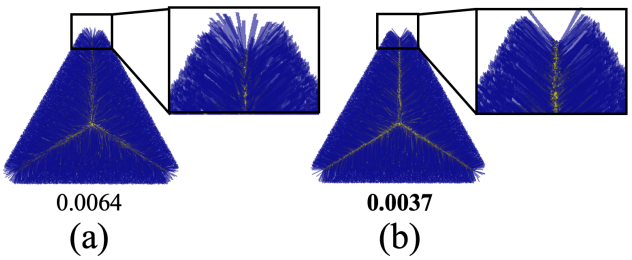


Figure 10. Visualised normals and MSAE (a) without, and (b) with the feature encoder.

## References

[1] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 39–48, Goslar, DEU,

2007. Eurographics Association. 1, 2

[2] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, page 39–48, New York, NY, USA, 1998. Association for Computing Machinery. 1, 2

[3] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. $l_1$-Sparse Reconstruction of Sharp Point Set Surfaces. *ACM Trans. Graph.*, 29(5), Nov. 2010. Place: New York, NY, USA Publisher: Association for Computing Machinery. 2

[4] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds Using Convolutional Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10104–10112, 2019. 1, 2, 5, 6, 10

[5] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35(5):281–290, 2016. 1, 2, 5, 6, 10

[6] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Transductive Zero-Shot Learning for 3D Point Cloud Classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020. 2

[7] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, Guido Ranzuglia, et al. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy, 2008. 6

[8] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[9] T.K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: a comparison study for a voronoi based method. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 39–46, 2005. 1, 2

[10] 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. 1, 2, 3, 5, 10

[11] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2):71–78, July 1992. 1, 2, 5, 10

[12] Junshi Huang, Rogerio Feris, Qiang Chen, and Shuicheng Yan. Cross-Domain Image Retrieval with a Dual Attribute-Aware Ranking Network. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1062–1070, 2015. 2

[13] Janghun Hyeon, Weonsuk Lee, Joo Hyung Kim, and Nakju Doh. NormNet: Point-wise normal estimation network for three-dimensional point cloud data. *International Journal of Advanced Robotic Systems*, 16(4):1729881419857532, 2019. tex.eprint: https://doi.org/10.1177/1729881419857532. 1, 2

[14] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. In *2009 IEEE International Conference on Robotics and Automation*, pages 3206–3211, 2009. 2

[15] Florent Lafarge and Pierre Alliez. Surface Reconstruction through Point Set Structuring. *Computer Graphics Forum*, 32(2):225–234, May 2013. Publisher: Wiley. 1

[16] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3278, 2015. 2

[17] J. Lenssen, C. Osendorfer, and J. Masci. Deep iterative surface normal estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11244–11253, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. 2, 5, 10

[18] Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Shiyao Jin. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34(2):94–106, 2010. 2

[19] Xiuping Liu, Jie Zhang, Junjie Cao, Bo Li, and Ligang Liu. Quality point cloud normal estimation by guided least squares representation. *Computers & Graphics*, 51:106–116, 2015. 2

[20] Dening Lu, Xuequan Lu, Yangxing Sun, and Jun Wang. Deep feature-preserving normal estimation for point cloud filtering. *Computer-Aided Design*, 125:102860, 2020. 1, 2, 5, 6

[21] Xuequan Lu, Scott Schaefer, Jun Luo, Lizhuang Ma, and Ying He. Low rank matrix approximation for 3d geometry filtering. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2020. 2, 6

[22] Xuequan Lu, Shihao Wu, Honghua Chen, Sai-Kit Yeung, Wenzhi Chen, and Matthias Zwicker. Gpf: Gmm-inspired feature-preserving point set filtering. *IEEE transactions on visualization and computer graphics*, 24(8):2315–2326, 2017. 1

[23] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, page 322–328, New York, NY, USA, 2003. Association for Computing Machinery. 2

[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 (CVPR)*, July 2017. 1, 2, 3

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2, 3

[26] Andrés Serna, B. Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. Paris-rue-madame database: a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. pages 1–4, 03 2014. 10

[27] Yujing Sun, Scott Schaefer, and Wenping Wang. Denoising point sets via L0 minimization. *Computer Aided Geometric Design*, 35-36:2–15, 2015. 2

[28] Jinxi Wang, Jincen Jiang, Xuequan Lu, and Meili Wang. Rethinking point cloud filtering: A non-local position based approach. *arXiv preprint arXiv:2110.07253*, 2021. 1

[29] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014. 2

[30] Zirui Wang and Victor Prisacariu. Neighbourhood-insensitive point cloud normal estimation network. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. 1, 2, 5, 6, 10

[31] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 630–646. Springer, 2018. 1, 2

[32] Mincheol Yoon, Yunjin Lee, Seungyong Lee, Ioannis Ivrissimtzis, and Hans-Peter Seidel. Surface and normal ensembles for surface reconstruction. *Computer-Aided Design*, 39(5):408–420, 2007. 2

[33] Zhiqiang Yu, Taiyong Wang, Ting Guo, Hongbin Li, and Jingchuan Dong. Robust point cloud normal estimation via neighborhood reconstruction. *Advances in Mechanical Engineering*, 11(4):1687814019836043, 2019. 1, 2

[34] Dongbo Zhang, Xuequan Lu, Hong Qin, and Ying He. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2015–2027, 2021. 1, 3, 4

[35] Jie Zhang, Junjie Cao, Xiuping Liu, Jun Wang, Jian Liu, and Xiquan Shi. Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics*, 37(6):697–706, 2013. 2

[36] Yingying Zhang, Qiaoyong Zhong, Liang Ma, Di Xie, and Shiliang Pu. Learning incremental triplet margin for person re-identification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), July 2019. 2

# A. Appendix

In this appendix, we provide more details regarding our methodology and additional experimental results. In Sec. B, we give a detailed illustration of the feature encoding and normal estimation networks. In Sec. C, we show that setting the patch size to 5% of the bounding box diagonal and sampling 500 points per patch are the empirically optimal choices. Then, from Sec. D onward, we demonstrate additional experimental results in comparison to PCA [11], HoughCNN [5], PCPNet [10], DI [17] and Nesti-Net [4] in order to demonstrate our method's ability to preserve sharp features in the presence of noise. We do not compare with NINormal [30] as NINormal requires a fixed input of 100,000 points. Therefore, we have excluded this method in the following comparisons.

# B. Network Architecture

Fig. 11 and Fig. 12 demonstrate the structures of the feature encoding network and normal estimation network, respectively. The training of the normal estimation network utilises the output from the trained feature encoding network.

# C. Patch Sizes

To find the optimal patch size, we test 6 sizes in total: 1%, 2%, 3%, 4%, 5% and 6% of the shape's bounding box diagonal respectively, and select 20, 50, 100, 250, 500 and 500 points per patch for each respective patch size. We use the test set in Fig. 13, and show the MSAE results in Table 3. The results demonstrate that the optimal choice of patch size is 5% of the bounding box diagonal with 500 points per patch, since that achieves the minimum MSAE overall.

# D. Visual Comparison on Scanned Shapes

We show the visual effects on 3 additional scanned shapes with noise. As can be seen from Fig. 14, our result produces smoother surfaces and sharper edges.

In addition, we demonstrate the visual results for a scanned street scene coming from the Paris-rue-Madame Database [26]. We did not compare with DI [17] as it encounters sudden issues when attempting to process point clouds with over 100K points. As shown in Fig. 15, our method smooths the ground while preserving sharp edges such as car roofs.

# E. Performance on Irregularly Sampled Point Clouds

Irregular sampling may appear in point clouds, which is challenging for normal estimation methods. To demonstrate our method's robustness in handling irregularly sampled points, we test methods on 3 point cloud shapes with irregular point distributions. As shown in Fig. 16, within the same shape, some surfaces have significantly less points compared to other surfaces. Our method still outperforms others at the normal estimation task for such shapes, with lower MSAE results which are listed in Fig. 17.

# F. Performance on Point Clouds with Less Points

While we train our network using point clouds with 100,000 points in each shape, our network also performs well on point clouds with less points. In Fig. 18, we demonstrate the MSAE comparisons for normal estimation results on a 20,000-point Pyramid and an 10,000-point Tetrahedron, where our approach outperforms other methods.

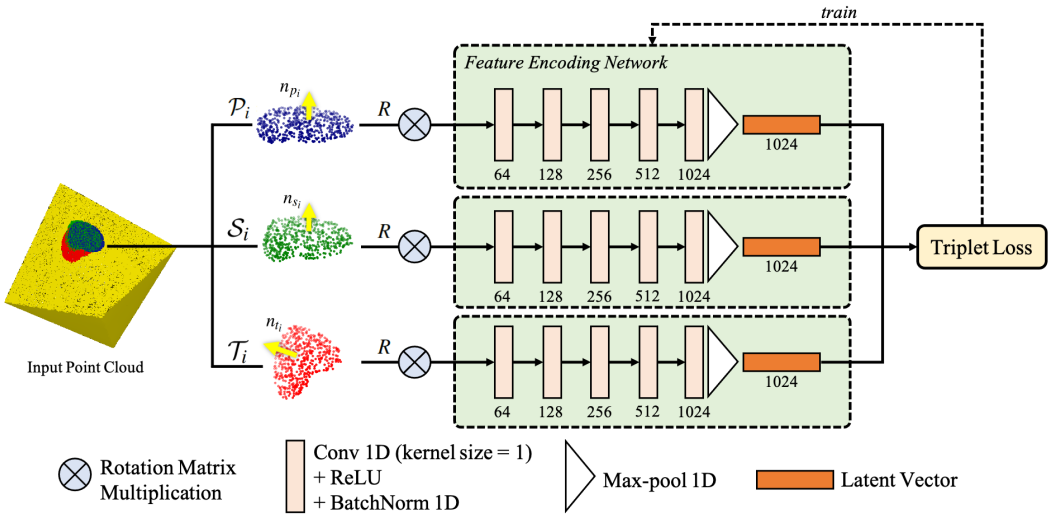


Figure 11. Our feature encoding network (with light green background). The triplet shares the same network (and weights). Note that the normal of each patch is used for triplet selection only.

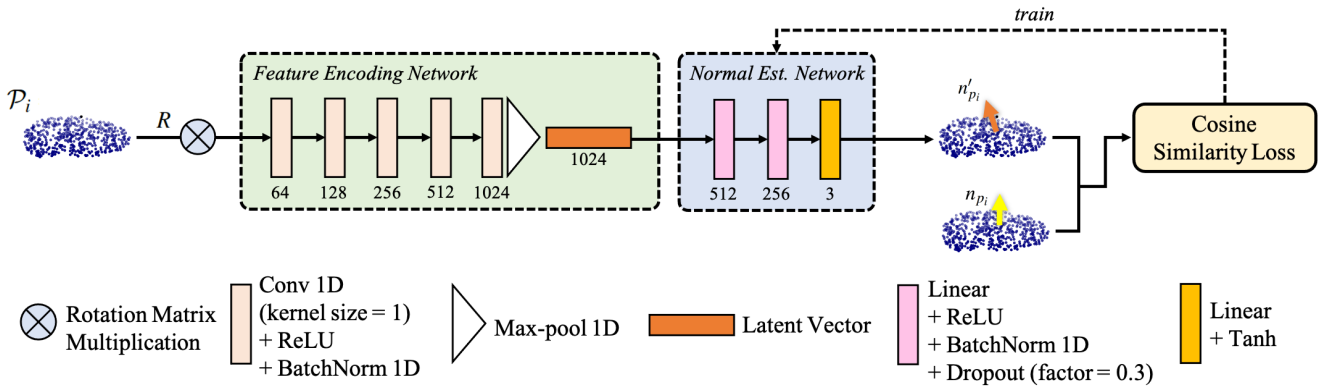


Figure 12. Our normal estimation network (with light blue background), which begins by consuming latent vectors produced by the trained feature encoding network.

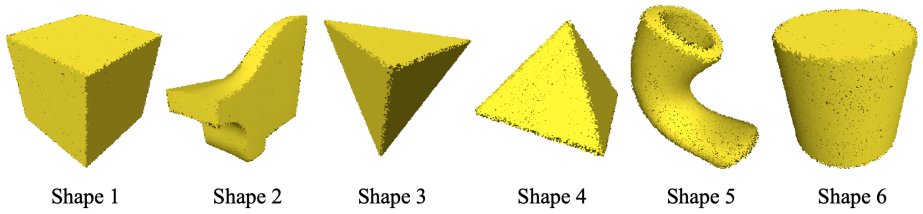


Figure 13. Shapes being tested for different patch sizes, where shape 1, 2 and 3 are corrupted with 0.5% noise and shape 4, 5 and 6 are corrupted with 1% noise.

| Patch Size | Shape 1 | Shape 2 | Shape 3 | Shape 4 | Shape 5 | Shape 6 |
|---|---|---|---|---|---|---|
| 1% | 0.3127 | 0.0908 | 0.0675 | 1.0184 | 1.1039 | 1.1417 |
| 2% | 0.0195 | 0.0367 | 0.0292 | 0.0455 | 0.0854 | 0.0320 |
| 3% | 0.0068 | 0.0195 | 0.0076 | 0.0235 | 0.0575 | 0.0141 |
| 4% | 0.0040 | 0.0165 | 0.0052 | 0.0200 | 0.0313 | 0.0082 |
| 5% | **0.0026** | **0.0152** | **0.0037** | **0.0129** | 0.0176 | **0.0056** |
| 6% | 0.0043 | 0.0244 | 0.0061 | 0.0205 | **0.0154** | 0.0064 |

Table 3. MSAE results on the test set shown in Fig. 13 with different patch sizes.

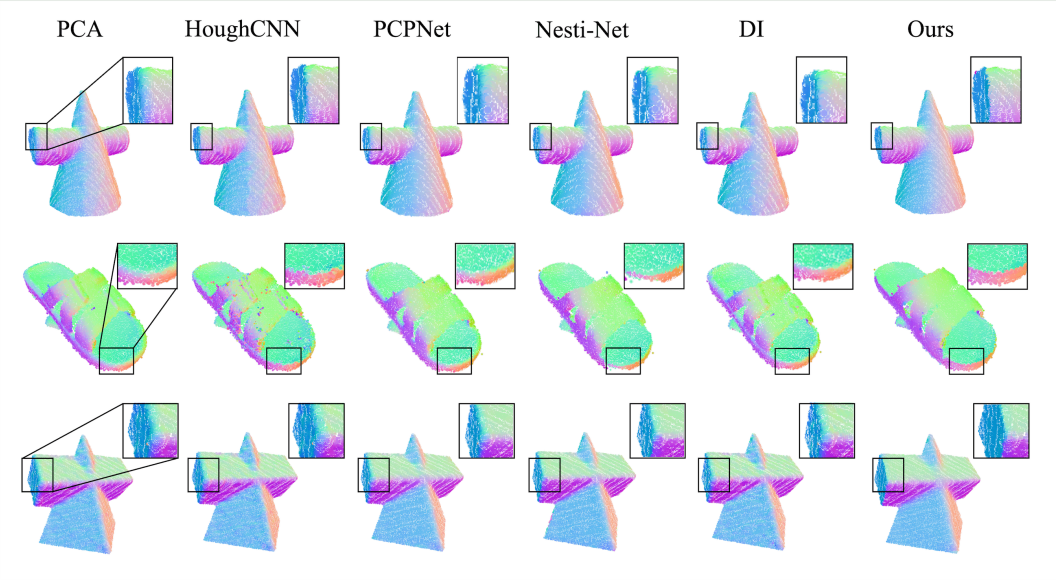


Figure 14. Visual comparison on 3 additional raw scanned shapes, where our method preserves sharp edges better.

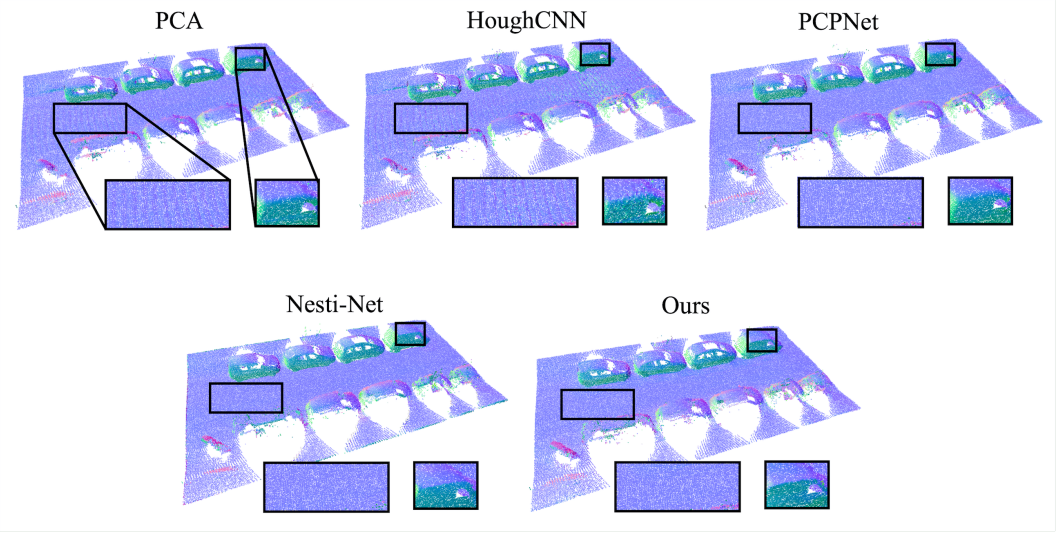


Figure 15. Visual comparison on a scanned street scene.

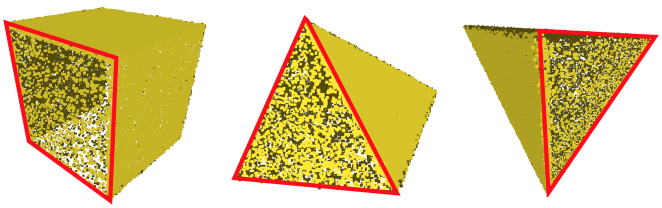

Figure 16. 3 point cloud shapes with irregular point sampling. Surfaces in red frames have significantly less points compared to other faces. Each shape is corrupted with 0.5% noise.

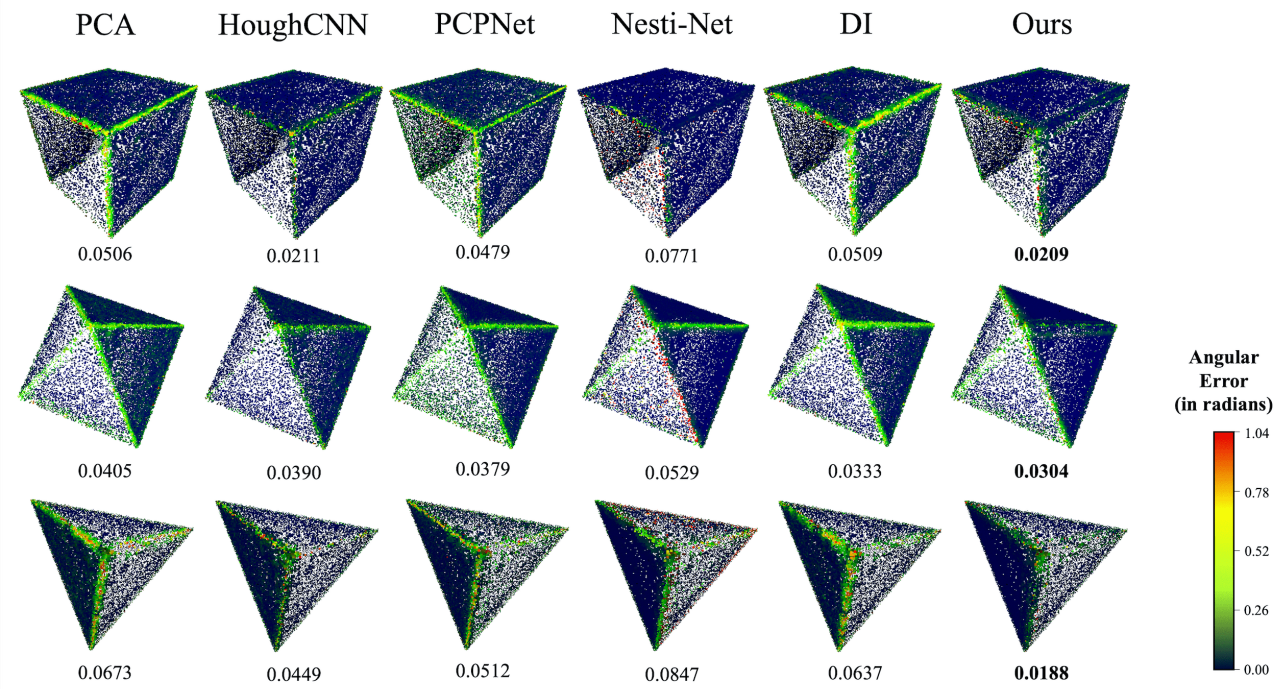


Figure 17. MSAE comparisons on the 3 shapes in Fig. 16.

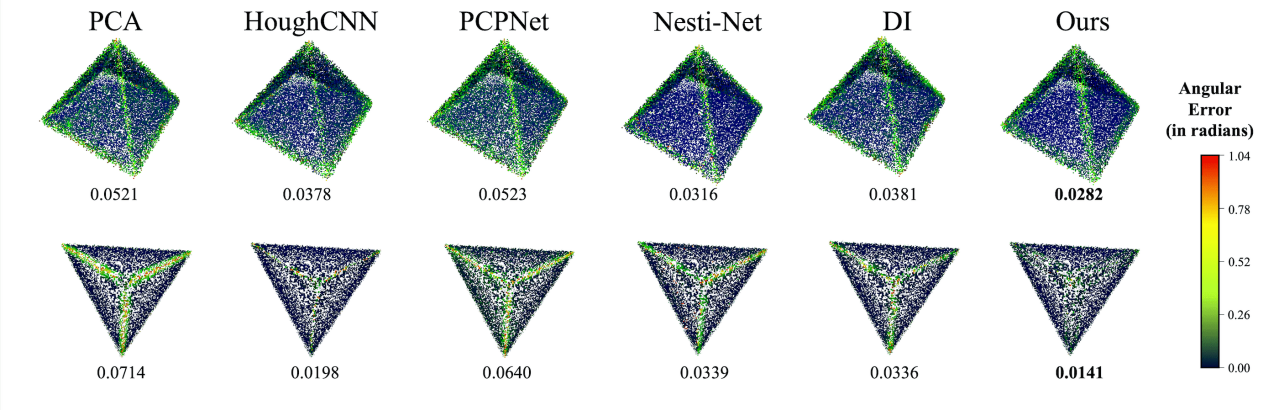


Figure 18. MSAE comparisons on 2 shapes with less points.