

# Contrastive Learning for Joint Normal Estimation and Point Cloud Filtering

Dasith de Silva Edirimuni, Xuequan Lu, *Senior Member, IEEE*, Gang Li, *Senior Member, IEEE*,  
and Antonio Robles-Kelly, *Senior Member, IEEE*

**Abstract**—Point cloud filtering and normal estimation are two fundamental research problems in the 3D field. Existing methods usually perform normal estimation and filtering separately and often show sensitivity to noise and/or inability to preserve sharp geometric features such as corners and edges. In this paper, we propose a novel deep learning method to jointly estimate normals and filter point clouds. We first introduce a 3D patch based contrastive learning framework, with noise corruption as an augmentation, to train a feature encoder capable of generating faithful representations of point cloud patches while remaining robust to noise. These representations are consumed by a simple regression network and supervised by a novel joint loss, simultaneously estimating point normals and displacements that are used to filter the patch centers. Experimental results show that our method well supports the two tasks simultaneously and preserves sharp features and fine details. It generally outperforms state-of-the-art techniques on both tasks. Our source code is available at <https://github.com/ddsediri/CLJNEPCF>.

**Index Terms**—Point cloud filtering, normal estimation, contrastive learning, machine learning.

## 1 INTRODUCTION

POINT clouds have numerous applications as they provide a natural representation of 3D geometric information. They have seen applications in fields such as autonomous driving, robotics, 3D printing and urban planning [1], [2], [3], [4]. Captured using 3D sensors, point clouds consist of unordered points which lack connectivity information between individual points. The captured point cloud information may be corrupted with noise. Therefore, one fundamental research problem is point cloud filtering, also known as denoising. Another fundamental task is normal estimation at individual points. Together, they facilitate other tasks such as 3D rendering and surface reconstruction.

Conventional normal estimation methods, such as Principal Component Analysis (PCA) and its variants [5], [6], [7], [8] and Voronoi diagram based approaches [9], [10], [11], perform poorly when estimating the normals at sharp features such as corners or edges and show high sensitivity to noise. To address these issues, a number of learning based methods have been recently proposed such as Deep Feature Preserving (DFP) [12] and Nesti-Net [13]. However, they have large network sizes and therefore are typically slow. Methods such as AdaFit [14] and Deep Iterative (DI) [15] offer more lightweight solutions that perform admirably, but still show less robust results at higher noise levels.

Point cloud filtering can be classified into two main types: normal based methods [12], [16], [17], [18] and position based methods [19], [20], [21], [22]. The former utilizes normal information at a given point in order to apply a position update algorithm [12], while the latter does not

require normal information and relies solely on position information. Among position based methods, a common issue is the inability to preserve sharp features during the filtering process while normal based methods rely heavily on normal accuracy. Learning based approaches seek to resolve this. In particular, Pointfilter [22], performs effectively at preserving sharp feature information on CAD-like shapes yet fails to generalize to large scenes. Methods such as PointCleanNet [21] and TotalDenoising [23] also perform sub-optimally, tending to smear sharp features.

In this paper, we propose a novel method capable of simultaneously inferring point normals and displacements while maintaining robustness to noise. Our method comprises of a feature encoder capable of generating latent representations of patches based on patch similarity and a regressor capable of inferring point normals and displacements simultaneously. We introduce a 3D patch based contrastive learning framework to train the feature encoder which employs noise corruption as an augmentation technique, allowing the encoder to identify the sharp geometric features of the underlying clean patch despite different levels of noise corruptions. The regressor consumes the latent representation of a patch and outputs the point normal and the displacement required to filter the central point of that patch. To train the regressor, we introduce a novel loss function that jointly penalizes inferred point position error and normal estimation error by exploiting the relationship between a point's position and normal. We intuitively assume that a filtered point's normal should correspond to a ground truth point's normal if this ground truth point first corresponds to that filtered point in position, thus leading to the relationship between filtering and normal estimation. The main contributions of this paper are as follows.

- We develop a novel framework capable of inferring both points' displacements and normals simultaneously by introducing a loss function capable of con-

• D. de Silva Edirimuni, X. Lu, G. Li and A. Robles-Kelly are with the School of Information Technology, Deakin University, Warrnambool, Victoria, 3216, Australia (e-mail: {dtdesilva, xuequan.lu, gang.li, antonio.robles-kelly}@deakin.edu.au). A. Robles-Kelly is also with the Defense Science and Technology Group, Australia.

Manuscript received Month Day, Year; revised Month Day, Year.  
(Corresponding author: Xuequan Lu.)

straining both filtering and normal estimation tasks. This joint loss penalizes both position regression error and normal estimation prediction error and allows the network to learn both filtered displacements and point normals.

- We introduce 3D patch based contrastive learning to generate effective patch-wise representations.

We conduct extensive experiments and demonstrate that our method, in general, outperforms state-of-the-art normal estimation and filtering techniques.

## 2 RELATED WORK

**Normal estimation.** In its earliest incarnation, normal estimation was based on Principal Component Analysis (PCA) [5]. Several variants of this initial PCA method have also been proposed [6], [7], [8]. Thereafter, approaches based on Voronoi cells were used to reconstruct surfaces while preserving sharp features and estimating normals [9], [10], [11]. Recently, Lu et al. [24] proposed a normal estimation method based on a Low Rank Matrix Approximation (LRMA) algorithm. Additionally, methods such as [25], [26], [27], [28] utilized point statistics and clustering to determine point normals.

**Normal estimation (learning-based).** One of the first learning models for normal estimation, HoughCNN, employs a voting mechanism for estimating normals. They utilize a local patch representation in Hough space that can be consumed by a CNN [29]. However, with the advent of PointNet [30] and PointNet++ [31], newer methods have been proposed that directly consume point sets. PCPNet is one such example, which consumes point cloud patches at multiple scales [32]. Similarly, Nesti-Net consumes patches at multiple scales but also employs multiple sub-networks, Mixture-of-Experts, that specialize in estimating normals at these scales [13]. Wang and Prisacariu introduced NINormal, a self-attention based normal estimation scheme [33] while Lu et al. proposed Deep Feature Preserving (DFP), a two step mechanism that classifies points into feature and non-feature points and, subsequently, estimates their normals based on this classification [12]. Finally, several deep learning methods based on weighted least squares plane fitting have been proposed [14], [15], [34]. While these methods focus on accurately determining unoriented normals, the work of Wang et al. [35] focuses on estimating point normals and their orientations.

**Point cloud filtering.** Traditional filtering applications center around Moving Least Squares (MLS) approaches [36], [37]. Alexa et al. [38] built on MLS techniques to minimize the approximation error of denoised point set surfaces. These methods perform poorly on point sets with sharp features, an issue that Adamson and Alexa [39] and Guennebaud and Gross [40] aimed to tackle. Lipman et al. developed the Locally Optimal Projection (LOP) operator which does not depend on a local data parametrization such as a local normal or tangent plane [19]. This projection operator was further enhanced by Huang et al. and Preiner et al., who proposed a Weighted LOP (WLOP) [20] and Continuous LOP (CLOP) [41], respectively. The main drawback to these MLS and LOP based techniques is their inability to preserve sharp features. Oztireli, Guennebaud and Gross [16]

proposed Robust Implicit Moving Least Squares [16] which improves the filtering ability to preserve sharp features but relies heavily on the accuracy of normal information. Lu et al. proposed a point cloud filtering scheme based on normals estimated by their LRMA algorithm [24]. Remil et al. reformulated point cloud filtering as a global, sparse optimization problem which is solved using Augmented Lagrangian Multipliers [42].

**Point cloud filtering (learning-based).** PointProNets used a CNN which consumes noisy height-maps and returns filtered ones [43]. EC-Net employed a supervised scheme for edge aware filtering and upsampling [44]. PCN uses a  $L_1$  norm loss based network to remove outliers and  $L_2$  norm loss based network to filter points [21]. Pointfilter takes into account local structure by considering points and their ground-truth normals, during training time, to infer filtered positions [22]. DFP [12] employs the position update mechanism of [24] to filter points based on the estimated normals. ScoreDenoise (SD) models a noisy point cloud's underlying surface with a 3D distribution supported by 2D manifolds and estimates the score for the gradient of the noise convolved distribution [45]. TotalDenoising (TD) offers an unsupervised learning alternative to the aforementioned supervised schemes [23].

**Contrastive learning.** Recently, we have seen the increased use of contrastive learning in generating faithful representations based on similarity between inputs. Self-supervised learning that maximizes agreement between similar inputs was first proposed by Becker and Hinton [46]. Thereafter, contrastive learning was further exploited to learn lower dimensional representations of high dimensional image data by the work of Hadsell, Chopra and LeCun [47]. Chen et al. utilized more recent neural network architectures and data augmentation methods in their SimCLR method [48]. Although initially designed for 2D image processing tasks, contrastive learning is now seeing applications in 3D representation learning [49], [50], [51], [52] and for specific point cloud processing tasks such as shape completion, segmentation and scene understanding [53], [54], [55]. However, it has never before been explored in terms of the problems of normal estimation and point cloud filtering, which we focus on in this work.

## 3 BACKGROUND AND MOTIVATION

In this section, we look at the motivation for our contrastive learning based joint normal estimation and filtering method.

### 3.1 Patch-based contrastive learning

As mentioned earlier, contrastive learning has emerged as an effective method of generating latent representations of inputs such as images or point clouds based on similarity between augmented pairs of inputs which are seen by the network during training [48]. The work of Xie et al. [56] extended this method to 3D point clouds. Crucially, their work focuses on generating representations of entire point clouds, i.e., they use a global approach. However, as Guerrero et al. [32] point out, normal estimation at a given point relies on the local structure of the point neighborhood rather than the global structure of the entire point cloud. This is also true

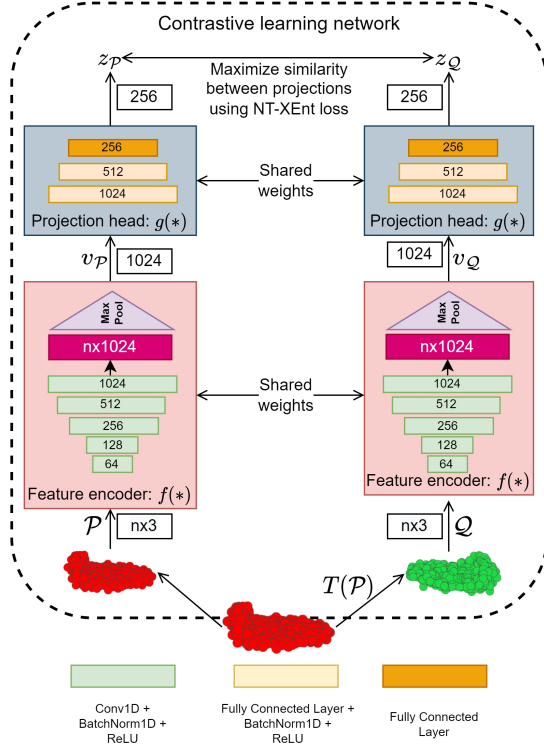


Fig. 1. 3D patch based contrastive learning.

for the problem of point cloud filtering [21] as effective filtering mechanisms must preserve sharp feature information locally. This motivates our approach of developing a patch-based contrastive learning mechanism where noise corruption of input patches is used as an augmentation to develop different views of the same underlying clean patch. Thereafter, we employ the Normalized Temperature-scaled Cross Entropy (NT-XEnt) loss function detailed in Sec. 4.3 which promotes similarity of generated latent representations for a given positive pair of augmented patches. Inspired by the work of [48], we do not explicitly sample negative pairs as the remaining augmented pairs within a batch can be used for this purpose. Furthermore, the goal of this contrastive process is to bring representations of patches of the same underlying clean structure closer together, which is unlike a triplet based learning process which simultaneously brings representations closer for similar patches while pushing away representations of dissimilar patches.

### 3.2 Joint normal estimation and filtering

Normal estimation and point cloud filtering are two interconnected tasks. Accurately predicted normals are central to reliable point cloud filtering and surface reconstruction as mentioned by [12], [16]. In a similar manner, predicting normals on less noisy patches provide more accurate results, as opposed to noisier patches where outliers affect the final prediction [15], [32]. This motivates our joint normal estimation and filtering approach where our regression network estimates patch normals along with point displacements to filter central patch points. Thereafter, the estimated normals are used to further refine the final filtered position. This approach helps exploit the interlinked relationship between

normal estimation and point cloud filtering and motivates our joint approach.

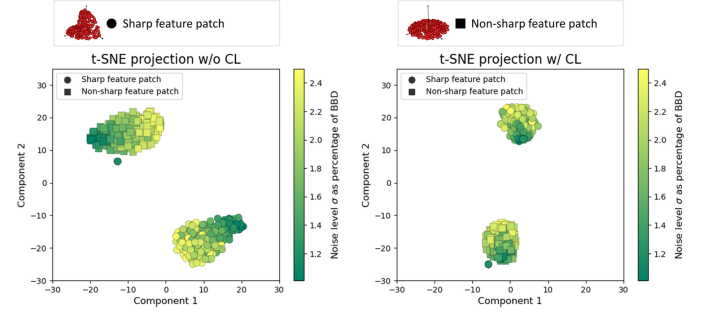


Fig. 2. The impact of using a feature encoder pretrained using contrastive learning compared to a feature encoder trained from scratch. We consider t-SNE projections of latent representations of a sharp feature patch and a non-sharp feature patch at different Gaussian noise levels  $\sigma$ , w.r.t. the bounding box diagonal (BBD) of the clean point cloud. Respective clean patches are illustrated at the top.

### 3.3 Link between contrastive learning and regression tasks

Feature encoders trained using contrastive learning are adept at generating similar representations of similar inputs and dissimilar representations of dissimilar inputs. Guided by the intuition that two noisy variants of the same underlying clean patch should generate similar latent representations, we develop a contrastive learning framework with noise corruption as an augmentation to train a feature encoder  $f(*)$  (see Fig. 1). This encoder is later used as part of a regression network (Fig. 3) to infer point normals and filtered displacements simultaneously. During training of the regression network, the pretrained feature encoder’s weights are kept frozen and only the regressor  $h(*)$  is trained. The pretrained feature encoder is robust to noise and generates representations that can be consumed more effectively by the regression network during training. Fig. 2 illustrates t-SNE projections of 250 noisy variants of a given sharp feature patch and 250 noisy variants of a non-sharp feature patch obtained from the Cube shape in our dataset. The corresponding clean patches are illustrated at the top of Fig. 2. Each noisy variant contains Gaussian noise of standard deviation  $\sigma$ , ranging from 1.0% to 2.5% of the clean point cloud’s bounding box diagonal. We observe that a feature encoder trained without contrastive learning generates latent representations that are less similar for differing noisy patch variants sharing a common underlying clean structure, for both the sharp feature and non-sharp feature patch. This is evident from Fig. 2 as low noise patch variants (dark green markers) have projections that are far apart from their respective higher noise counterparts (light green/yellow markers).

The feature encoder pretrained using contrastive learning, with only noise corruption as an augmentation, generates latent representations whose t-SNE projections are clustered more closely, indicating that representations are more similar even as noise increases. This is due to the contrastive pretraining that exploits noise corruption as an augmentation and ensures robustness to noise. As such,

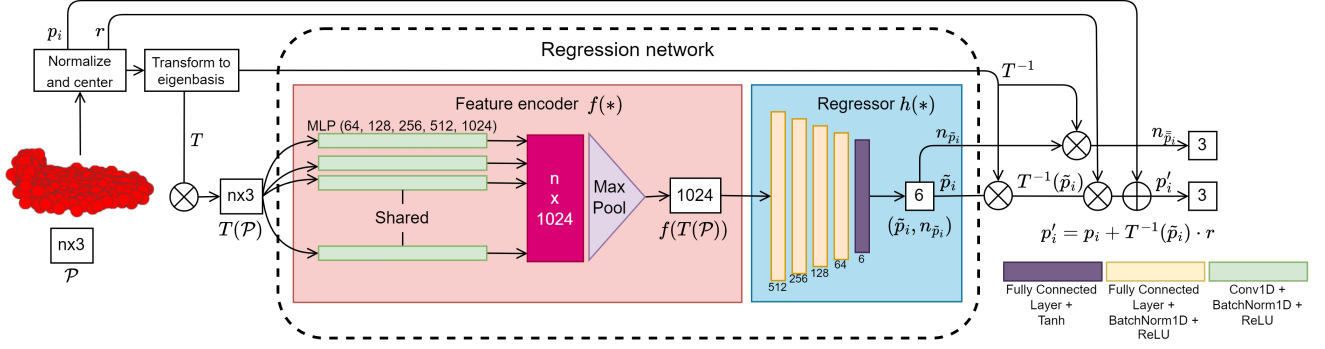


Fig. 3. Overview of the Regression network for outputting filtered points and normals. Feature encoder is taken from contrastive learning.

latent representations generated by the feature encoder are easier for the regression network to distinguish, even for high noise patches, as these representations are similar to that of the underlying clean counterparts. Thereby, the contrastive learning based pretraining facilitates our joint normal estimation and point cloud filtering method.

## 4 PROPOSED METHODOLOGY

### 4.1 Overview

We first introduce 3D patch-based contrastive learning to train a feature encoder capable of producing representations of point cloud patches (Fig. 1). This feature encoder consists of a PointNet-like architecture, with 5 Conv1D layers and a global max pool layer that generates a 1024 dimensional representation of input point cloud patches. These representations are projected to a 256 dimensional vector by a projection head consisting of 3 fully connected layers. Once the feature encoder has been trained, we use it to generate latent representations of input patches for regression tasks. Next, we train a regression network (Fig. 3) to predict patch normals and displacements simultaneously (i.e., normal and displacement of the central point of a patch). The regressor consists of the pretrained feature encoder and a MLP of 5 fully connected layers that output the desired normals and displacements. During the testing phase, these displacements are added to the initial point cloud, which produces a filtered point cloud that is refined and becomes the input for the next iteration of inference.

### 4.2 Contrastive pair construction

A clean point cloud consisting of  $n$  points is described by  $\mathbf{P}_s = \{p_i \mid p_i \in \mathbb{R}^3, i = 1, \dots, n\}$  where  $s = 1, \dots, M$  enumerates all training shapes. A noisy point cloud  $\hat{\mathbf{P}}_s$  can, thereafter, be characterized by the addition of noise onto the clean point cloud

$$\hat{\mathbf{P}}_s(\sigma) = \mathbf{P}_s + \mathbf{N}(0, \sigma^2), \quad (1)$$

where  $\mathbf{N}(0, \sigma^2)$  corresponds to additive Gaussian noise with a mean of 0 and standard deviation of  $\sigma$ . For our training set,  $\sigma$  takes on values of 0.25%, 0.5%, 1.0%, 1.5% and 2.5% of the bounding box diagonal length of  $\mathbf{P}_s$ . For a given shape, the set of 6 variant point clouds (1 clean and 5 noisy), is given by

$$\mathbf{A}_s = \{\mathbf{P}_s, \hat{\mathbf{P}}_s(\sigma) \mid \sigma = 0.25\%, \dots, 2.5\%\}, \quad (2)$$

Patches sampled from point clouds in  $\mathbf{A}_s$  are utilized in the contrastive learning process.

An input patch centered at  $p_i$ , the  $i^{th}$  point of a given point cloud  $\mathbf{P}_s(\sigma_1) \in \mathbf{A}_s$ , can be described by

$$\mathcal{P} = \{p_j \mid p_j \in \mathbf{P}_s(\sigma_1) \wedge \|p_j - p_i\|_2 < r_{\mathcal{P}}\}, \quad (3)$$

We create a contrastive pair  $(\mathcal{P}, \mathcal{Q})$  by randomly sampling another point cloud  $\mathbf{P}_s(\sigma_2) \in \mathbf{A}_s$  such that,

$$\mathcal{Q} = \{q_j \mid q_j \in \mathbf{P}_s(\sigma_2) \wedge \|q_j - p_i\|_2 < r_{\mathcal{Q}}\}, \quad (4)$$

Here,  $\mathbf{P}_s(\sigma_1)$  and  $\mathbf{P}_s(\sigma_2)$  correspond to two noisy variants of the same underlying clean point cloud and  $\sigma_1$  and  $\sigma_2$  are chosen randomly. The process of pairing  $\mathcal{P}$  with  $\mathcal{Q}$ , where both share the same underlying clean patch structure, as both are subsets of  $\mathbf{A}_s$  and are centered at  $p_i$ , is the first augmentation. We note that  $\mathcal{P}$  may have a different patch radius to  $\mathcal{Q}$  as the bounding box diagonal length of  $\mathbf{P}_s(\sigma_1)$  differs from  $\mathbf{P}_s(\sigma_2)$  unless  $\sigma_1 = \sigma_2$ . The patch radii  $r_{\mathcal{P}}$  and  $r_{\mathcal{Q}}$  are taken to be 5% of the respective bounding box diagonals. Patches are maintained at a fixed size to simplify the training procedure. Empirically, for each patch we sample 500 points. If the number of points is fewer, we increase it by copying random points in the patch and downsampling is applied otherwise. All patches are translated to the origin and normalized to  $[-1, 1]$ , i.e.,  $\mathcal{P} = (\mathcal{P} - p_i)/r_{\mathcal{P}}$  and  $\mathcal{Q} = (\mathcal{Q} - p_i)/r_{\mathcal{Q}}$ .

For a batch of size  $N$ , we have a total of  $2N$  augmented patches, i.e, we have input patches and their noise contrasted versions. The input patches are drawn from the training set  $\mathbf{T}$ , given by  $\mathbf{T} = \bigcup_{s=1}^M \mathbf{A}_s$ . All pairs  $(\mathcal{P}_k, \mathcal{Q}_k)$  and  $(\mathcal{Q}_k, \mathcal{P}_k)$ , where  $k = 1, \dots, N$ , form positive contrastive pairs as these patches are drawn from point clouds of the same shape, the set  $\mathbf{A}_s$ , and correspond to the same central point  $p_i$ . Any pairs  $(\mathcal{P}_k, \mathcal{Q}_l)$  or  $(\mathcal{Q}_l, \mathcal{P}_k)$  where  $k, l = 1, \dots, N \wedge k \neq l$  constitute negative pairs. For each augmented patch, there exists 1 positive pair and  $2(N - 1)$  negative pairs. We do not pair the augmented patch with itself, i.e., we do not consider  $(\mathcal{P}_k, \mathcal{P}_k)$  or  $(\mathcal{Q}_k, \mathcal{Q}_k)$ .

Positive contrastive pairs are put into the canonical basis by taking the inverse of the eigenvector matrix, obtained from the covariance matrix of  $\mathcal{P}_k$ , Eq. (5), and matrix multiplying both patches by it.

$$\mathcal{C}_{\mathcal{P}_k} = \frac{1}{|\mathcal{P}_k|} \sum_{p_j \in \mathcal{P}_k} (p_i - p_j)(p_i - p_j)^T, \quad (5)$$



Finally, we apply a second augmentation to  $\mathcal{Q}_k$  by randomly rotating it around either the  $x$ ,  $y$ , or  $z$  axis by an angle  $\theta \in \{0, \pi/12, \pi/6, \pi/4, \pi/3, \pi/2, 7\pi/12, 2\pi/3, 3\pi/4, 5\pi/6, \pi\}$ . This second augmentation allows the network to learn patch similarity based on patch structure despite rotations and reinforces the effect of contrastive learning.

### 4.3 Contrastive learning

Once contrastive pairs have been generated, we train our feature encoder  $f(*)$  in a contrastive learning manner. As patches within a positive pair correspond to the same ground-truth patch, albeit with different levels of additive noise and arbitrary rotations applied to them, they should produce similar representations. As such, we adopt the Normalized Temperature-scaled Cross Entropy (NT-XEnt) loss [48] to achieve this. The loss for a positive pair  $(\mathcal{P}_k, \mathcal{Q}_k)$  in the batch is given by

$$L_{\mathcal{P}_k, \mathcal{Q}_k} = -\log \frac{\exp(\mathbf{z}_{\mathcal{P}_k} \cdot \mathbf{z}_{\mathcal{Q}_k})/\tau}{\sum_{l=1}^{2N} \mathbb{1}_{[k \neq l]} \exp(\mathbf{z}_{\mathcal{P}_k} \cdot \mathbf{z}_{\mathcal{P}_l})/\tau}, \quad (6)$$

where  $l = 1, \dots, 2N$  enumerates over all projections. The function  $\mathbb{1}_{[k \neq l]} = 0$  if  $k = l$  and 1 otherwise. We empirically set  $\tau = 0.01$  based on results. In Eq. (6), the projection  $\mathbf{z}_* = g(f(*))$  with  $g(*)$  and  $f(*)$  parameterized by the projection head and feature encoder, respectively. Finally, the contrastive loss for the entire batch is,

$$L = \frac{1}{2N} \sum_{k=1}^N (L_{\mathcal{P}_k, \mathcal{Q}_k} + L_{\mathcal{Q}_k, \mathcal{P}_k}), \quad (7)$$

where  $k = 1, \dots, N$  enumerates over all positive pairs. Fig. 1 shows how a single positive pair within a batch is processed by the feature encoder and projection head. All positive and negative pairs within the batch see the same copy of the feature encoder and projection head and all pairs are processed simultaneously. The larger the batch size, the greater the number of negative pairs available for the contrastive loss calculation. Once the training of the contrastive learning network has been completed, we discard the projection head and use the feature encoder as the backbone for our regression network. Representations generated by the feature encoder are consumed by the regressor, comprising 5 MLPs, and outputs a 2-tuple of displacement and normal vectors. During the training of the regressor, the weights for the feature encoder are kept frozen. The regression process is visualized in Fig. 3.

### 4.4 Joint loss

We propose a novel, joint position and normal based loss for training our regression network. This joint loss allows our network to, when trained, perform both a position update of a noisy point while also estimating its corresponding normal. Therefore, our regressor loss function comprises two main components:  $L_{pos}$ , the position loss and  $L_{normal}$ , the normal loss.

**Approximating the clean surface.** The position loss term is inspired by the work of [21]. We consider all points within the ground-truth patch  $\mathcal{P}^*$  and seek to minimize the squared  $L_2$  norm instead of solely using a single fixed target.

$$L_{pos}^1(\tilde{p}_i, \mathcal{P}^*) = \min_{p_j \in \mathcal{P}^*} \|\tilde{p}_i - p_j\|_2^2, \quad (8)$$

where  $\tilde{p}_i$  is the filtered point from the regressor given a noisy patch  $\mathcal{P}$  centered at  $p_i$ , i.e., the  $i^{th}$  point in  $\mathcal{P}$ . The ground truth patch  $\mathcal{P}^* = \{p_j \mid p_j \in \mathbf{P}_s \wedge \|p_j - p_i\|_2 < r_{\mathcal{P}}\}$  is translated to the origin and normalized such that  $\mathcal{P}^* = (\mathcal{P} - p_i)/r_{\mathcal{P}}$ . This loss term allows the regressor to filter the noisy central point back to the clean patch surface. However, this does not ensure that the filtered point is centered within the ground truth patch which leads to unwanted clustering of filtered points.

**Ensuring regular distribution of points.** To avoid unwanted point clustering, we employ a regularization term which promotes the centering of filtered points within ground truth patches. By ensuring that filtered points lie close to the central points of their corresponding ground truth patches, we recover a regular distribution of points within the filtered point cloud.

$$L_{pos}^2(\tilde{p}_i, \mathcal{P}^*) = \max_{p_j \in \mathcal{P}^*} \|\tilde{p}_i - p_j\|_2^2, \quad (9)$$

Intuitively, if the filtered point lies away from the ground truth patch center, this leads to a larger penalization due to Eq. (9). For example, given a circular patch with radius  $r$ , the minimum value of Eq. (9) is  $r^2$  where the inferred point lies at the center of the ground truth patch and the furthest point is  $r$  away.  $L_{pos}^1$  and  $L_{pos}^2$  form the position based loss contribution to the final loss

$$L_{pos} = (1 - \beta)L_{pos}^1 + \beta L_{pos}^2, \quad (10)$$

where  $\beta$  is the parameter that controls the regularization term's contribution to the position loss. We empirically set it to 0.01, as we notice that a large contribution affects the convergence of the final loss function.

**Normal estimation.** We then develop a relationship between a regressed position and its normal. Intuitively, if the ground-truth patch point, which minimizes the squared  $L_2$  norm between positions, corresponds to the true position of the filtered point, then that point's normal should correspond to the true normal of the filtered point. If the ground-truth patch's central point which minimizes the squared  $L_2$  distance is given by

$$p_j^* = \operatorname{argmin}_{p_j \in \mathcal{P}^*} (\|\tilde{p}_i - p_j\|_2^2), \quad (11)$$

then the angle difference between the predicted normal and the ground-truth normal can be expressed in terms of cosine similarity between the two:

$$\cos(\theta) = n_{\tilde{p}_i} \cdot n_{p_j^*}, \quad (12)$$

where  $n_{\tilde{p}_i}$  and  $n_{p_j^*}$  correspond to the normals at  $\tilde{p}_i$  and  $p_j^*$ , respectively. This cosine term can now be used to construct our normal loss:

$$L_{normal} = 1 - [\delta \cos(\theta)^2 + (1 - \delta) \cos(\theta)^\gamma]. \quad (13)$$

The loss function in Eq. (13) is a periodic function which penalizes  $\theta$  values away from 0 and  $\pi$ . Therefore, it encourages the predicted normal to be as close to the ground-truth normal as possible. It also assumes that the predicted normal with an angle difference of  $\pi$  is equivalent to the ground-truth normal. The  $\delta$  term, which is empirically set to 0.3, serves to control the shape of loss function  $L_{normal}$ , wherein, angle differences close to  $\pi/2$  are heavily

penalized. This penalization decreases closer to 0 and  $\pi$ . Finally, we express our joint loss as

$$L_{final} = \alpha L_{pos} + (1 - \alpha) L_{normal}, \quad (14)$$

where  $\alpha$  controls the relative contributions of the position and normal losses to the final loss function. We empirically set  $\alpha$  to 0.9 as the emphasis for the regressor is denoising the point cloud iteratively. This in turn leads to better normal estimation results.

#### 4.5 Alternative joint loss

We also examine Eq. (17), a variant of the joint loss function, Eq. (14), which utilizes the point-to-point correspondences between ground truth points  $p_i^*$  and filtered points  $\tilde{p}_i$ , i.e., using fixed ground truth targets.

$$L_{pos}^{alt} = \|\tilde{p}_i - p_i^*\|_2^2, \quad (15)$$

$$L_{normal}^{alt} = 1 - [\delta \cos(\theta)^2 + (1 - \delta) \cos(\theta)^\gamma], \quad (16)$$

$$L_{final}^{alt} = \alpha L_{pos}^{alt} + (1 - \alpha) L_{normal}^{alt}, \quad (17)$$

where  $\cos(\theta) = n_{\tilde{p}_i} \cdot n_{p_i^*}$  with  $n_{\tilde{p}_i}$  and  $n_{p_i^*}$  being the predicted normal and ground truth normal, respectively. As we regress the filtered point directly back to the ground truth central point,  $L_{pos}^{alt}$  does not contain a repulsion term. The regressor trained with this loss function performs sub-optimally to that of the regressor trained using Eq. (14). As noted in [21], multiple clean points, within a given neighbourhood, may be perturbed in such a way as to result in the same noisy point. Therefore, the  $L_2$  norm minimization between a filtered point and ground truth point, Eq. (15), cannot successfully remove the noise component tangential to the surface and leads to a lower filtering performance. This motivates our use of Eq. (14) which regresses the filtered point back to the surface while ensuring it is centered as best possible within the ground truth patch. More details are given in Sec. 6.2.

#### 4.6 Inference

The regression network  $h(*)$ , trained based on the loss function defined by Eq. (14), outputs a 6D vector or 2-tuple  $(h_0, h_1)$  of 3D vectors. The first element corresponds to the displacement required to obtain the filtered point  $p'_i$ , and the second to the corresponding normal vector  $\tilde{n}_i$ . As these two vectors are in the space defined by the eigenvectors of the patch covariance matrix, they must first be transformed back to the original space. Subsequently, the filtered point is given by  $p'_i = p_i + T^{-1}(h(f(T(\mathcal{P})))_0) \cdot r_{\mathcal{P}}$  with the original noisy point  $p_i$ . The normal vector in the original space is  $n_{\tilde{p}_i} = T^{-1}(h(f(T(\mathcal{P})))_1)$  where  $f(*)$  is the feature encoder.

Given  $p'_i$ , we apply refinement during post-processing to obtain the final filtered point as suggested by [21], [24]. The first is a Taubin smoothing-like inflation step [21] and the second is the LRMA position update [24] to combat shrinking of the point cloud and avoid incorrect displacement of points along the surface, respectively. The ablation study on the post-processing refinement is discussed in Section 6.6. The new position after applying the inflation step is given by,

$$\bar{p}_i = p'_i - \frac{1}{|\mathcal{N}(p'_i)|} \sum_{p'_j \in \mathcal{N}(p'_i)} (p'_j - p_j), \quad (18)$$

where we take  $p'_j \in \mathcal{N}(p'_i)$  as the neighborhood of 100 filtered points in the vicinity of  $p'_i$  and  $p_j$  is the original point position before filtering. The LRMA position update yields our final filtered point. The position update is given by,

$$\bar{\bar{p}}_i = \bar{p}_i + \frac{1}{3|\mathcal{N}(\bar{p}_i)|} \sum_{\bar{p}_j \in \mathcal{N}(\bar{p}_i)} (\bar{p}_j - \bar{p}_i)(n_{\bar{p}_j}^T n_{\bar{p}_i} + n_{\bar{p}_i}^T n_{\bar{p}_j}), \quad (19)$$

with  $\mathcal{N}(\bar{p}_i)$  being the neighborhood of 20 points in the vicinity of  $\bar{p}_i$ .

## 5 EXPERIMENTAL RESULTS

### 5.1 Dataset

Our training set consists of 22 synthetic point clouds (Fig. 4): 11 CAD shapes and 11 non-CAD shapes. The validation set consists of 3 shapes, 1 CAD and 2 non-CAD shapes, while the test set consists of 23 shapes: 14 CAD and 9 non-CAD. Each shape is a point cloud of 100K points, which have been randomly sampled from their original surfaces. For training, we create 5 additional noisy variants of each training shape by adding Gaussian noise with standard deviations of 0.25%, 0.5%, 1.0%, 1.5% and 2.5% of the clean point cloud's bounding box diagonal. These 6 variants (clean and 5 noise levels) for each shape give a total of 132 point clouds for training purposes. For validation, we consider 2 noisy variants of the initial 3 clean validation shapes, with 0.5% and 1.0% noise, resulting in a total of 6 validation point clouds. In the testing phase, we examine the robustness of our model at unseen noise levels by utilizing 0.6%, 0.8%, 1.1%, 1.5% and 2.0% Gaussian noise for each test shape, yielding 115 test point clouds.

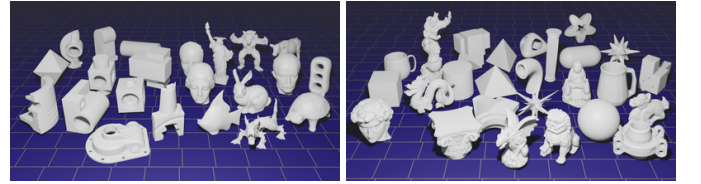


Fig. 4. Left: Meshes of synthetic point clouds used for training. Right: Meshes of synthetic point clouds used for validation (3 on the left) and for testing (23 on the right).

**Sharp features.** To evaluate performance at sharp features, we classify points within our synthetic dataset as feature and non-feature points. Please refer to the supplementary document for more details.

### 5.2 Implementation

The contrastive learning and regression networks are both trained on NVIDIA A100 GPUs using PyTorch 1.7.1 with CUDA 11.0. The contrastive learning network is trained for 150 epochs, with the Adam optimizer and a learning rate of  $3 \times 10^{-4}$ . The regression network is trained for 30 epochs, utilizing the SGD optimizer with a learning rate of  $1 \times 10^{-2}$ .

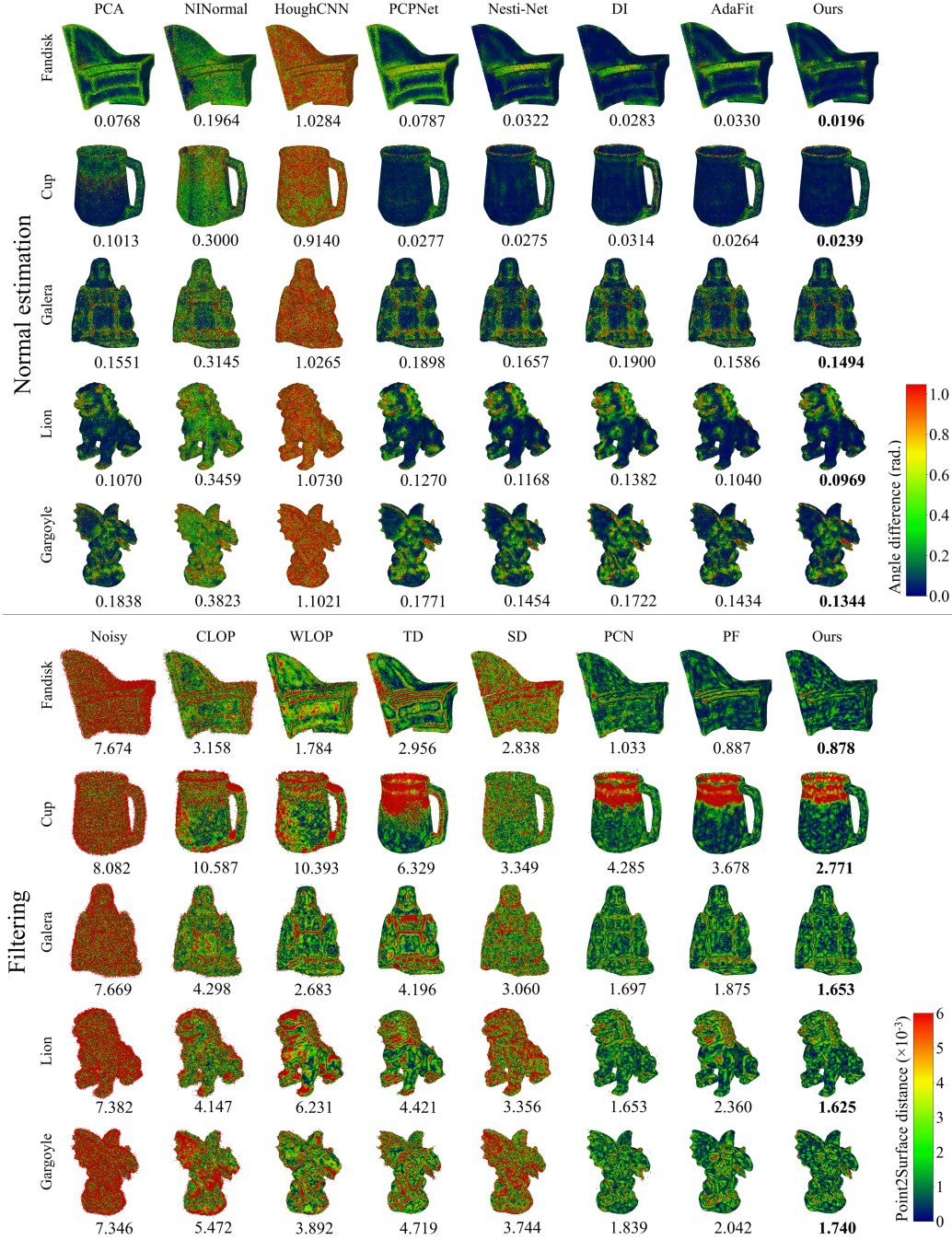


Fig. 5. Normal estimation (top half) and filtering (bottom half) results on shapes with 0.8% Gaussian noise w.r.t. the bounding box diagonal. For normal estimation, the respective mean squared angular error (MSAE) is given below each shape and the heat map corresponds to the angle difference at each point. For filtering, the Chamfer distance ( $\times 10^{-5}$ ) is given below each shape and the heat map corresponds to the scale normalized Point2Surface distance ( $\times 10^{-3}$ ).

### 5.3 Comparisons

We compare our method with state-of-the-art normal estimation and point cloud filtering methods. For normal estimation, we consider conventional PCA [5], HoughCNN [29], NINormal [33], PCPNet [32], Nesti-Net [13], Deep Iterative (DI) [15] and AdaFit [14] on our test set including synthetic and scanned point clouds. We do not compare with DeepFit [34] as AdaFit is based on DeepFit and achieves better results. PCA requires the manual selection of neighborhood sizes for plane-fitting. For synthetic

shapes, we utilize three different neighborhood sizes, i.e., 60 points for 0.6% Gaussian noise, 150 points for 0.8% Gaussian noise and 200 points for 1.1%, 1.5% and 2.0% Gaussian noise. For scanned surfaces a neighborhood of 100 points is used. Our metric for comparison is the Mean Squared Angular Error (MSAE) where we calculate angle differences between ground truth normals  $n_i$  and predicted normals  $\tilde{n}_i$  and take the mean of their squares.

For point cloud filtering, we compare with conventional methods CLOP [41] and WLOP [20] and deep learning methods TotalDenoising (TD) [23], ScoreDenoise (SD) [45],

TABLE 1

Normal estimation results on synthetic and scanned data given by the MSAE average. The MSAE for NINormal on scanned data is not given as it was not possible to test this method on input point clouds with over 100K points. The Gaussian noise standard deviation  $\sigma$  is with respect to the bounding box diagonal of the clean point cloud. Top results in bold and second best results are underlined.

	PCA	NINormal	HoughCNN	PCPNet	Nesti-Net	DI	AdaFit	Ours
Syn. ( $\sigma=0.6\%$ )	0.1637	0.2113	0.7418	0.1184	<u>0.1029</u>	0.1037	<b>0.0958</b>	0.1038
Syn. ( $\sigma=0.8\%$ )	0.1514	0.3277	1.0427	0.1414	<u>0.1197</u>	0.1246	0.1143	<b>0.1126</b>
Syn. ( $\sigma=1.1\%$ )	0.1954	0.4813	1.1595	0.1762	<u>0.1467</u>	0.1539	<u>0.1472</u>	<b>0.1268</b>
Syn. ( $\sigma=1.5\%$ )	0.2664	0.6160	1.1879	0.2174	<u>0.1827</u>	0.1932	0.1914	<b>0.1502</b>
Syn. ( $\sigma=2.0\%$ )	0.3632	0.7175	1.1853	0.2646	<u>0.2319</u>	0.2405	0.2440	<b>0.1951</b>
Syn. average	0.2280	0.4708	1.0634	0.1836	<u>0.1568</u>	0.1632	0.1585	<b>0.1377</b>
Sharp feat. ave.	0.4216	0.5872	1.0003	0.3947	<u>0.3504</u>	0.3913	<u>0.3445</u>	<b>0.3230</b>
Scanned average	0.1010	—	0.4960	0.0550	0.0430	<u>0.0370</u>	<b>0.0250</b>	0.0390
Overall average	0.2237	0.4550	1.0443	0.1793	<u>0.1530</u>	<u>0.1590</u>	0.1540	<b>0.1344</b>

TABLE 2

Average Chamfer and Point2Surface distance results on synthetic and scanned data. The Gaussian noise standard deviation  $\sigma$  is with respect to the bounding box diagonal of the clean point cloud. Top results in bold and second best results are underlined.

		Noisy	CLOP	WLOP	TD	SD	PCN	PF	Ours
Chamfer dist. ( $\times 10^{-5}$ )	Syn. ( $\sigma=0.6\%$ )	4.762	2.366	3.374	4.638	1.867	1.467	<u>1.444</u>	<b>1.241</b>
	Syn. ( $\sigma=0.8\%$ )	7.517	4.163	3.921	4.038	3.127	1.752	<u>1.749</u>	<b>1.545</b>
	Syn. ( $\sigma=1.1\%$ )	12.746	8.311	6.533	3.508	6.903	2.572	<u>2.355</u>	<b>1.987</b>
	Syn. ( $\sigma=1.5\%$ )	21.637	16.13	14.24	7.472	14.301	5.787	<u>3.544</u>	<b>3.244</b>
	Syn. ( $\sigma=2.0\%$ )	35.906	29.177	27.318	22.838	27.943	17.658	<u>5.307</u>	8.055
	Syn. average	16.514	12.029	11.077	8.499	10.828	5.847	<u>2.880</u>	<u>3.214</u>
	Scanned average	2.764	3.317	5.618	3.835	3.443	0.542	<u>0.482</u>	<b>0.472</b>
	Overall average	16.052	11.736	10.894	8.342	10.58	5.669	<u>2.799</u>	<u>3.122</u>
Point2Surf. dist. ( $\times 10^{-3}$ )	Syn. ( $\sigma=0.6\%$ )	4.658	2.392	2.423	3.195	2.262	1.380	<u>1.141</u>	<b>1.067</b>
	Syn. ( $\sigma=0.8\%$ )	6.143	3.730	3.052	3.078	3.520	1.592	<u>1.386</u>	<b>1.324</b>
	Syn. ( $\sigma=1.1\%$ )	8.306	5.976	4.810	3.063	6.009	2.108	<u>1.810</u>	<b>1.807</b>
	Syn. ( $\sigma=1.5\%$ )	11.073	8.959	8.418	5.471	9.259	3.100	<u>2.472</u>	<u>2.937</u>
	Syn. ( $\sigma=2.0\%$ )	14.432	12.685	12.460	11.282	13.315	5.124	<u>3.440</u>	<u>5.020</u>
	Syn. average	8.922	6.748	6.233	5.218	6.873	2.661	<u>2.050</u>	<u>2.431</u>
	Sharp feat. ave.	9.082	6.867	6.324	6.409	6.893	3.366	<u>2.339</u>	<u>2.887</u>
	Scanned average	3.795	3.410	4.037	2.990	3.957	1.000	<u>0.722</u>	<b>0.705</b>
	Overall average	8.750	6.636	6.159	5.143	6.775	2.605	<u>2.005</u>	<u>2.373</u>

PointCleanNet [21] and Pointfilter [22] on two metrics, the Chamfer distance (CD) and the Point2Surface distance (P2S) [57]. The Chamfer distance between a ground truth point cloud  $\mathbf{P}_s$  and a filtered point cloud  $\tilde{\mathbf{P}}$  is defined by Eq. (20). The first term provides a measure of the distance from filtered points to the ground truth surface while the second provides a measure of the relative even distribution of filtered points w.r.t. the clean point cloud. P2S measures the average distance between filtered points and the reconstructed ground truth mesh. All learning based methods are retrained on our dataset.

$$C(\mathbf{P}_s, \tilde{\mathbf{P}}) = \frac{1}{|\tilde{\mathbf{P}}|} \sum_{p_i \in \tilde{\mathbf{P}}} \min_{p_j \in \mathbf{P}_s} \|p_i - p_j\|_2^2 + \frac{1}{|\mathbf{P}_s|} \sum_{p_j \in \mathbf{P}_s} \min_{p_i \in \tilde{\mathbf{P}}} \|p_j - p_i\|_2^2 \quad (20)$$

**Evaluation at sharp features.** We take the MSAE for sharp feature points as a measure of normal estimation accuracy and the P2S distance for these points as a measure of filtering accuracy. Tables 1, 2, 5 and 6 give the MSAE and P2S averages for each method on our dataset.

#### 5.4 Performance on synthetic data

Tables 1 and 3 and the top half of Fig. 5 demonstrate normal estimation results on shapes with Gaussian noise. For the normal estimation task, our method outperforms all other

methods including ones which employ larger networks, such as Nesti-Net, and recovers accurate normals in the presence of noise. AdaFit offers competitive results at lower noise levels but performs sub-optimally at higher noise. Another key attribute is its ability to accurately predict normals at sharp features. Both these attributes of the network can be seen by its performance on the Fandisk shape. Overall, it has the lowest MSAE at sharp feature points and shows higher robustness to noise. Contrastive learning facilitates this as patches from anisotropic surfaces are trained to produce distinct feature representations which can be better distinguished by the regressor. Additionally, as positive contrastive pairs share the same underlying clean surface, representations of a given patch at differing noise levels remain similar. This allows the regressor to reliably estimate normals as noise increases.

Table 2 and 3 and the bottom half of Fig. 5 demonstrate filtering results on shapes with Gaussian noise. A core attribute of our method is that it generalises well between both CAD-like and non-CAD like shapes. It is able to recover the sharp feature information of CAD shapes such as Fandisk while also achieving the highest accuracy on non-CAD shapes such as Galera and Gargoyle. Furthermore, on complex shapes such as Cup, our method outperforms others at filtering noisy points along closely neighboring surfaces. Pointfilter specializes in the point cloud filtering task and it only outperforms our method on both metrics



TABLE 3

Individual MSAE and Chamfer distance values for shapes presented in Fig. 5, at Gaussian noise levels of standard deviation  $\sigma$ , with respect to the bounding box diagonal of the clean point cloud.

Shape	$\sigma$	Normal estimation - MSAE					Filtering - Chamfer distance ( $\times 10^{-5}$ )			
		PCPNet	Nesti-Net	DI	AdaFit	Ours	Noisy	PCN	PF	Ours
Fandisk	0.6%	0.0615	0.0205	<b>0.0148</b>	0.0197	<b>0.0159</b>	4.735	0.896	<b>0.748</b>	<b>0.747</b>
	0.8%	0.0787	0.0322	0.0283	0.0329	<b>0.0196</b>	7.674	1.033	<b>0.887</b>	<b>0.878</b>
	1.1%	0.1005	<u>0.0534</u>	<u>0.0542</u>	0.0563	<b>0.0266</b>	13.232	1.546	<u>1.246</u>	<b>1.244</b>
	1.5%	0.1250	<u>0.0708</u>	0.0828	0.0948	<b>0.0442</b>	22.610	3.520	<b>1.678</b>	<b>2.285</b>
	2.0%	0.1498	<u>0.0945</u>	0.1152	0.1366	<b>0.0716</b>	37.450	13.952	<b>2.916</b>	<b>6.613</b>
Cup	0.6%	0.0194	0.0218	0.0262	<b>0.0198</b>	<b>0.0213</b>	5.276	3.488	2.384	<b>2.117</b>
	0.8%	0.0277	0.0275	0.0314	<u>0.0264</u>	<b>0.0239</b>	8.082	4.285	<u>3.678</u>	<b>2.771</b>
	1.1%	0.0451	0.0399	0.0389	0.0446	<b>0.0287</b>	12.961	5.609	<u>5.228</u>	<b>2.957</b>
	1.5%	0.0773	0.0645	<u>0.0518</u>	0.0749	<b>0.0390</b>	21.329	10.267	<u>8.462</u>	<b>2.995</b>
	2.0%	0.1153	0.1108	<u>0.0755</u>	0.1027	<b>0.0640</b>	34.351	22.665	<u>10.917</u>	<b>5.057</b>
Galera	0.6%	0.1504	0.1357	0.1469	<b>0.1256</b>	0.1324	4.924	<u>1.422</u>	1.516	<b>1.334</b>
	0.8%	0.1898	0.1657	0.1899	0.1586	<b>0.1494</b>	7.669	<u>1.697</u>	1.875	<b>1.653</b>
	1.1%	0.2416	<u>0.2085</u>	0.2472	<u>0.2091</u>	<b>0.1747</b>	12.922	<u>2.430</u>	2.478	<b>2.128</b>
	1.5%	0.2904	<u>0.2609</u>	0.2947	0.2670	<b>0.2166</b>	21.713	<u>5.930</u>	3.493	<b>3.324</b>
	2.0%	0.3403	<u>0.3136</u>	0.3441	0.3219	<b>0.2775</b>	35.215	19.101	<b>6.112</b>	<b>6.879</b>
Lion	0.6%	0.0957	0.0896	0.0989	<b>0.0741</b>	<b>0.0798</b>	4.691	<u>1.309</u>	1.768	<b>1.219</b>
	0.8%	0.1270	0.1168	0.1382	<u>0.1040</u>	<b>0.0969</b>	7.382	<u>1.653</u>	2.360	<b>1.625</b>
	1.1%	0.1707	0.1570	0.1786	<u>0.1481</u>	<b>0.1262</b>	12.489	<u>2.567</u>	3.160	<b>2.273</b>
	1.5%	0.2205	0.2053	0.2185	<u>0.2022</u>	<b>0.1684</b>	21.135	<u>6.105</u>	4.130	<b>3.606</b>
	2.0%	0.2755	0.2630	0.2718	<u>0.2595</u>	<b>0.2279</b>	34.894	19.627	<b>6.181</b>	<b>8.796</b>
Gargoyle	0.6%	0.1305	0.1113	0.1250	<b>0.1038</b>	0.1134	4.710	<u>1.518</u>	1.661	<b>1.345</b>
	0.8%	0.1771	<u>0.1454</u>	0.1722	<u>0.1434</u>	<b>0.1344</b>	7.346	<u>1.839</u>	2.042	<b>1.74</b>
	1.1%	0.2445	<u>0.2037</u>	0.2340	<u>0.2159</u>	<b>0.1753</b>	12.16	<u>2.968</u>	3.034	<b>2.441</b>
	1.5%	0.3077	<u>0.2672</u>	0.2953	0.2865	<b>0.2253</b>	19.903	<u>6.594</u>	5.179	<b>3.902</b>
	2.0%	0.3725	<u>0.3383</u>	0.3616	0.3523	<b>0.3019</b>	32.178	19.408	<b>8.000</b>	<b>8.597</b>

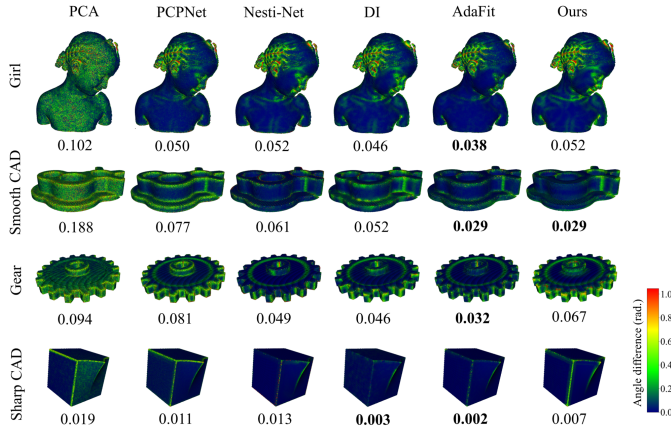


Fig. 6. Normal estimation results on scanned shapes.

at the highest noise level. Although, at higher noise, Point-filter has slightly lower Point2Surface distances, the higher Chamfer distance indicates a relatively uneven distribution of points along the filtered surface. This is due to the fact that there are multiple displacements along which a point may return to the underlying clean surface. Therefore, even if the point is returned to the underlying surface, it may be incorrectly positioned along the surface. Our method reduces this longitudinal jitter by using estimated normals in conjunction with the LRMA position update scheme of [24]. This takes regressed point positions, the output from our network, as input and updates them based on the neighboring points' positions and normal information from the previous iteration. Thereby, we are able to optimize both the position and normal estimation predictions to generate a filtered point cloud which more accurately represents the

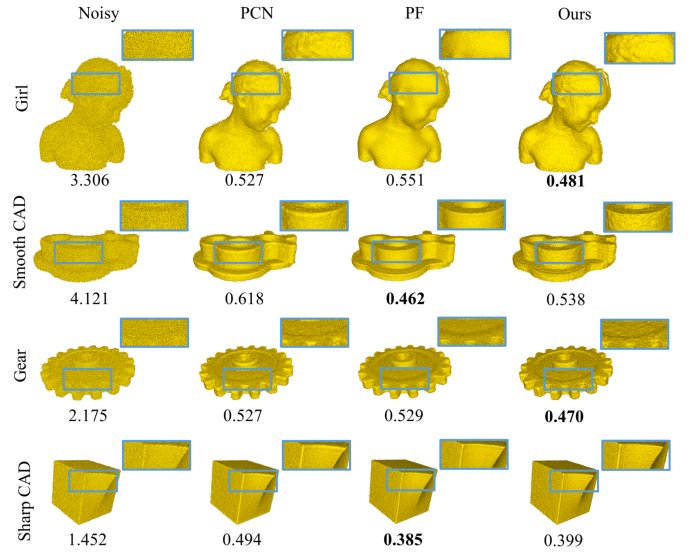


Fig. 7. Visual filtering results on scanned shapes.

original clean point cloud. Additional visual results are given in the supplementary document.

### 5.5 Performance on scanned data

Next, we look at the performance of each method on the scanned dataset which comprises 3 CAD-like scans and 1 non-CAD like scan where the ground-truth normal and position information are known. These scans are obtained using the virtual scanner introduced by Yu et al [44]. Additionally, we present comparisons on the Kinect v1 and Kinect v2 datasets, introduced by Wang, Liu and Tong [58], comprising 71 and 72 scans respectively, obtained using



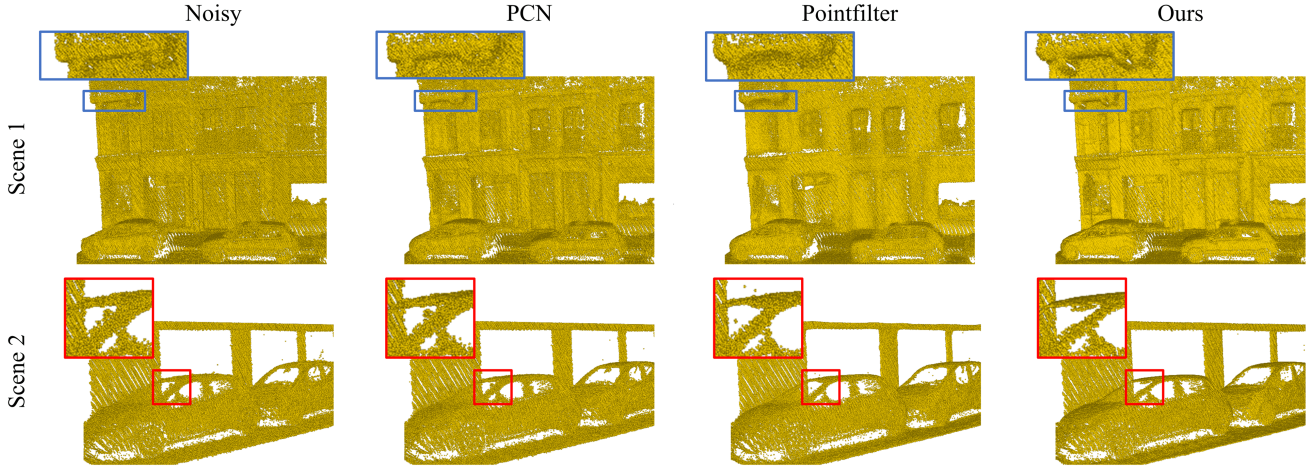


Fig. 8. Visual filtering results for the two scenes extracted from the Rue-Madame data. Our method recovers sharp features and fine details of vehicles and building facades while other methods perform sub-optimally.

Kinect v1 and v2 sensors. As these scans have no ground truth normal information, we only present filtering results. We also test the learning based filtering methods on two scans extracted from the Paris-Rue-Madame database [59]. They are scans taken from Rue Madame, a street in the 6<sup>th</sup> district of Paris. These scans capture many intricate details like the street’s building facades and parked vehicles and also provide a good example of real-world noise that is encountered when obtaining such scans. Finally, we provide results on two scans extracted from the Kitti-360 dataset [60] in the supplementary document. They capture scenes in several suburbs of Karlsruhe, Germany, using Velodyne HDL-64E sensors. As the Rue Madame and Kitti-360 scans have no ground truth, only visual results are presented. Tables 1 and 2 illustrate quantitative normal estimation and filtering results on the scanned data. AdaFit outperforms other methods in estimating normals on scanned data, but is less robust to high noise. Our method, on average, performs better than other methods and remains competitive at estimating normals near sharp features, as shown in Fig. 6.

TABLE 4

Filtering results on the Kinect v1 and Kinect v2 datasets. Average Chamfer distance ( $\times 10^{-5}$ ) and Point2Surface distance ( $\times 10^{-3}$ ) values are presented for each method.

		Noisy	PCN	PF	Ours
Kinect v1	CD	14.489	13.469	12.623	<b>12.083</b>
	P2S	6.272	5.763	5.029	<b>4.947</b>
Kinect v2	CD	22.633	21.985	20.174	<b>18.785</b>
	P2S	7.505	7.269	6.265	<b>5.889</b>

Among filtering methods, our method shows an advantage in recovering sharp feature information as well as fine details, such as the braids in the girl’s hair in Fig. 7. Our method also performs optimally on the Kinect v1 and Kinect v2 datasets, as demonstrated by Table 4. We outperform other methods on both the Chamfer distance and Point2Surface metrics, indicating its ability to approximate the surface and generate a regular distribution of points. Visual results on the Kinect data are presented in the supplementary document. Furthermore, Fig. 8 demonstrates our method’s ability to filter complex scenes. Results

on Scene 1 and Scene 2 demonstrate our method’s ability to maintain sharp features, in the presence of real-world noisy artifacts, while other methods such as PointCleanNet (PCN) and Pointfilter (PF) [21], [22] tend to smear feature information. For example, in Scene 1, the outlines of doors and windows are filtered accurately by our method while smoothing planar surfaces (surfaces of doors, walls). Fine details such as the headlights of vehicles and the outlines of tire rims are also recovered. Conversely, Pointfilter smears such details while PointCleanNet only partially filters them and is not successful in smoothing planar surfaces. In Scene 2, we demonstrate an ability to reliably filter details of vehicles including side-mirrors and windows while other methods perform sub-optimally.

Based on these results, we see that our method is very competitive in filtering while holding a clear advantage in normal estimation. On average (Tables 1 and 2), our method outperforms other methods in both tasks.

## 5.6 Performance on shapes with varying density and different noise patterns

The synthetic data results presented in Tables 1 and 2 correspond to point clouds generated by uniformly sampling their original meshes, with Gaussian random noise subsequently added to them. We also evaluate normal estimation and filtering methods on uniformly sampled point clouds with an impulsive noise pattern and on point clouds sampled with varying density. To generate point clouds with an impulsive noise pattern, we apply Gaussian noise of  $\sigma = 1.5\%$ , with respect to the bounding box diagonal, to 30% of points in each clean point cloud within the test set. To obtain varying density point clouds, two different sampling regimes, gradient and striped, are applied to the original test meshes. Thereafter, Gaussian noise of  $\sigma = 0.8\%$  is applied to these point clouds. Further details on varying density point clouds can be found in the supplementary document. When evaluating the performance of methods on test point clouds with varying density and different noise patterns, we use models trained on our synthetic dataset comprising of point clouds with Gaussian noise. Table 5 and Table 6 demonstrate our method’s ability to generalize well

TABLE 5

MSAE results on point clouds with varying density (VD) and Impulsive Noise (IN). The Gaussian noise standard deviation  $\sigma$  is with respect to the bounding box diagonal of the clean point cloud.

	PCA	PCPNet	Nesti-Net	DI	AdaFit	Ours
VD ( $\sigma=0.8\%$ )	0.143	0.102	0.098	0.089	0.083	<b>0.075</b>
VD sharp feat. ave.	0.298	0.294	0.273	0.278	0.247	<b>0.242</b>
IN ( $\sigma=1.5\%$ )	0.118	0.125	0.106	<b>0.077</b>	0.104	0.111
IN sharp feat. ave.	0.334	0.329	<b>0.275</b>	<b>0.208</b>	0.304	0.309

TABLE 6

Average Chamfer and Point2Surface distance results on point clouds with varying density (VD) and Impulsive Noise (IN). The Gaussian noise standard deviation  $\sigma$  is with respect to the bounding box diagonal of the clean point cloud.

		Noisy	PCN	PF	Ours
CD ( $\times 10^{-5}$ )	VD ( $\sigma=0.8\%$ )	7.560	1.730	1.680	<b>1.530</b>
	IN ( $\sigma=1.5\%$ )	6.366	3.279	1.001	<b>0.563</b>
P2S ( $\times 10^{-3}$ )	VD ( $\sigma=0.8\%$ )	6.272	1.844	<b>1.564</b>	1.637
	VD sharp feat. ave.	6.380	2.598	<b>2.336</b>	2.338
	IN ( $\sigma=1.5\%$ )	3.411	1.862	1.451	<b>0.897</b>
	IN sharp feat. ave.	3.568	2.993	<b>2.597</b>	<b>1.383</b>

on point clouds with varying density for normal estimation while performing competitively in filtering. For point clouds with an impulse noise pattern, we perform optimally at the filtering task.

## 5.7 Comparative runtimes

Finally, we look at the comparative runtimes for the different methods among the best performing normal estimation and filtering methods, respectively. The runtimes for normal estimation are 20.9, 2.0 minutes/100K points for Nesti-Net and AdaFit while for filtering, PointCleanNet and Point-filter take 27.2, 1.8 minutes/100K points, respectively. Our method, by comparison, takes 4.9 minutes/100K points for the combined normal estimation and filtering tasks.

## 6 ABLATION STUDY

We empirically found optimal values for the three main hyper-parameters, which are  $\alpha = 0.9$ ,  $\gamma = 12$  and the contrastive batch size of 512. Please refer to Sec. 6.4 for details.

TABLE 7

MSAE and Chamfer distance results on the validation set when only one task, normal estimation ( $\gamma=12$ , CBS=512) or filtering (CBS=512), is performed as opposed to when they are jointly performed ( $\alpha = 0.9$ ,  $\gamma=12$ , CBS=512).

	Normal estimation	Filtering	Both
MSAE	<b>0.052</b>	—	<b>0.034</b>
CD ( $\times 10^{-5}$ )	—	<b>1.609</b>	<b>1.431</b>

### 6.1 Merely normal estimation or filtering

We test the variants with merely normal estimation or point cloud filtering. Table 7 shows the results of merely normal estimation, merely point cloud filtering, and the proposed joint approach on the validation set. We see that our joint approach achieves best quantitative results in terms of both MSAE and Chamfer distance, confirming the effectiveness of our joint method.

### 6.2 Alternative joint loss

Next, we look at the performance of the alternative loss function, Eq. (17), which considers the point-to-point correspondences between ground truth points  $p_i^*$  and filtered points  $\tilde{p}_i$ , i.e., using fixed ground truth targets. We see poorer results for this alternative loss function on the validation set, i.e., MSAE: 0.038 vs 0.034, Chamfer distance:  $1.58 \times 10^{-5}$  vs  $1.43 \times 10^{-5}$ . The latter results correspond to the original joint loss in Eq. (14). As expected, the original joint loss produces filtered point clouds closer to the corresponding ground truth point clouds, with smaller Chamfer distances between them while the  $L_2$  norm minimization of the alternative joint loss function, Eq. (17), is less successful at the filtering task.

### 6.3 Effect of 3D patch based contrastive learning

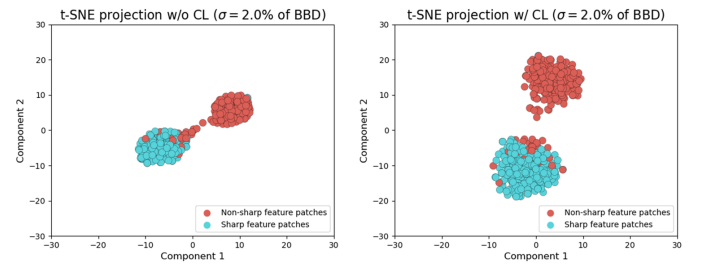


Fig. 9. T-SNE projections of latent representations of sharp feature patches and non-sharp feature patches. On the left, projections are for latent representations generated by a feature encoder without contrastive learning-based (CL) pretraining and on the right, projections generated by a feature encoder pretrained using contrastive learning. The pretrained feature encoder is able to better distinguish between sharp feature and non-sharp feature patches.

Contrastive pretraining can be used to better distinguish between latent representations of sharp feature patches from non-sharp feature patches. We can see this in Fig. 9 where the t-SNE projections of latent representations of sharp feature patches are better separated from those of non-sharp feature patches when a feature encoder with contrastive pretraining is used. Here, we looked at 250 sharp feature and 250 non-sharp feature patches, extracted from the Cube shape, and the projections of their respective representations.

We evaluate a variant without the contrastive learning stage on the validation set to gauge the effect of contrastive learning. This is done by training a regressor with a feature encoder whose weights are randomly initialized and updated along with the weights of the regressor. As given by Table 7, the model with contrastive learning achieves a MSAE of 0.034 and a Chamfer distance of  $1.431 \times 10^{-5}$ . The network trained without contrastive learning has a MSAE of 0.037 and a Chamfer distance of  $1.445 \times 10^{-5}$ , indicating the performance gain from contrastive learning.

Next, we look at the effect of utilizing a larger dataset for pretraining the feature encoder while training the regressor on a limited amount of labeled data. To achieve this, we partition our training set into 2 parts. Dataset Partition 1 (DSP1) consists of 18 shapes and is used to train the feature encoder. Dataset Partition 2 (DSP2) consists of the remaining 4 shapes and is only used to train the regressor. Again,

TABLE 8

Performance of regression network (1) on the test set when trained on the limited DSP2 data with its feature encoder pretrained on DSP1 using contrastive learning (CL). We compare this to regression network (2) trained on DSP2, without pretraining its feature encoder.

Trained model	MSAE	CD ( $\times 10^{-5}$ )
(1) Regr. network trained on DSP2 w/ CL	<b>0.1660</b>	<b>3.961</b>
(2) Regr. network trained on DSP2 wo/ CL	0.1713	4.057

we train a variant from scratch, without the contrastive pretraining, given by (2) in Table 8. We evaluate the normal estimation and filtering performance on the test set. We see that, with limited training data for the regression task, the positive impact of the contrastive pretraining becomes more apparent.

Finally, we investigate the impact of pretraining the feature encoder on different noise patterns. Therefore, in addition to the Gaussian noise pattern, we augment DSP1 with simulated Lidar noise. In order to achieve this, we utilize the Blensor toolkit [61] to create noisy variants of the initial 18 shapes of DSP1. The regressor is trained on the limited DSP2 point clouds containing only Gaussian noise, i.e., simulated Lidar noise is not seen by the regressor during training. Thereafter, we look at filtering results on simulated Lidar point clouds with the noise level set to 1.5%. We compare these results with network (1), with contrastive pretraining on DSP1 with only Gaussian noise and (2), trained from scratch on DSP2. The network trained without contrastive learning, (2) has a Chamfer distance of  $3.796 \times 10^{-5}$ . Network (1), pretrained on DSP1 with noisy point clouds containing only Gaussian noise, fairs marginally better with a Chamfer distance of  $3.787 \times 10^{-5}$ . However, for network (3) pretrained on DSP1 with both Gaussian and simulated Lidar noise patterns, the average Chamfer distance is  $3.731 \times 10^{-5}$ . This indicates the power of pretraining the feature encoder on additional noise patterns.

#### 6.4 Ablation study on hyperparameters

The three main hyper-parameters which control the predictive power of the network are the following:

- $\alpha$ , the weight controlling the relative contributions of the position and normal losses to the overall final loss, Eq (14), during training of the regression network,
- $\gamma$ , the exponent of the cosine similarity term in Eq. (13) of the main paper, which penalizes the angle difference between predicted and ground-truth normals,
- and the contrastive batch size (CBS) used during training of the feature encoder.

We use MSAE as our main evaluation metric for the ablation study as it typically provides a larger degree of agreement with qualitative (visual) comparisons [62]. Furthermore, we observe a noticeably large deviation in MSAE due to the choice of hyperparameters, as opposed to the deviation in Chamfer distance values.

In order to find the optimal values of  $\gamma$  and the contrastive batch size, we perform a grid search over both. We search over  $\gamma$  values of 8, 10 and 12 and CBS values of 32, 128 and 512. Each respective network, for a given pair of values, is evaluated on the validation set and corresponding

TABLE 9

MSAE results on the validation set for different contrastive batch sizes and different values of  $\gamma$ . Here,  $\alpha$  is set to 0.9.

	$\gamma$	CBS		
		32	128	512
MSAE	8	0.038	0.038	0.037
	10	0.036	0.036	0.036
	12	<u>0.035</u>	0.039	<b>0.034</b>

TABLE 10

MSAE results on the validation set for different values of  $\alpha$  with  $\gamma=12$  and a contrastive batch size of 512.

$\alpha$	0.8	0.85	0.9	0.92	0.95
MSAE	0.038	0.037	<b>0.034</b>	<u>0.035</u>	0.037

MSAE values are given in Table 9. Here, it was necessary to set a value of  $\alpha$  to perform the grid search. We set  $\alpha = 0.9$  based on empirical results. The pair (12, 512) of  $\gamma$ , CBS values optimizes MSAE results. Once the best  $\gamma$  and CBS values are determined, we use them to gauge the effect of varying  $\alpha$  and justify our selection of  $\alpha = 0.9$  as demonstrated in Table 10. The triplet of values  $\alpha = 0.9$ ,  $\gamma = 12$  and CBS = 512 indeed minimizes the MSAE.

TABLE 11

MSAE results on the validation set for different  $\beta$  and  $\delta$ . Here,  $\alpha = 0.9$ ,  $\gamma = 12$  and CBS = 512.

	$\beta$	$\delta$		
		0.2	0.3	0.4
MSAE	0.01	0.036	<b>0.034</b>	<u>0.035</u>
	0.02	<u>0.035</u>	<b>0.034</b>	0.036
	0.03	<u>0.035</u>	0.037	0.036

Next we perform additional ablation studies to confirm our chosen values of  $\beta$  and  $\delta$  which appear in Eq. (10) and Eq. (13), respectively. Table 11 displays the validation results. The values  $\beta = 0.01$  and  $\delta = 0.3$  minimize the MSAE.

#### 6.5 Contrastive learning without rotational augmentation

We consider the case of a feature encoder trained without applying the second augmentation to contrastive patches  $\mathcal{Q}$ . Therefore, the only augmentation applied to generate augmented pairs is noise corruption. For such an encoder, we retrain the regression network with hyperparameters  $\alpha = 0.9$ ,  $\beta = 0.01$ ,  $\delta = 0.3$ ,  $\gamma = 12$  and CBS = 512. We recover a MSAE of 0.037, compared to a lower MSAE of 0.034 with the second, rotational, augmentation. This indicates the importance of the rotation augmentation to the representation learning process.

#### 6.6 Evaluation without post-processing refinement

Finally, we observed our iterative filtering strategy would shrink the filtered point cloud, which necessitates Taubin smoothing-like inflation (TS). PointCleanNet (PCN) [21] also used a similar post-step, so we compare with it. Furthermore, to maximize the interaction between normal estimation and point cloud filtering, a point update strategy

TABLE 12

Results on the validation set for PCN and our method when using post-processing Taubin Smoothing (TS) and Low Rank Matrix Approximation (LRMA) position update. Our method outperforms PCN on the filtering task, with and without the Taubin smoothing-like post processing step. Moreover, our method, unlike PCN, estimates point normals that are then used to further improve filtering results.

	Post Processing	PCN	Ours
MSAE	None	N/A	0.054
	TS	N/A	<b>0.034</b>
	TS+LRMA	N/A	<b>0.034</b>
CD ( $\times 10^{-5}$ )	None	2.83	2.10
	TS	1.71	1.63
	TS+LRMA	N/A	<b>1.43</b>

like that in [24] (LRMA) is used. Previous research rarely exploits this interconnected relationship between normal estimation and filtering. Table 12 shows that Taubin smoothing is important to PCN and our method, and our pure network output is better than PCN’s pure network output (CD:  $2.10 \times 10^{-5}$  vs  $2.83 \times 10^{-5}$ ).

## 7 LIMITATIONS

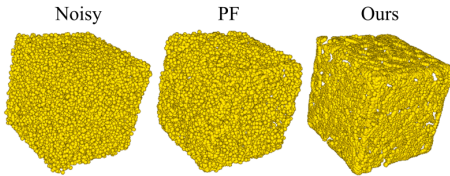


Fig. 10. Visual results for the Cube shape with 10K points and 0.8% Gaussian noise.

Similar to previous deep learning methods [22], our method cannot handle sparsely sampled point clouds well. Fig. 10 demonstrates the filtering results for the Cube shape with 10K points (0.8% Gaussian noise). Our method is able to recover some sharp features such as edges, yet fails to recover the planar surfaces. Pointfilter, by comparison, does not recover edge information.

## 8 CONCLUSION

In this paper, we introduced a deep learning method that jointly learns point displacements and their respective normals from point cloud data. This approach of a simultaneous inference of both position and normal information yields accurate filtered positions and normals. Our method displays an ability to preserve sharp features and fine details. This ability is derived from the 3D patch based contrastive learning which faithfully outputs patch representations based on their distinct geometric structure and the introduced joint loss that encourages the effective learning of normal and position information. We conduct extensive experiments and show that our method generally performs better than state-of-the-art methods in terms of both point cloud filtering and normal estimation. It also generalizes well between both CAD-like and non-CAD-like point clouds.

## REFERENCES

- [1] C. Luo, X. Yang, and A. Yuille, “Self-supervised pillar motion learning for autonomous driving,” in *CVPR*, 2021, Conference Proceedings.
- [2] B. Yasemin, B. Mårten, G. Gabriela Zarzar, E. Johannes, E. Carl Henrik, and K. Danica, “Visual and tactile 3d point cloud data from real robots for shape modeling and completion,” *Data in Brief*, vol. 30, no. 105335-, 2020.
- [3] Y. Kim, K. Kwon, and D. Mun, “Mesh-offset-based method to generate a delta volume to support the maintenance of partially damaged parts through 3d printing,” *Journal of Mechanical Science and Technology*, vol. 35, no. 7, pp. 3131–3143, 2021.
- [4] P. R. W. Urech, M. Dissegna, C. Girot, and A. Grêt-Regamey, “Point cloud modeling as a bridge between landscape design and planning,” *Landscape and Urban Planning*, vol. 203, p. 103903, 2020.
- [5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992.
- [6] N. Mitra and A. Nguyen, “Estimating surface normals in noisy point cloud data,” in *Proceedings of the nineteenth annual symposium on Computational geometry*, ser. SCG ’03. Association for Computing Machinery, 2003, Conference Proceedings, p. 322–328.
- [7] M. Pauly, M. Gross, and L. Kobbelt, “Efficient simplification of point-sampled surfaces,” *IEEE Visualization*, 2002. *VIS 2002.*, pp. 163–170, 2002.
- [8] M. Yoon, Y. Lee, S. Lee, I. Ivriissimtzis, and H. Seidel, “Surface and normal ensembles for surface reconstruction,” *Comput. Aided Des.*, vol. 39, pp. 408–420, 2007.
- [9] N. Amenta and M. Bern, “Surface reconstruction by voronoi filtering,” *Discrete & Computational Geometry*, vol. 22, pp. 481–504, 1999.
- [10] 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*, ser. ACM International Conference Proceeding Series, A. G. Belyaev and M. Garland, Eds., vol. 257. Eurographics Association, 2007, Conference Proceedings, pp. 39–48.
- [11] T. Dey, G. Li, and J. Sun, “Normal estimation for point clouds: a comparison study for a voronoi based method,” *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics*, 2005., pp. 39–46, 2005.
- [12] D. Lu, X. Lu, Y. Sun, and J. Wang, “Deep feature-preserving normal estimation for point cloud filtering,” *Computer-Aided Design*, vol. 125, 2020.
- [13] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, “Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 104–10 112, 2019.
- [14] R. Zhu, Y. Liu, Z. Dong, Y. Wang, T. Jiang, W. Wang, and B. Yang, “AdaFit: Rethinking learning-based normal estimation on point clouds,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 6118–6127.
- [15] J. E. Lenssen, C. Osendorfer, and J. Masci, “Deep iterative surface normal estimation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020, Conference Proceedings, pp. 11 244–11 253.
- [16] A. C. Öztireli, G. Guennebaud, and M. Gross, “Feature preserving point set surfaces based on non linear kernel regression,” *Computer Graphics Forum*, vol. 28, 2009.
- [17] Y. Sun, S. Schaefer, and W. Wang, “Denoising point sets via 10 minimization,” *Comput. Aided Geom. Des.*, vol. 35-36, pp. 2–15, 2015.
- [18] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, “L1-sparse reconstruction of sharp point set surfaces,” *ACM Trans. Graph.*, vol. 29, pp. 135:1–135:12, 2010.
- [19] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, “Parameterization-free projection for geometry reconstruction,” *ACM SIGGRAPH 2007 papers*, 2007.
- [20] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, “Consolidation of unorganized point clouds for surface reconstruction,” *ACM SIGGRAPH Asia 2009 papers*, 2009.
- [21] M.-J. Rakotosaona, V. L. Barbera, P. Guerrero, N. Mitra, and M. Ovsjanikov, “Pointcleannet: Learning to denoise and remove outliers from dense point clouds,” *Computer Graphics Forum*, vol. 39, 2020.



- [22] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, pp. 2015–2027, 2021.
- [23] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3d point cloud cleaning," in *Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, Conference Proceedings, pp. 52–60.
- [24] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3d geometry filtering," *IEEE transactions on visualization and computer graphics*, vol. PP, 2020.
- [25] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin, "Robust normal estimation for point clouds with sharp features," *Comput. Graph.*, vol. 34, pp. 94–106, 2010.
- [26] J. Zhang, J. Duan, K. Tang, J. Cao, and X. Liu, "Quality point cloud normal estimation by guided least squares representation," *IEEE Access*, vol. 8, pp. 101 580–101 590, 2020.
- [27] J. Zhang, J. Cao, X. Liu, J. Wang, J. Liu, and X. Shi, "Point cloud normal estimation via low-rank subspace clustering," *Comput. Graph.*, vol. 37, pp. 697–706, 2013.
- [28] J. Zhang, J. Cao, X. Liu, H. Chen, B. Li, and L. Liu, "Multi-normal estimation via pair consistency voting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, pp. 1693–1706, 2019.
- [29] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," *Computer Graphics Forum*, vol. 35, 2016.
- [30] C. Qi, H. Su, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.
- [31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, Conference Proceedings.
- [32] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. Mitra, "Pcpnet learning local shape properties from raw point clouds," *Computer Graphics Forum*, vol. 37, 2018.
- [33] Z. Wang and V. Prisacariu, "Neighbourhood-insensitive point cloud normal estimation network," in *BMVC 2020*, 2020, Conference Proceedings.
- [34] Y. Ben-Shabat and S. Gould, "Deepfit: 3d surface fitting via neural network weighted least squares," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Springer International Publishing, 2020, Conference Proceedings, pp. 20–34.
- [35] S. Wang, X. Liu, J. Liu, S. Li, and J. Cao, "Deep patch-based global normal orientation," *Computer-Aided Design*, vol. 150, p. 103281, 2022.
- [36] D. Levin, "The approximation power of moving least-squares," *Math. Comput.*, vol. 67, pp. 1517–1531, 1998.
- [37] R. Kolluri, "Provably good moving least squares," *ACM Trans. Algorithms*, vol. 4, pp. 18:1–18:25, 2005.
- [38] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, pp. 3–15, 2003.
- [39] A. Adamson and M. Alexa, "Point-sampled cell complexes," *ACM Trans. Graph.*, vol. 25, pp. 671–680, 2006.
- [40] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *SIGGRAPH 2007*, 2007, Conference Proceedings.
- [41] R. Preiner, O. Mattausch, M. Arikian, R. Pajarola, and M. Wimmer, "Continuous projection for fast I1 reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 33, pp. 1–13, 2014.
- [42] O. Remil, Q. Xie, X. Xie, K. Xu, and J. Wang, "Data driven sparse priors of 3d shapes," *Computer Graphics Forum*, vol. 36, 2017.
- [43] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "Pointpronets: Consolidation of point clouds with convolutional neural networks," *Computer Graphics Forum*, vol. 37, no. 2, pp. 87–99, 2018.
- [44] L. Yu, X. Li, C. W. Fu, P. A. Heng, and D. Cohen-Or, *EC-Net: An edge-aware point set consolidation network*, ser. Lecture Notes in Computer Science. Springer Verlag, 2018, vol. 11211 LNCS.
- [45] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 4583–4592.
- [46] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, pp. 161–163, 1992.
- [47] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, Conference Proceedings, pp. 1735–1742.
- [48] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, I. Hal, Daumé and S. Aarti, Eds., vol. 119. PMLR, 2020, Conference Proceedings, pp. 1597–1607.
- [49] J. Jiang, X. Lu, W. Ouyang, and M. Wang, "Unsupervised representation learning for 3d point cloud data," *arXiv preprint arXiv:2110.06632*, 2021.
- [50] S. Lal, M. Prabhudesai, I. Mediratta, A. W. Harley, and K. Fragkiadaki, "Coconets: Continuous contrastive 3d scene representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, Conference Proceedings, pp. 12 487–12 496.
- [51] B. Du, X. Gao, W. Hu, and X. Li, "Self-contrastive learning with hard negative sampling for self-supervised point cloud learning," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, Conference Proceedings, pp. 3133–3142.
- [52] M. Afham, I. Dissanayake, D. Dissanayake, A. Dharmasiri, K. Thilakarathna, and R. Rodrigo, "Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2022, Conference Proceedings, pp. 9892–9902.
- [53] A. Alliegro, D. Valsesia, G. Fracastoro, E. Magli, and T. Tommasi, "Denoise and contrast for category agnostic shape completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, Conference Proceedings, pp. 4629–4638.
- [54] M. Li, Y. Xie, Y. Shen, B. Ke, R. Qiao, B. Ren, S. Lin, and L. Ma, "Hybridcr: Weakly-supervised 3d point cloud semantic segmentation via hybrid contrastive regularization," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, Conference Proceedings, pp. 14910–14919.
- [55] J. Hou, B. Graham, M. Niessner, and S. Xie, "Exploring data-efficient 3d scene understanding with contrastive scene contexts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, Conference Proceedings, pp. 15 587–15 597.
- [56] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," *ArXiv*, vol. abs/2007.10985, 2020.
- [57] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, Conference Proceedings.
- [58] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Trans. Graph.*, vol. 35, no. 6, p. Article 232, 2016.
- [59] A. Serna, B. Marcotegui, F. Goulette, and J.-E. Deschaud, "Paris-rue-madame database - a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *ICPRAM*, 2014.
- [60] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," *arXiv preprint arXiv:2109.13410*, 2021.
- [61] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "Blensor: Blender sensor simulation toolbox," in *Advances in Visual Computing*. Springer Berlin Heidelberg, 2011, Conference Proceedings, pp. 199–208.
- [62] X. Lu, Z. Deng, and W. Chen, "A robust scheme for feature-preserving mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 3, pp. 1181–1194, 2016.





**Dasith de Silva Edirimuni** received the BSc degree in Physics from Hardin-Simmons University, Texas, USA in 2013 and the MSc degree in Physics from the University of Melbourne, Australia, in 2017. In 2020 he received the Master of Information Technology degree from Swinburne University, Australia and is currently a PhD student in Information Technology at Deakin University, Australia. His research interests include 3D computer vision areas such as point cloud filtering and scene understanding.



**Xuequan Lu** received the PhD degree from Zhejiang University, China, in June 2016. He is currently an assistant professor with the School of Information Technology, Deakin University, Australia. He has spent more than two years as a research fellow in Singapore. His research interests include visual computing, for example, geometry modeling, processing and analysis, animation or simulation, and 2D data processing and analysis. He was a member of the International Program Committee of GMP 2021, a PC

member in CVM 2020, and a TPC member in ICONIP 2019.



**Gang Li** (Senior Member, IEEE) is currently an Associate Professor with the School of Information Technology, Deakin University, Melbourne, VIC, Australia. His research interests include data mining, data privacy, causal discovery, and business intelligence.



**Antonio Robles-Kelly** received the BEng degree in Electronics and Telecommunications with honours in 1998 and the PhD degree in Computer Science from the University of York, United Kingdom, in 2003. He was in York until December 2004 as a research associate under the MathFitEPSRC framework. In 2005, he moved to Australia and took a research scientist appointment with National ICT Australia (NICTA). In 2006 he became the project leader of the Imaging Spectroscopy team at NICTA and, from

2007 to 2009, he was a postdoctoral research fellow of the Australian Research Council. In 2016, he joined CSIRO where he is a principal researcher with Data61 and, in 2018, became a Machine Learning and Artificial Intelligence professor at Deakin University, Australia. His research has been applied to areas such as biosecurity, forensics, food quality assurance, and biometrics and is now being deployed by CSIRO under the trademark of Scyllarus ([www.scyllarus.com](http://www.scyllarus.com)). He has served as the president of the Australian Pattern Recognition Society (APRS) and is an associate editor of the Pattern Recognition Journal and IET Computer Vision Journal. He is a senior member of the IEEE, the president of the TC2 (Technical Committee on structural and syntactical pattern recognition) of the International Association for Pattern Recognition (IAPR) and an adjunct associate professor at the ANU. He has also been a technical committee member, area, and general chair of several mainstream computer vision and pattern recognition conferences.

# Contrastive Learning for Joint Normal Estimation and Point Cloud Filtering: Supplementary

Dasith de Silva Edirimuni, Xuequan Lu, *Senior Member, IEEE*, Gang Li, *Senior Member, IEEE*, and Antonio Robles-Kelly, *Senior Member, IEEE*

Here we provide supplementary material to the main paper. In particular, we provide additional information on the following:

- 1) Sharp feature points classification
- 2) Varying density point clouds
- 3) Further comparisons on synthetic and real-world scan data

## A SHARP FEATURE POINTS CLASSIFICATION

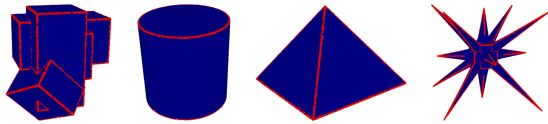


Fig. 11. Sharp feature points are given in red.

In order to explicitly analyze the performance of normal estimation and filtering methods at sharp features on our synthetic data, we must first determine which points correspond to sharp features, i.e., points that lie on anisotropic surfaces. To achieve this, we exploit the ground truth normal information for each clean synthetic point cloud. For each point in the clean point cloud, we consider neighborhoods of their 10 nearest neighbors. Nearest neighbors whose normals make an angle  $\pi/6 < \theta < 5\pi/6$  with the neighborhood's central point's normal are classified as feature points. Here, the threshold angle between normals for determining a sharp feature point is  $\pi/6$ . If the angle is below the threshold, the surface varies smoothly. An angle greater than  $5\pi/6$  most likely corresponds to a normal of a point on a surface parallel to that of the central point and is not considered. This scheme allows us to extract points along sharp edges and corners. The unique indices corresponding to these points (on ground truth and filtered point clouds) are used to determine MSAE and Point2Surface distance values at sharp features.

- D. de Silva Edirimuni, X. Lu, G. Li and A. Robles-Kelly are with the School of Information Technology, Deakin University, Waurn Ponds, Victoria, 3216, Australia (e-mail: {dtdesilva, xuequan.lu, gang.li, antonio.robles-kelly}@deakin.edu.au). A. Robles-Kelly is also with the Defense Science and Technology Group, Australia.

Manuscript received Month Day, Year; revised Month Day, Year.  
(Corresponding author: Xuequan Lu.)

## B VARYING DENSITY POINT CLOUDS

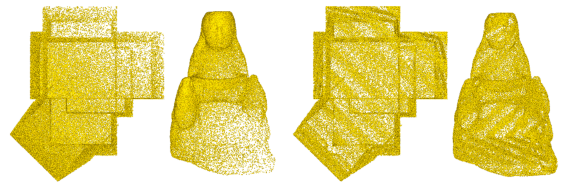


Fig. 12. Left: Point clouds sampled with the gradient varying density regime. Right: Point clouds sampled with the striped varying density regime.

Varying density point clouds are obtained by applying two different sampling regimes, gradient and striped, to the original test meshes. Fig. 12 illustrates point clouds from these two regimes. Thereafter, Gaussian noise of  $\sigma = 0.8\%$  is added to each point cloud in order to evaluate the robustness of normal estimation and filtering methods on noisy, varying density point clouds.

## C FURTHER COMPARISONS ON SYNTHETIC AND REAL-WORLD SCAN DATA

Fig. 13 displays normal estimation and filtering results on additional shapes within our synthetic data for Gaussian noise with standard deviation of 0.8% of the bounding box diagonal and Table 13 details performance across all noise levels, for each respective shape. In the normal estimation task, we outperform other methods and generalize well between CAD shapes such as StarSharp and non-CAD shapes such as Netsuke, especially at higher noise levels. For the filtering task, we perform competitively, and our method is able to reliably recover sharp features on CAD shapes such as StarSharp and PipeCurve and fine details on non-CAD shapes such as Netsuke.

On the Kinect v1 and v2 datasets, we outperform other methods as depicted by Fig. 14. The average Chamfer distance and Point2Surface results are given in Table 4 of the main paper. Fig. 15 provides an evaluation on 2 scenes of the Kitti-360 dataset. This dataset contains sparse point clouds with high amounts of real world noise which makes it difficult to filter these scenes. However, as shown in scene 1, we perform better at filtering noise and retrieving the underlying shapes of parked cars as compared to other methods.

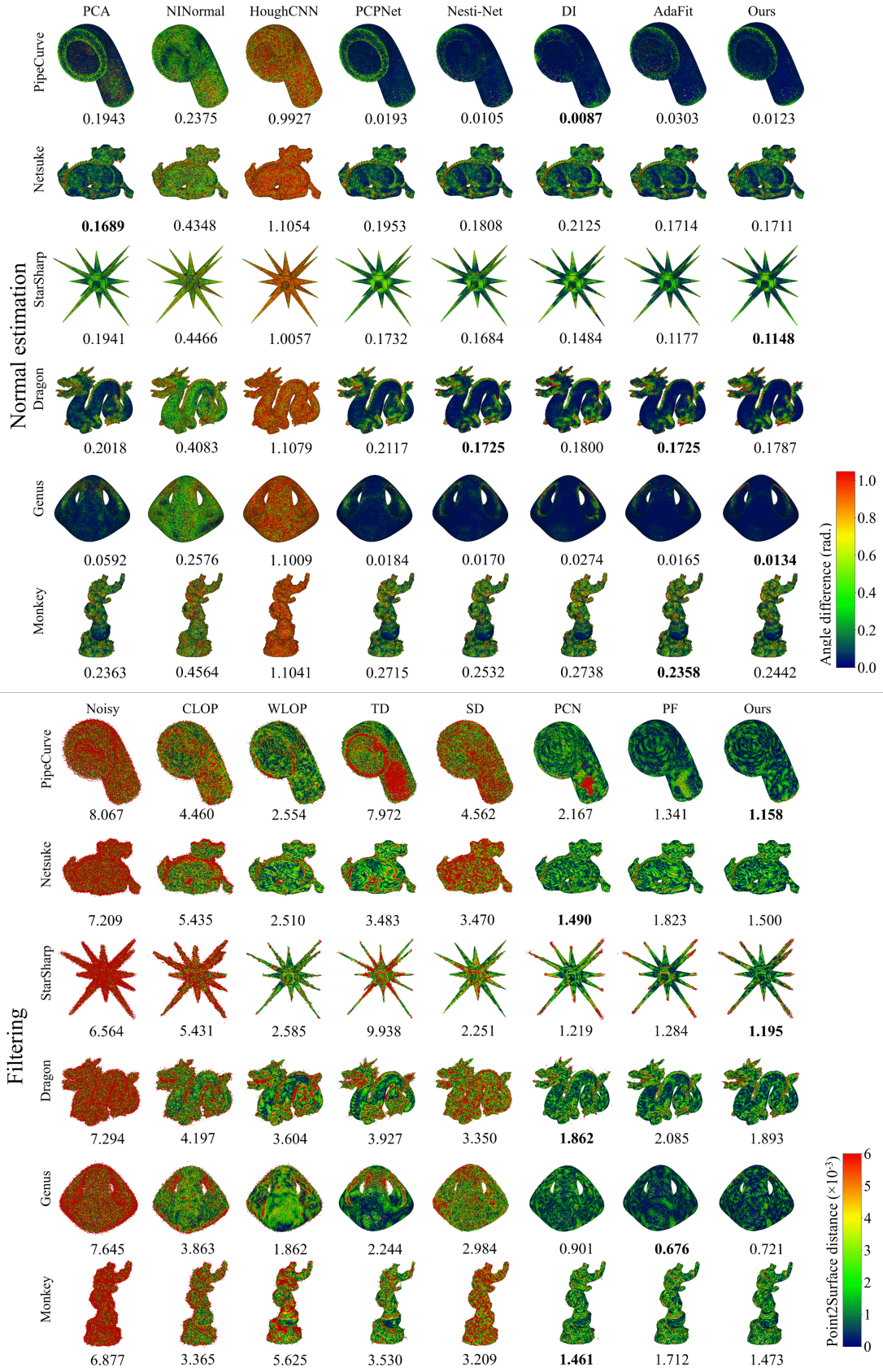


Fig. 13. Normal estimation (top half) and filtering (bottom half) results on shapes with 0.8% Gaussian noise with respect to the bounding box diagonal. For normal estimation, the respective mean squared angular error (MSAE) is given below each shape and the heat map corresponds to the angle difference at each point. For filtering, the Chamfer distance ( $\times 10^{-5}$ ) is given below each shape and the heat map corresponds to the scale normalized Point2Surface distance ( $\times 10^{-3}$ ).

TABLE 13

Individual MSAE and Chamfer distance values for shapes presented in Fig. 13, at different noise levels. The standard deviation  $\sigma$  is with respect to the bounding box diagonal of the clean point cloud. Top results in bold and second best results are underlined.

Shape	$\sigma$	Normal estimation - MSAE					Filtering - Chamfer distance ( $\times 10^{-5}$ )			
		PCPNet	Nesti-Net	DI	AdaFit	Ours	Noisy	PCN	PF	Ours
PipeCurve	0.6%	0.0169	0.0068	<b>0.0056</b>	0.0146	0.0115	5.190	1.482	1.015	<b>1.024</b>
	0.8%	0.0193	<u>0.0105</u>	<b>0.0087</b>	0.0303	0.0123	8.067	2.167	<u>1.341</u>	<b>1.158</b>
	1.1%	0.0230	0.0173	0.0132	0.0689	<b>0.0121</b>	12.922	5.041	<u>4.691</u>	<b>1.651</b>
	1.5%	0.0304	0.0324	<u>0.0214</u>	0.0597	<b>0.0196</b>	19.805	11.395	<u>11.26</u>	<b>3.214</b>
	2.0%	0.0398	0.0433	<u>0.0280</u>	0.0421	<b>0.0272</b>	30.269	25.024	<u>16.563</u>	<b>7.103</b>
Netsuke	0.6%	0.1628	<u>0.1552</u>	0.1793	<b>0.1440</b>	0.1591	4.555	<u>1.263</u>	1.545	<b>1.237</b>
	0.8%	0.1953	0.1808	0.2125	0.1714	<b>0.1712</b>	7.209	1.49	1.823	<b>1.500</b>
	1.1%	0.2462	0.2163	0.2526	<u>0.2155</u>	<b>0.1964</b>	12.209	<u>2.160</u>	2.294	<b>2.040</b>
	1.5%	0.3122	<u>0.2736</u>	0.3054	0.2763	<b>0.2459</b>	20.832	4.585	<b>3.171</b>	3.826
	2.0%	0.3714	<u>0.3447</u>	0.3631	0.3463	<b>0.3044</b>	33.812	14.521	<b>5.077</b>	9.593
StarSharp	0.6%	0.1371	0.1257	0.1029	<b>0.0801</b>	0.1039	4.002	0.93	0.895	<b>0.822</b>
	0.8%	0.1732	0.1684	0.1484	0.1177	<b>0.1148</b>	6.564	<u>1.219</u>	1.284	<b>1.195</b>
	1.1%	0.2280	0.2309	0.1977	<u>0.1671</u>	<b>0.1338</b>	11.572	2.416	<b>1.980</b>	2.597
	1.5%	0.2776	0.2811	0.2558	<u>0.2226</u>	<b>0.1622</b>	20.749	7.205	<b>3.535</b>	<u>6.190</u>
	2.0%	0.3219	0.3206	0.3020	<u>0.2849</u>	<b>0.1696</b>	36.616	20.958	<b>5.770</b>	<u>14.617</u>
Dragon	0.6%	0.1562	<b>0.1302</b>	0.1337	<u>0.1310</u>	0.1526	4.644	1.494	1.613	<b>1.443</b>
	0.8%	0.2117	0.1725	0.1800	<b>0.1725</b>	0.1787	7.294	<u>1.862</u>	2.085	<b>1.893</b>
	1.1%	0.2857	<u>0.2279</u>	0.2329	0.2287	<b>0.2087</b>	12.183	<u>2.858</u>	2.863	<b>2.609</b>
	1.5%	0.3765	<u>0.3170</u>	0.3301	0.3245	<b>0.2772</b>	20.215	6.577	4.491	<b>4.394</b>
	2.0%	0.4768	<u>0.4279</u>	0.4386	0.4240	<b>0.3684</b>	32.734	19.231	<u>6.269</u>	9.000
Genus	0.6%	0.0128	0.0119	0.0218	<b>0.0103</b>	0.0112	4.720	0.817	0.642	<b>0.638</b>
	0.8%	0.0184	0.0170	0.0274	0.0165	<u>0.0134</u>	7.645	0.901	<b>0.676</b>	0.721
	1.1%	0.0318	<u>0.0303</u>	0.0394	0.0370	<b>0.0178</b>	13.249	1.370	<b>0.909</b>	<u>0.973</u>
	1.5%	0.0570	<u>0.0486</u>	0.0526	0.0628	<b>0.0248</b>	22.718	3.647	<u>2.025</u>	<b>2.000</b>
	2.0%	0.1022	<u>0.0865</u>	0.0724	0.1119	<b>0.0478</b>	37.945	14.607	<b>3.835</b>	6.064
Monkey	0.6%	0.2225	0.2113	0.2298	<b>0.1953</b>	0.2174	4.371	1.220	1.393	<b>1.150</b>
	0.8%	0.2715	0.2532	0.2738	<b>0.2358</b>	<u>0.2442</u>	6.877	<u>1.461</u>	1.712	<b>1.473</b>
	1.1%	0.3372	0.3083	0.3297	0.2964	<b>0.2788</b>	11.706	<u>2.112</u>	2.175	<b>1.978</b>
	1.5%	0.4125	<u>0.3719</u>	0.3948	0.3745	<b>0.3326</b>	19.968	4.956	<b>2.896</b>	<u>3.621</u>
	2.0%	0.4856	<u>0.4516</u>	0.4754	0.4626	<b>0.4103</b>	32.578	16.503	<b>4.651</b>	<u>9.033</u>

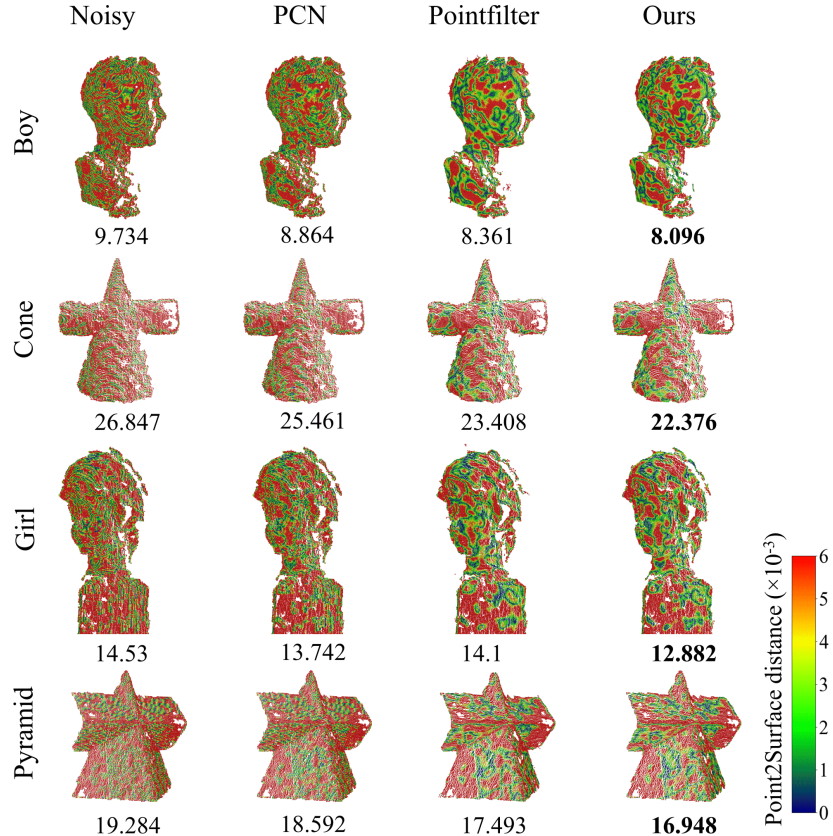


Fig. 14. Filtering results on the Kinect v1 and v2 datasets, the Chamfer distance ( $\times 10^{-5}$ ) is given below each shape and the heat map corresponds to the scale normalized Point2Surface distance ( $\times 10^{-3}$ ).



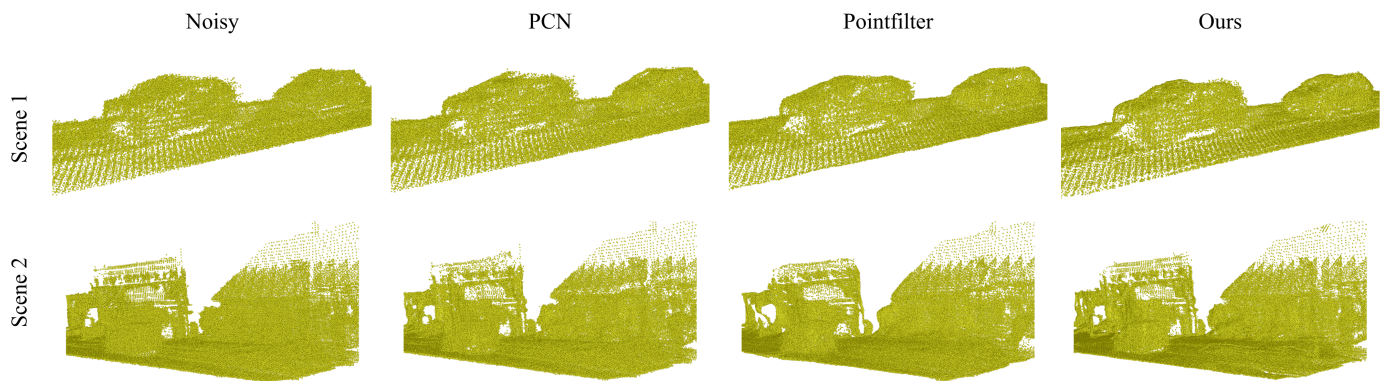


Fig. 15. Visual filtering results on 2 scans from the Kitti-360 dataset. Scene 1 corresponds to vehicles parked on grass, next to a road while scene 2 depicts parked vehicles and houses along a street.