Volume xx (200y), Number z, pp. 1–17

POINTCLEANNET: Learning to Denoise and Remove Outliers from Dense Point Clouds

Marie-Julie Rakotosaona[1] Vittorio La Barbera[2] Paul Guerrero[2] Niloy J. Mitra[2] Maks Ovsjanikov[1]

[1]LIX, École Polytechnique, CNRS [2]University College London

Abstract

Point clouds obtained with 3D scanners or by image-based reconstruction techniques are often corrupted with significant amount of noise and outliers. Traditional methods for point cloud denoising largely rely on local surface fitting (e.g., jets or MLS surfaces), local or non-local averaging, or on statistical assumptions about the underlying noise model. In contrast, we develop a simple data-driven method for removing outliers and reducing noise in unordered point clouds. We base our approach on a deep learning architecture adapted from PCPNet, which was recently proposed for estimating local 3D shape properties in point clouds. Our method first classifies and discards outlier samples, and then estimates correction vectors that project noisy points onto the original clean surfaces. The approach is efficient and robust to varying amounts of noise and outliers, while being able to handle large densely-sampled point clouds. In our extensive evaluation, both on synthesic and real data, we show an increased robustness to strong noise levels compared to various state-of-the-art methods, enabling accurate surface reconstruction from extremely noisy real data obtained by range scans. Finally, the simplicity and universality of our approach makes it very easy to integrate in any existing geometry processing pipeline. Both the code and pre-trained networks can be found on the project page[†].

1. Introduction

Raw 3D point clouds obtained directly from acquisition devices such as laser scanners or as output of a reconstruction algorithm (e.g., image-based reconstruction) are regularly contaminated with noise and outliers. The first stage of most geometry processing workflows typically involves *cleaning* such raw point clouds by discarding the outlier samples and denoising the remaining points to reveal the (unknown) scanned surface. The clean output is then used for a range of applications like surface reconstruction, shape matching, model retrieval, etc.

Any good point cloud cleanup algorithm should (i) *balance between denoising and feature-preservation*, i.e., remove outliers and noise while retaining data fidelity by preserving sharp edges and local details of the underlying scanned surface; (ii) *be self-tuning*, i.e., not require as input precise estimates of the noise model or statistics of the unknown scanned surface (e.g., local surface type or curvature characteristics); (iii) *be invariant to permutation and rigid transform applied to the pointset*, i.e., the denoised output should not depend on angle of scanning or choice of coordinate system; and (iv) *avoid unnecessarily degrading the input*, i.e., leave the points on the scanned surface if the input happens to be noise-free. Note that the last criterion implies that the algorithm should not oversmooth the output if the algorithm is iterated multiple times.

Decades of research have produced many variants of denoising approaches targeted for different surface types and noise models (see survey [HJW*17]). Such approaches can be broadly categorized as: classifying points as outliers using statistical methods, e.g., [Agg15]; projecting points to estimated local surfaces (e.g., MLS surface, jet-fitting, etc.) [FCOS05, CP05, CP07]; consolidating similar patches to cancel out iid noise perturbations (e.g., non-local means, dictionary-based sparse coding), e.g., [Dig12]; or, local smoothing using auxiliary input information (e.g., bilateral smoothing) [HWG*13], among many others. Unfortunately, there is no single winner among these methods. The choice of algorithm and its parameters often depends on the scanned surface and the noise characteristics of the acquisition setup. Given that the complexity of the underlying geometry and the noise characteristics are, at best, partially known at acquisition time, choosing an optimal algorithm with associated parameters is typically an iterative trial-and-error process.

Inspired by the recent successes of applying deep learning techniques for the analysis and processing of geometric data, including [MBBV15, BBL*17, WHC*16] among many others, and especially the seminal works designed for learning directly on point clouds [QSMG17, WSL*18], in this paper, we present POINT-

[†] https://github.com/mrakotosaon/pointcleannet

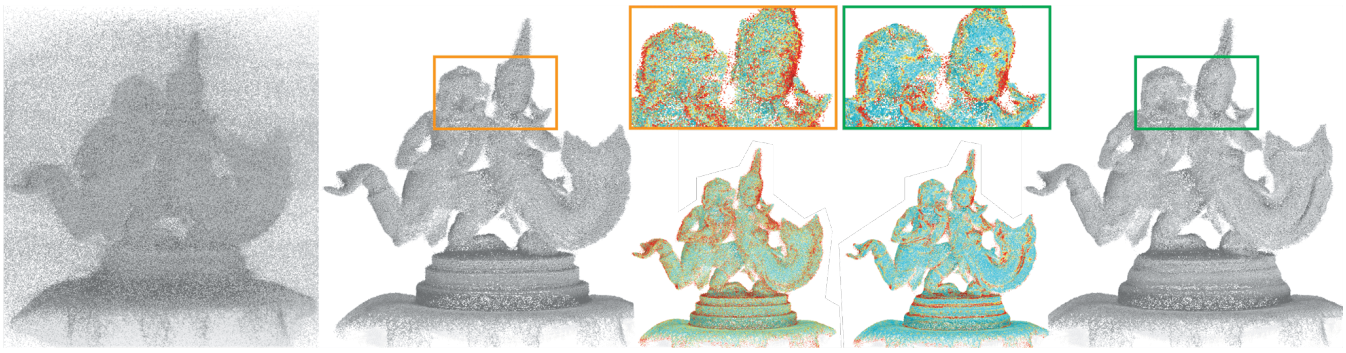

Figure 1: *We present* POINTCLEANNET*, a two-stage network that takes a raw point cloud (left) and first removes outliers (middle) and then denoises the remaining pointset (right). Our method, unlike many traditional approaches, is parameter-free and automatically discovers and preserves high-curvature features without requiring additional information about the underlying surface type or device characteristics. Here, point clouds are colored based on error compared to the ground truth point cloud (blue denoting low error, red denoting high error).*

CLEANNET, a simple data-driven denoising approach. Specifically, we design a two stage point cloud cleaning network based on the recently proposed PCPNet architecture [GKOM18] to estimate robust local features and use this information to denoise the point cloud. At training time, a variety of surface patches extracted from a set of shapes is synthetically corrupted with outliers and noise of varying magnitudes (including zero noise). This artificially corrupted set is then used to train POINTCLEANNET. Our two-stage method first removes outlier samples and then estimates correction vectors for the remaining points. Figure 1 shows an example on a raw real-world scanned point cloud from the ETHZ dataset [WKZ*16].

The process is enabled by a novel loss function that effectively cleans pointsets without requiring explicit information about the underlying surface or noise characteristics. Intuitively, the network learns to identify local noise-free patches based on estimated features extracted from corresponding raw pointsets and proposes per-point correction vectors. In other words, the network implicitly builds a dictionary of local surface patches in the form of local learned features and uses it to classify input points as outliers and project the remaining ones onto an ensemble of dictionary patches. At test time, our denoising network directly consumes raw input point clouds, classifies and discards outlier measurements, and denoises the remaining points. The approach is simple to train and use, and does not expect the user to provide parameters to characterize the surface or noise model. Additionally, unlike traditional approaches, our denoising network can easily be adapted to particular shape families and non-standard noise models. Similarly to other data-driven techniques, our framework is based on paired noisy-clean data, which we generate synthetically during training, and is thus unlikely to succeed for significantly different test noise.

We qualitatively and quantitatively evaluate POINTCLEANNET on a range of synthetic datasets (with access to groundtruth surfaces) and real world datasets. In our extensive tests, our approach performed better than a variety of state-of-the-art denoising approaches (even with manually-tuned parameters) across both shape and medium to high noise variations. Additionally, the simplicity and universality of our approach makes it very easy to integrate in any existing geometry processing workflow.

2. Related Work

Point cloud denoising and outlier removal have a long and rich history in diverse areas of computer science and a full overview is beyond the scope of the current article. Below, we briefly review the main general trends for addressing these problems, while concentrating on solutions most closely related to ours, and refer the interested reader to a recent survey [HJW*17].

Outlier removal. The earliest classical approaches for outlier detection, classification and removal have been proposed primarily in the statistics and data mining communities, in the general setting of point clouds in arbitrary dimensions, with several monographs dedicated specifically to this topic [Pin95, BG05, RH11, Agg15]. These methods are typically based on robust local statistics and most often come with rigorous theoretical guarantees. At the same time, their generality often comes at a cost, as purely statistical methods are often not adapted to the specific features found in geometric 3D shapes, and in most cases require non-trivial parameter tuning.

More recently, several approaches have been proposed for outlier detection, with emphasis on utility for 3D point clouds, arising e.g., from acquisition data, including [CCSM11, GMM13, WKZ*16]. The two former methods are implemented in widely used libraries such as CGAL and have also been used in the context of surface reconstruction from noisy point clouds [GCSA13]. These approaches are very robust, but are also based on setting critical parameters or rely on using additional information such as color [WKZ*16]. This makes it difficult to apply them, for example, across general noise models, without additional user input and tuning of parameters.

Local surface fitting, bilateral filtering. Denoising and outlier removal also arise prominently, and have therefore been considered in the context of surface fitting to noisy point clouds, including the widely-used Moving Least Squares (MLS) approach and its robust variants [ABCO*03, MVdF03, FCOS05, ÖGG09, GP11]. Similarly, other local fitting approaches have also been used for point cloud denoising, using robust jet-fitting with reprojection [CP05, CP07] or various forms of bilateral filtering on point clouds [HWG*13, DDF17], which take into account both point coordinates and normal directions for better preservation of edge fea-

tures. A closely related set of techniques is based on sparse representation of the point normals for better feature preservation [AGCO10, SSW15, MC17]. Denoising is then achieved by projecting the points onto the estimated local surfaces. These techniques are very robust for small noise but can lead to significant over smoothing or over-sharpening for high noise levels [MC17, HJW*17].

Non-local means, dictionary-based methods. Another very prominent category of methods, inspired in part from image-based techniques consist in using non-local filtering based most often on detecting similar shape parts (patches) and consolidating them into a coherent noise-free point cloud [DG10, ZSW*10, Dig12, DCV14, ZCN*18]. Closely related are also methods, based on constructing "dictionaries" of shapes and their parts, which can then be used for denoising and point cloud filtering, e.g., [YLÖ*16, DVC18] (see also a recent survey of dictionary-based methods [LOM*18]). Such approaches are particularly well-suited for feature-preserving filtering and avoid excessive smoothing common to local methods. At the same time, they also require careful parameter setting and, as we show below, are difficult to apply across a wide variety of point cloud noise and artefacts.

Denoising in images. Denoising has also been studied in depth in other domains such as for images, with a wide variety of techniques based on both local filtering, total variation smoothing and non-local including dictionary-based methods [BCM05, EA06, CCC*10, Mai10, Ela10].

More recently, to address the limitations mentioned above, and inspired by the success of deep learning for other tasks, several learning-based denoising methods have also been proposed for both images [ZZC*17, ZZGZ17, JMFU17] and more recently meshes [WLT16, BBZ*18], among others. These methods are especially attractive, since rather than relying on setting parameters, they allow the method to learn the correct model from data and adapt for the correct noise setting at test time, without any user intervention. In signal processing literature, it is widely believed that image denoising has reached close to optimal performance [CM10, LN11]. One of our main motivations is therefore to show the applicability of this general idea, and especaially the supervised approaches such as [WLT16], that learn them from a set of noisy meshes and their ground-truth counterparts, to the setting of 3D point clouds.

Learning in Point Clouds. Learning-based approaches, and especially those based on deep learning, have recently attracted a lot of attention in the context of Geometric Data Analysis, with several methods proposed specifically to handle point cloud data, including PointNet [QSMG17] and several extensions such as PointNet++ [QYSG17] and Dynamic Graph CNNs [WSL*18] for shape segmentation and classification, PCPNet [GKOM18] for normal and curvature estimation, P2P-Net [YHCOZ18] and PU-Net [YLF*18] for cross-domain point cloud transformation and upsampling respectively. Other, convolution-based architectures have also been used for point-based filtering, including most prominently the recent PointProNet architecture [RÖPG18], designed for consolidating input patches, represented via height maps with respect to a local frame, into a single clean point set, which can be used for surface reconstruction. Although such an approach has the advantage of leveraging image-based denoising solutions, error creeps in in the local normal estimation stage, especially in the presence of noise and outliers.

Unlike these techniques, our goal is to train a general-purpose method for removing outliers and denoising point clouds, corrupted with potentially very high levels of structured noise. For this, inspired by the success of PCPNet [GKOM18] for normal and curvature estimation, we propose a simple framework aimed at learning to both classify outliers and to displace noisy point clouds by applying an adapted architecture to point cloud patches. We show through extensive experimental evaluation that our approach can handle a wide range of artefacts, while being applicable to dense point clouds, without any user intervention.

3. Overview

As a first step in digitizing a 3D object, we usually obtain a set of approximate point samples of the scanned surfaces. This *point cloud* is typically an intermediate result used for further processing, for example to reconstruct a mesh or to analyze properties of the scanned object. The quality of these downstream applications depends heavily on the quality of the point cloud. In real-world scans, however, the point cloud is usually degraded by an unknown amount of outliers and noise. We assume the following point cloud formation model:

$$\mathbb{P}' = \{p_i'\} = \{p_i + n_i\}_{p_i \in \mathbb{P}} \cup \{o_j\}_{o_j \in \mathbb{O}}, \tag{1}$$

where $\mathbb{P}'$ is the observed noisy point cloud, $\mathbb{P}$ are perfect surface samples (i.e., $p_i \in \mathcal{S}$ lying on the scanned surface $\mathcal{S}$), n_i is additive noise, and $\mathbb{O}$ is the set of outlier points. We do not make any assumptions about the noise model n or the outlier model $\mathbb{O}$. The goal of our work is to take the low-quality point cloud $\mathbb{P}'$ as input, and output a higher quality point cloud closer to $\mathbb{P}$, that is better suited for further processing. We refer to this process as *cleaning*. We split the cleaning into two steps: first we remove outliers, followed by an esimation of per-point displacement vectors that denoise the remaining points:

$$\tilde{\mathbb{P}} = \{p_i' + d_i\}_{p_i' \in \mathbb{P}' \setminus \tilde{\mathbb{O}}}, \tag{2}$$

where $\tilde{\mathbb{P}}$ is the output point cloud, d are the displacement vectors and $\tilde{\mathbb{O}}$ the outliers estimated by our method. We first discuss our design choices regarding the desirable properties of the resulting point cloud and then how we achieve them.

Approach. Traditional statistical scan cleaning approaches typically make assumptions about the scanned surfaces or the noise model, which need to be manually tuned by the user to fit a given setting. This precludes the use of these methods by non-expert users or in casual settings. One desirable property of any cleaning approach is therefore robustness to a wide range of conditions without the need for manual parameter tuning. Recently, deep learning approaches applied to point clouds [QSMG17, QYSG17, WSL*18, GKOM18] have shown a remarkable increase in robustness compared to earlier hand-crafted approaches.

Most of these methods perform a *global* analysis of the point cloud and produce output that depends on the whole point cloud. This is necessary for global properties such as the semantic class,

but is less suited for tasks that only depend on local neighborhoods; processing the entire point cloud simultaneously is a more challenging problem, since the network needs to handle a much larger variety of shapes compared to working with small local patches, requiring more training shapes and more network capacity. Additionally, processing *dense* point clouds becomes more difficult, due to high memory complexity. In settings such as ours, local methods such as PCPNet [GKOM18] perform better. Both steps of our approach are based on the network architecture described in this method, due to its relative simplicity and competitive performance. We adapt this architecture to our setting (Section 4) and train it to perform outlier classification and denoising.

While our cleaning task is mainly a local problem, the estimated displacement vectors d need to be consistent across neighborhoods in order to achieve a smooth surface. With a local approach such as PCPNet, each local estimate is computed separately based on a different local patch. The difference in local neighborhoods causes inconsistencies between neighboring estimates that can be seen as *residual noise* in the result (see Figure 3). We therefore need a method to coordinate neighboring results. We observed that the amount of difference in local neighborhoods between neighboring estimates correlates with the noise model. Thus, the resulting residual noise has a similar noise model as the original noise, but with a smaller magnitude. This means we can iterate our network on the residual noise to keep improving our estimates. See Figure 2 for an overview of the full denoising approach. We will provide extensive experiments with different numbers of denoising iterations in Section 6.

Desirable properties of a point cloud. The two stages (i.e., outlier classification and denoising) of our method use different loss functions. The properties of our denoised point cloud are largely determined by these loss functions. Thus, we need to design them such that their optimium is a point cloud that has all desirable properties. We identify two key desirable properties: First, *all points should be as close as possible to the original scanned surface*. Second, *the points should be distributed as regularly as possible on the surface*. Note that we do not want the denoised points to exactly undo the additive noise and approximate the original perfect surface samples, since the component of the additive noise that is tangent to the surface cannot be recovered from the noisy point cloud.

Section 5 describes our loss functions, and in Section 6, we compare several alternative loss functions.

4. Cleaning Model

As mentioned above, our goal is to take a noisy point cloud $\mathbb{P}'$ and produce a cleaned point cloud $\hat{\mathbb{P}}$ that is closer to the unknown surface that produced the noisy samples. We treat denoising as a local problem: the result for each point $p'_i \in \mathbb{P}'$ only depends on a local neighborhood $\mathbb{P}'_i$ of radius r around the point. Focusing on local neighborhoods allows us to handle dense point clouds without losing local detail. Increasing the locality (or scale) radius r provides more information about the point cloud, at the cost of reducing the capacity available for local details. Unlike traditional analytic denoising approaches, a single neighborhood setting is robust to a wide range of noise settings, as we will demonstrate in Section 6. In all of our experiments we set r to 5% of the point cloud's bounding box diagonal.

We assume the point cloud formation model described in Equation (1), i.e., the noisy point cloud consists of surface samples with added noise and outliers. We then proceed in two stages: first, we train a non-linear function g that removes outliers:

$$\tilde{o}_i = g(\mathbb{P}'_i),$$

where $\tilde{o}_i$ is the estimated probability that point p'_i is an outlier. We add a point to the set of estimated outliers $\tilde{\mathbb{O}}$ if $\tilde{o}_i > 0.5$. After removing the outliers, we obtain the point cloud $\hat{\mathbb{P}} = \mathbb{P}' \setminus \tilde{\mathbb{O}}$. We proceed by defining a function f that estimates displacements for these remaining points to move them closer to the unknown surface:

$$d_i = f(\hat{\mathbb{P}}_i).$$

The final denoised points are obtained by adding the estimated displacements to the remaining noisy points: $\tilde{p}_i = \hat{p}_i + d_i$. Both f and g are modeled as deep neural networks with a PCPNet-based architecture. We next provide a short overview of PCPNet before describing our modifications.

A major challenge when applying deep learning methods directly to point clouds is achieving invariance to the permutation of the points: all permutations should produce the same result. Training a network to learn this invariance is difficult due to the exponential number of such permutations. As a solution to this problem, PointNet [QSMG17] proposes a network architecture that is order-invariant by design. However, PointNet is a global method, processing the whole point cloud in one forward iteration of the network. This results in a degraded performance for shape details. PCPNet [GKOM18] was proposed as a local variant of PointNet that is applied to local patches, gives better results for shape details, and is applicable to dense point clouds, possibly containing millions of points. We base our denoising architecture on PCPNet.

Creating a local patch. Given a point cloud $\mathbb{P} = \{p_1, \ldots, p_n\}$, the local patch $\mathbb{P}_i$ contains all the points within the constant radius r inside a ball centered around p_i. Using this patch as input, we want to compute the outlier probability $\tilde{o}_i$ and a displacement vector d_i, for the remaining non-outlier points. We first normalize this patch by centering it and scaling it to unit size. The PCPNet architecture requires patches to have a fixed number of points; like in the original paper, we pad patches with too few points with zeros and take a random subset from patches with too many points. Intuitively, this step, makes the network more robust to additional points.

Network architecture. An overview of our network architecture is shown in Figure 2. Given the normalized local patch $\mathbb{P}_i$, the network first applies a spatial transformer network [JSZ*15] that is constrained to rotations, called a quaternion spatial transformer network (QSTN). This is a small sub-network that learns to rotate the patch to a canonical orientation (note that this estimation implicitly learns to be robust to outliers and noise, similar to robust statistical estimation). At the end of the pipeline, the final estimated displacement vectors are rotated back from the canonical orientation to world space. The remainder of the network is divided into three main parts:

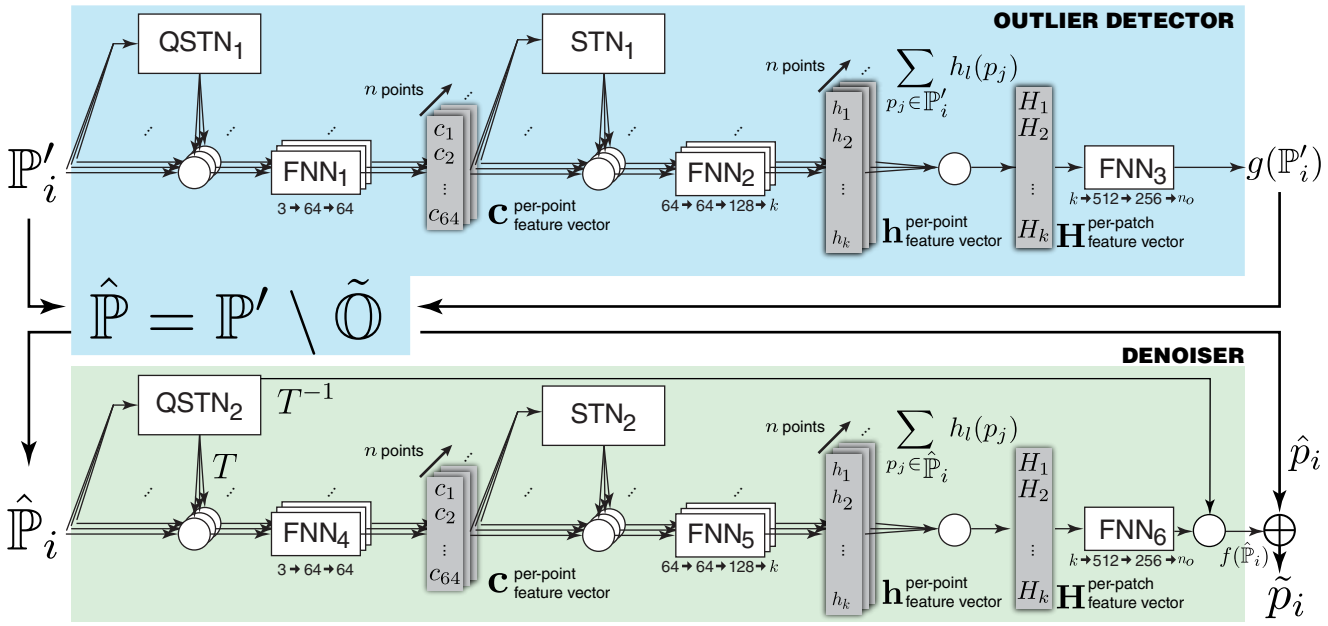

Figure 2: *Our two-stage point cloud cleaning architecture. (Top) Given a noisy point cloud $\mathbb{P}'$, we first apply a local outlier detection network that uses an architecture based on PointNet [QSMG17] and PCPNet [GKOM18] to detect and remove outliers to obtain $\hat{\mathbb{P}}$ (bottom). We then apply a second network, with a similar architecture, but a different loss, aimed at reducing the noise level in $\hat{\mathbb{P}}$ by estimating correcting displacement vectors, which results in the denoised point cloud $\tilde{\mathbb{P}}$. FNN and (Q)STN stand for fully connected and (Quaternion) Spatial Transformer networks [JSZ*15], similar to their definition and use in PCPNet [GKOM18].*

- a *feature extractor* $h(p)$ that is applied to each point in the patch separately,
- a *symmetric operation* $H(\mathbb{P}_i) = \sum_{p_j \in \mathbb{P}_i} h(p_j)$ that combines the computed features for each point into an order-invariant feature vector for the patch, and
- a *regressor* that estimates the desired properties d_i and $\tilde{o}_i$ from the feature vector of the patch.

Following the original design of PointNet [QSMG17], the feature extractor is implemented with a multi-layer perceptron that is applied to each point separately, but shares weights between points. Computing the features separately for each point ensures that they are invariant to the ordering of the points. The feature extractor also applies an additional spatial transformer network to intermediate point features.

In our implementation, we add skip connections to the multi-layer perceptrons, similar to ResNet blocks. Empirically, we found this to help with gradient propagation and improve training performance. The regressor is also implemented with a multi-layer perceptron. Similar to the feature extractor, we add skip connections to help gradient propagation and improve training performance. We use the same network width as in the original PCPNet (please refer to the original paper for details). However, the network is two times deeper as we replace the original layers with two layers ResBlocks. This architecture is used to compute both outlier indicators and displacement vectors. We change the number of channels of the last regressor layer to fit the size of the desired output (1 for outlier indicators and 3 for displacement vectors).

Importantly, for a each point p_i in the point cloud, we compute its local neighborhood $\mathbb{P}_i$ and only estimate the outlier probability and displacement vector for the center point p_i, i.e., we do not estimate outlier probabilities or displacement vectors for other points in the patch. Thus, each point in the original point cloud is processed independently by considering its own neighborhood and indirectly gets coupled by the iterative cleaning, as described next.

Iterative cleaning. At test time, after applying the displacement vectors computed from a single iteration of the architecture, we are left with residual noise. The residual error vectors from denoised points $\tilde{p}_i$ to the target surface that are introduced by our method do not vary smoothly over the denoised points. Empirically, we found

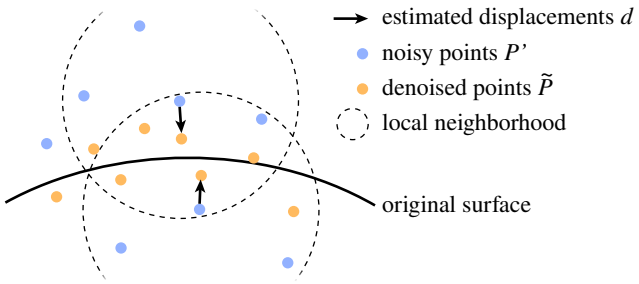

Figure 3: *Residual noise. Different local neighborhoods for adjacent denoised points cause slightly different results, which can be seen as residual noise in the denoised points. Iterating the denoising approach improves the results.*

that this residual noise has a similar type, but a lower magnitude than the original noisy points. Intuitively, this can be explained by looking at the content of input patches for points that are neighbors in the denoised point cloud. As shown in Figure 3, input patches that are far apart have different content, resulting in different network predictions, while patches that are close together have similar content, and similar predictions. The distance of these input patches correlates with the noise model and the noise magnitude, therefore the network predictions, and the denoised points, are likely to have noise of a similar type, but a lower magnitude than the original noisy points. This allows us to iterate our denoising approach to continue improving the denoised points.

In practice, we observed shrinking of the point cloud after several iterations. To counteract this shrinking, we apply an inflation step after each iteration, inspired by Taubin smoothing [Tau95]:

$$d_i' = d_i - 1/k \sum_{p_j \in N(p_i)} d_j, \tag{3}$$

where d_i' are the corrected displacements vectors and $N(p_i)$ are the k nearest neighbours of point p_i, we set $k = 100$. Note that this step approximately removes the low-frequency component from the estimated displacements.

5. Training Setup

To train the denoising model, we use a dataset of paired noisy point clouds and corresponding clean ground truth point clouds. We do not need to know the exact correspondences of points in a pair, but we assume we do know the ground truth outlier label for each noisy point. Using a point cloud as ground truth instead of a surface description makes it easier to obtain training data. For example, a ground truth point cloud can be obtained from a higher-quality scan of the same scene the noisy point cloud was obtained from. Since we work with local patches instead of entire point clouds, we can train with relatively few shapes. To handle different noise magnitudes, and to enable our iterative denoising approach, we train with multiple noise levels. This includes several training examples with zero noise magnitude, which trains the network to preserve the shape of point clouds without noise.

Loss function. Choosing a good loss function is critical, as this choice has direct impact on the properties of the cleaned point clouds. For the outlier removal phase, we use the L_1 distance between the estimated outlier labels and the ground truth outlier labels:

$$L_o(\tilde{p}_i, p_i) = \|\tilde{o}_i - o_i\|_1, \tag{4}$$

where $\tilde{o}_i$ is the estimated outlier probability and o_i is the ground truth label. We also experimented with the binary cross-entropy loss, but found the L_1 loss to perform better, in practice.

In the denoising setting, designing the loss function is less straight-forward. Two properties we would like our denoised point clouds to have are proximity of the points to the scanned surface, and a regular distribution of the points over the surface. Assuming the ground truth point cloud has both of these properties, a straightforward choice for the loss would be the $L2$ distance between the

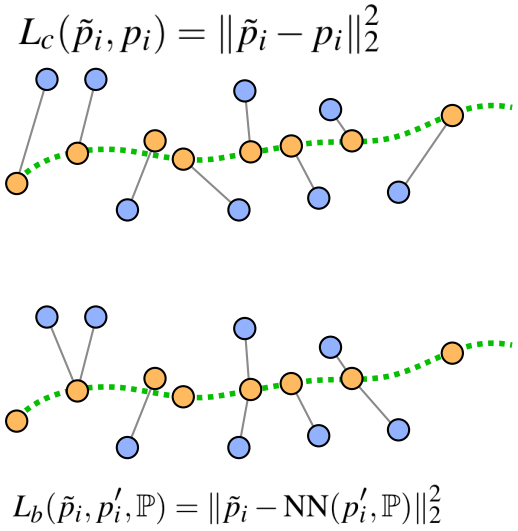

Figure 4: *Alternate loss functions that result in comparatively worse performance (see Figure 12). Dotted green line denotes the underlying scanned surface, orange points denote original points, and blue points denote the noisy points. The error function L_c (top) tries to learn denoising as denoised points $\tilde{p}_i$ going back to the original ground truth points p_i; while, the error function L_b tries to learn denoising as denoised points $\tilde{p}_i$ going to the closest point in the cleaned point set $\mathbb{P}$, i.e., $NN(p_i', \mathbb{P})$.*

cleaned and the ground truth point cloud:

$$L_c(\tilde{p}_i, p_i) = \|\tilde{p}_i - p_i\|_2^2, \tag{5}$$

where $\tilde{p}_i$ and p_i are corresponding cleaned and ground truth points in a patch. Note that, for simplicity of notation, we have unrolled the displacement vector expressions directly in terms of point coordinates. However, this assumes knowledge of a point-wise correspondence between the point clouds; and even if the correspondence is known, we can in general not recover the component of the additive noise that is tangent to the surface. The minimizer of this loss is therefore an average between all potential candidates the noisy point may have originated from. This average will in general not lie on the surface, and lead to poor overall performance. Figure 4, top, illustrates this baseline loss. Fortunately, we do not need to exactly undo the additive noise. There is a large space of possible point clouds that satisfy the desired properties to the same degree as the ground truth point cloud, or even more so.

We propose a main loss function and an alternative with a slightly inferior performance, but simpler and more efficient formulation. The main loss function has one term for each of the two properties we would like to achieve: *Proximity to the surface* can be approximated as the distance of each denoised point to its nearest neighbour in the ground truth point cloud:

$$L_s(\tilde{p}_i, \mathbb{P}_{\tilde{p}_i}) = \min_{p_j \in \mathbb{P}_{\tilde{p}_i}} \|\tilde{p}_i - p_j\|_2^2. \tag{6}$$

For efficiency, we restrict the nearest neighbor search to the local patch $\mathbb{P}_{\tilde{p}_i}$ of ground truth points centered at $\tilde{p}_i$. Originally, we

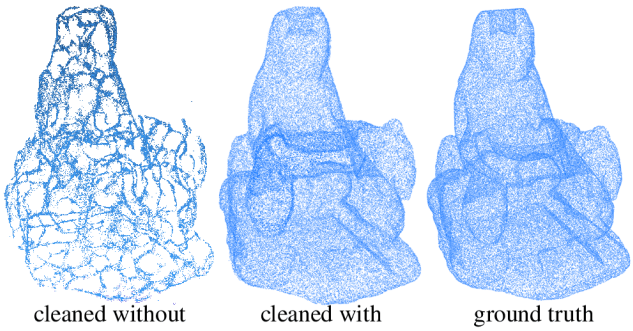

Figure 5: *Omitting the regularity term from the loss causes the points to cluster into filament structures on the surface after multiple iterations (left). Compare to results with the regularity term (center) and the ground truth point cloud (right).*

experimented with only this loss function, but noticed a filament structures forming on the surface after several denoising iterations, as shown in Figure 5. Since the points are only constrained to lie on the surface, there are multiple displacement vectors that bring them equally close to the surface. In multiple iterations, the points drift tangent to the surface, forming clusters. To achieve a more *regular distribution on the surface*, we introduce a regularization term:

$$L_r(\tilde{p}_i, \mathbb{P}_{\tilde{p}_i}) = \max_{p_j \in \mathbb{P}_{\tilde{p}_i}} \|\tilde{p}_i - p_j\|_2^2. \tag{7}$$

By minimizing this term, we minimize the squared distance to the *farthest* point in the local patch $\mathbb{P}_{\tilde{p}_i}$. Intuitively, this keeps the cleaned point centered in the patch and discourages a drift of the point tangent to the surface. Assuming the noisy point clouds are approximately regularly distributed, this results in a regular distribution of the cleaned points since, in this case Eq. 7 promotes the clean point to lie in the barycenter of the points in its patch. With this term, we want to avoid the excessive clustering of points (for example, into filament structures), which is especially important when applying our approach iteratively. The full loss function is a weighted combination of the two loss terms:

$$L_a = \alpha\, L_s + (1-\alpha)\, L_r. \tag{8}$$

Since the second term can be seen as a regularization, we set α to 0.99 in our experiments.

Importantly, the loss defined in Eq. (8) depends on the current point cloud, so that the point searches in Equations (6) and (7) need to be *updated in every training epoch*. Alternatively, these target points can be fixed. Thus, our alternative loss function uses an explicit ground truth for the cleaned point that can be precomputed:

$$L_b(\tilde{p}_i, p_i', \mathbb{P}) = \|\tilde{p}_i - \mathrm{NN}(p_i', \mathbb{P})\|_2^2, \tag{9}$$

where $\mathrm{NN}(p_i', \mathbb{P})$ is the closest point to the initial noisy point p_i' (before denoising) in the ground truth point set $\mathbb{P}$. Figure 4, bottom, illustrates this loss. Since both p_i' and the ground truth point cloud are constant during training, this mapping can be precomputed, making this loss function more efficient and easier to implement. Additionally, the fixed target prevents the points from drifting tangent to the surface. However, this loss constrains the network more than L_a and we observed a slightly lower performance.

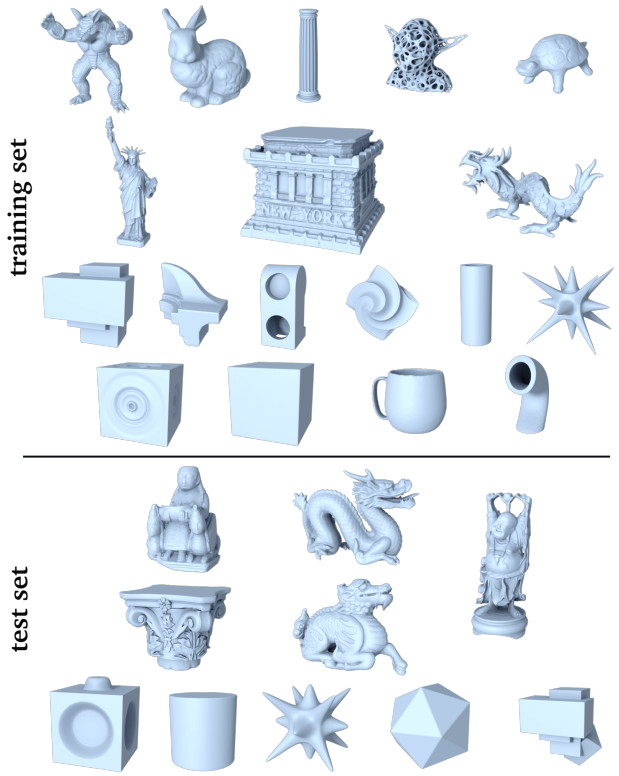

Figure 6: *The shapes used for the* POINTCLEANNET *training and test sets.*

For the outlier removal network we use a learning rate of 10^{-4} and uniform Kaiming initialization [HZRS15] of the network weights. When training the denoising network, we observed that network weights converge to relatively small values. To help convergence, we lower the intial values of the weights to uniform random values in $[-0.001, 0.001]$ and decrease the learning rate to 10^{-8}. This improves convergence speed for the denoising network and lowers the converged error.

5.1. Relation to PCPNet

While being directly based on PCPNet [GKOM18], our approach has several characteristics, specifically adapted to the point cloud denoising and outlier detection problem:

Loss. We use an adapted loss, summarized in Eq. (6) and (7), which, importantly, not only includes a regularization via the distance to the farthest point, but is also updated at every training iteration, through the change of the corresponding points. We have experimented with several alternatives such as the loss described in Eq. (9) and found them to perform consistently worse than ours.

Iterative deep network. Importantly, we apply our network *iteratively* for improved noise reduction. While perhaps non-standard,

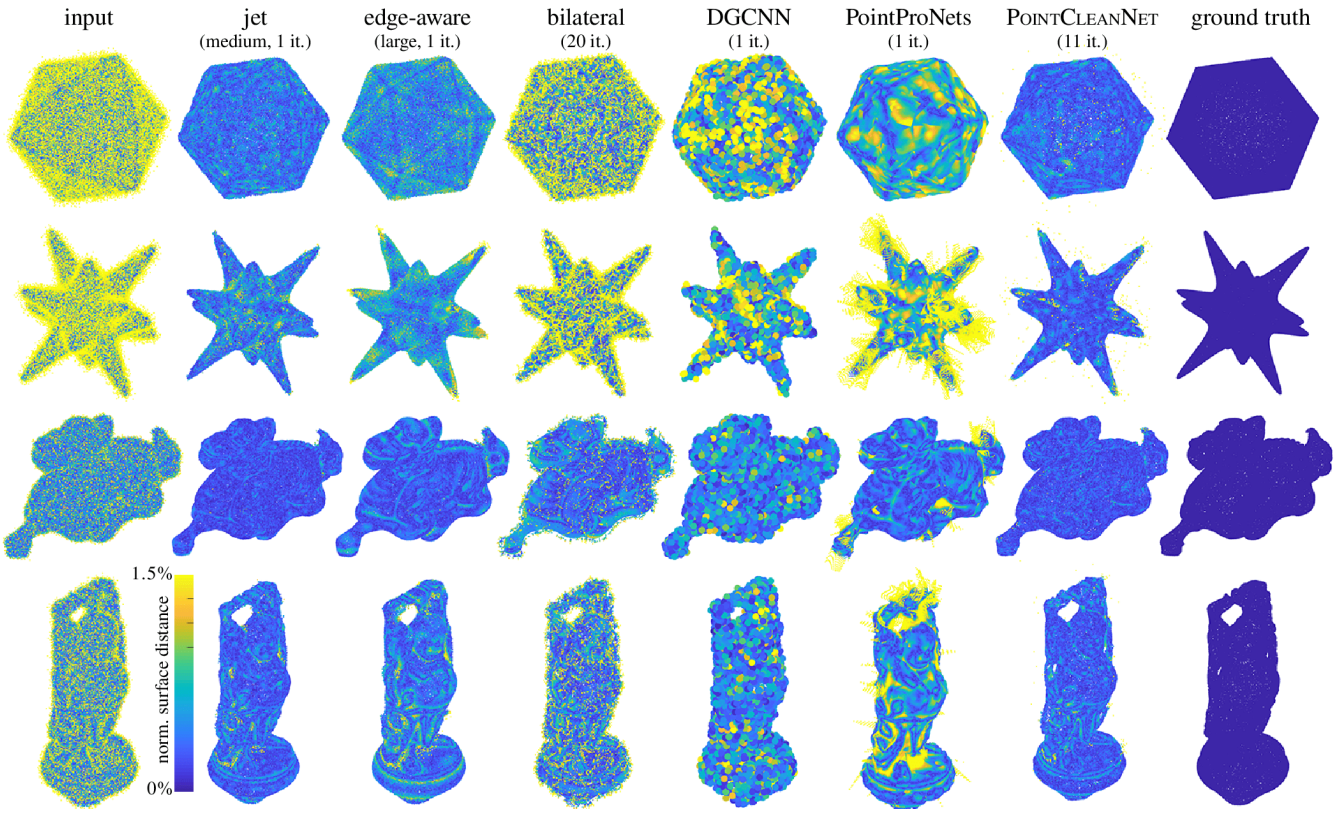

Figure 7: *Qualitative comparison to the state-of-the-art. We compare simple shapes in the top row and increase shape complexity towards the bottom. DGCNN can only handle small point clouds, thus we use a sparser sampling for this method. Colors illustrate the denoising error, we use the distance-to-surface for each denoised point.*

this results in very significant improvement in our setting. Moreover, we found that a straightforward implementation might not converge, while with the proper loss and with an inflation term, the network can both stabilize and achieve higher accuracy.

Practical applicability. Finally, we remark that POINTCLEANNET, as a specialized adaptation of an existing network, both simplifies its integration in practice and establishes its applicability for point cloud denoising and outlier removal.

6. Results

We first describe our dataset and evaluation metric in Sections 6.1 and 6.2. Based on this dataset and metric, we compare the denoising performance (Sec. 6.3) and the outlier detection performance (Sec. 6.4) of our method to several baselines and state-of-the-art methods, including the recent learning-based approaches PointProNets [RÖPG18] and an adapted version of Dynamic Graph CNNs [WSL*18] among others. Experiments on additional datasets with different noise distributions, including simulations of non-uniform scanner noise, and noise from real world scans are presented in Section 6.5. We evaluate our method under different forms of point cloud artefacts classified in a recent survey [BTS*17]. In particular, we treat noisy data and outliers in Sec. 6.3 and 6.4, and consider misaligned scans in 6.5.3. Although we do not perform an extensive evaluation, we also remark that our method is, to some extent, robust to non-uniform sampling due to the regularization term from Eq. (7) that promotes regularity in the denoised point cloud.

6.1. Datasets

Our main dataset contains 28 different shapes, which we split into 18 training shapes and 10 test shapes. See Figure 6 for a gallery of all shapes. From the original triangle meshes of each shape, we sample 100K points, uniformly at random on the surface, to generate a clean point cloud.

For the *denoising task*, noisy point clouds are generated by adding Gaussian noise with the standard deviation of 0.25%, 0.5%, 1%, 1.5% and 2.5% of the original shape's bounding box diagonal. In total, the denoising training set contains 108 shape variations, arising from 6 levels of noise (including the clean points) for each of the 18 shapes.

For the *outlier removal task*, we use the same training and test shapes, however we use only clean point clouds and with a larger sample count of 140k points per shape. To generate outliers, we added Gaussian noise with standard deviation of 20% of the shape's bounding box diagonal to a random subset of points. The training set contains point clouds with proportions starting at 10% until 90%

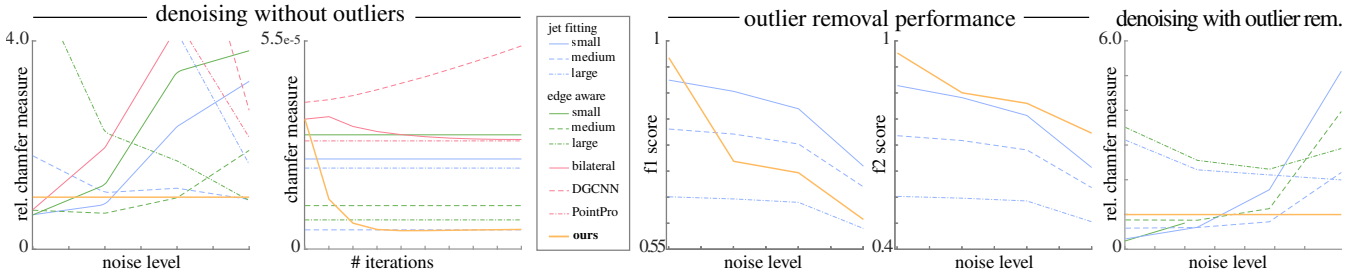

Figure 8: *Quantitative comparison. We compare the performance of our model to jet smoothing [CP05], edge-aware denoising [HWG*13], the bilateral point cloud filter by Digne et al. [DDF17], Dynamic Graph CNNs [WSL*18], and PointProNets [RÖPG18]. The two plots on the left are evaluated on our test set without outliers, the two following plots compare the outlier removal performance using the f1 and the f2 scores, and the right-most plot shows the denoising performance after outlier removal.*

in intervals of 10% of the points converted to outliers. Only the outliers that are farther from the surface than the standard deviation of the noise distribution are selected.

In total, the outlier removal training set contains 432 example shapes, arising from 6 outlier densities and 4 levels of noise for each of the 18 training shapes.

The test set contains point clouds with 30% of outliers points. To test the generality of our outlier removal, we added a second method to generate outliers to our test set only. In this setting, outliers are distributed uniformly inside the shape's bounding box that has been scaled up by 10%.

In Section 6.5 we also evaluate our method on point clouds generated with alternative methods, including simulated non-uniform noise and noise from real real acquisition devices.

POINTCLEANNET training datasets for denoising and outlier removal are available on our project page.

6.2. Evaluation Metric

The evaluation metric should be sensitive to the desired properties of the point cloud described earlier: point clouds should be close to the surface and have an approximately regular distribution. If we assume the ground truth point clouds have a regular distribution, the following *Chamfer measure* [FSG17, ADMG18], a variant of the Chamfer distance [BTBW77], measures both of these properties:

$$c(\tilde{\mathbb{P}},\mathbb{P}) = \frac{1}{N}\sum_{p_i\in\tilde{\mathbb{P}}}\min_{p_j\in\mathbb{P}}\|p_i-p_j\|_2^2 + \frac{1}{M}\sum_{p_j\in\mathbb{P}}\min_{p_i\in\tilde{\mathbb{P}}}\|p_j-p_i\|_2^2.$$

Here N and M are the cardinalities of the cleaned $\tilde{\mathbb{P}}$ and ground truth $\mathbb{P}$ point clouds, respectively. Note that the first term measures an approximate distance from each cleaned point to the target surface, while the second term intuitively rewards an even coverage of the target surface and penalizes gaps. All our point clouds are scale-normalized to have a unit bounding box diagonal, making the point distances comparable for different shapes.

For a dataset with simulated scanner noise that we will describe in Section 6.5, the clean point set has a non-uniform point distribution. For this dataset we use only the root mean square distance-to-surface (*RMSD*) of each point as evaluation metric:

$$d(\tilde{\mathbb{P}},\mathbb{P}) = \sqrt{\frac{1}{N}\sum_{p_i\in\tilde{\mathbb{P}}}\min_{p_j\in\mathbb{P}}\|p_i-p_j\|_2^2}.$$

6.3. Evaluating Denoising

We first evaluate the denoising task alone, without outlier removal. We compare the results of our method on different noise levels to several state-of-the-art techniques for point cloud denoising.

We first consider a qualitative evaluation of our results in Figure 7, showing the denoised point clouds for four different input noisy point clouds (Icosahedron, Star smooth, Netsuke, Happy) with two different noise intensities, 1% and 2.5% of the original shape bounding box diagonal. The distances from each of the denoised points to the ground truth surface are color-coded. In the same figure, we can also compare the performance of our method to other successful algorithms.

We compare against five other methods, as described next. It is important to note that in most of these methods, it is necessary to tune some parameters, such as the neighborhood size, to adjust to the different noise levels, while our method works across all noise levels with the same hyper-parameters. When applicable, we manually adjusted parameters for best performance. Also, in some algorithms, we allowed multiple parameter settings (small, medium, large) to handle different levels of noise.

(i) *Polynomial fitting with osculating jets* [CP05, CP07]: Osculating jet-fitting performs well if the right neighborhood size is chosen for the given noise level. Otherwise, neighborhood sizes that are too small overfit to strong noise (Figure 7, first row), and neighborhood sizes that are too large do not preserve detailed features (Figure 7, second and fourth row).

(ii) *Edge-aware point set resampling* [HWG*13]: Edge-aware point set resampling has larger errors near detailed features (Figure 7, third and fourth row), while obtaining good results near sharp edges, like the edges of the icosahedron.

(iii) *Bilateral filtering for point clouds* [DDF17]: bilateral filtering performs poorly in strong noise settings (Figure 7, first two rows).

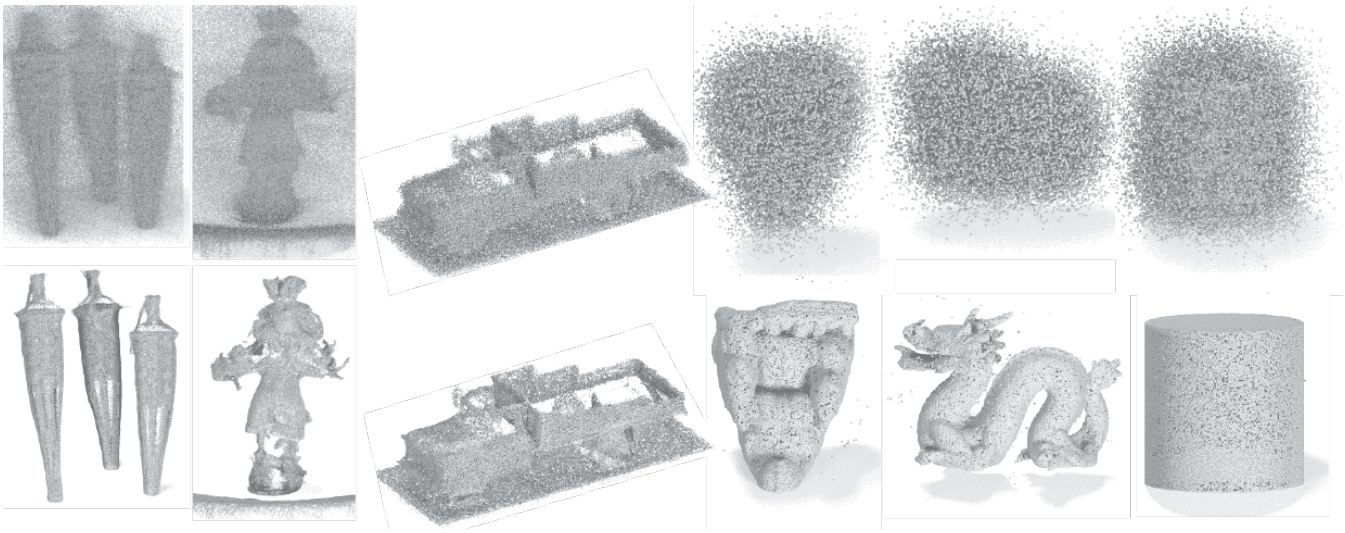

Figure 9: *Example input outlier and noise corrupted pointclouds (top) and their corresponding cleaned output (bottom) produced by* POINTCLEANNET. *From left to right: torch, scarecrow, tanks-and-temple, galera, dragon, cylinder. The left two examples are corrupted with real-world scanning noise and outliers, the other examples with synthetic noise and outliers.*

(iv) *Dynamic Graph CNN (DGCNN)* [WSL*18]: Note that Dynamic Graph CNNs were not designed for local operations, such as denoising. We modify the segmentation variant of this method to output a displacement vector per point instead of class probabilities. For the loss, the displacements are added to the original points and the result is compared to the target point cloud using the same Chamfer measure used as the error metric in our evaluation.
Since the whole point cloud is processed in a single go, we need to heavily sub-sample our dense point clouds before using them as input for DGCNN. We also restrict DGCNN to a single iteration as we found the result set to diverge over iterations. Similar to bilateral filtering, DGCNN also performs poorly in strong noise settings (Figure 7, first two rows).

(v) *PointProNets* [RÖPG18]: PointProNets requires oriented normals during training. Where available, these are obtained from the ground truth source meshes, or estimated with PCPNet [GKOM18] otherwise. Differently from the original method, we also use *ground truth normals* to orient the predicted height maps, instead of trying to estimate an orientation, as we found the in-network estimation used in the original method to be unstable for our datasets. Note that this provides an upper bound for the performance of PointProNets. The denoised patches do not accurately reconstruct detailed surfaces, presumably due to the smoothing effect of the image convolutions, and suffer from artefacts caused by the smoothing of the height map at the boundaries of a patch.

In contrast, our method works on local patches directly in the point domain, can apply several iterations of denoising to improve results, and is robust to a large range of noise levels with the same choice of hyper-parameters. This results in lower residual error, especially in detailed surface regions.

We also present quantitative comparisons that summarize the performance of each method on the entire dataset. The previously described Chamfer measure is used as evaluation metric that captures both the distance from denoised points to the ground truth surface and the regularity of the points.

Results are shown in Figure 8 left (without outlier removal). We can observe that POINTCLEANNET performs noticeably better under mid to high noise level and using multiple iterations compared to all the other methods. The performance of our method is also more stable to changes in noise levels, while most other methods perform well only for a specific level of noise.

Comparison to the alternative loss. As shown in Figure 12, our alternative loss performs slightly worse than our main loss. However, it is more efficient and easier to implement, so the choice of loss function depends on the setting.

6.4. Evaluating Outlier Removal

Figure 8 shows the performance of our outlier removal method (right). For the purpose of cleaning dense data, a model should prioritize classifying outlier points correctly (true positives) over limiting the number of false positives. Therefore we consider that recall has more importance than precision for this task. The F_β score conveys the balance between recall and precision. Plots 3 and 4 compare our method to jet-fitting [CP05] using F_1 and F_2 scores. We observe that when recall and precision are weighted equally, our method has the best performance when removing outliers on clean point clouds while remaining effective on the other noise levels. POINTCLEANNET performs the best for all noise levels when using F_2 score which gives larger weight to recall.

The last plot in Figure 8 compares our approach to jet-fitting and edge-aware filtering [HWG*13] with both outlier removal and denoising on the test set. In this experiment, we first removed outliers

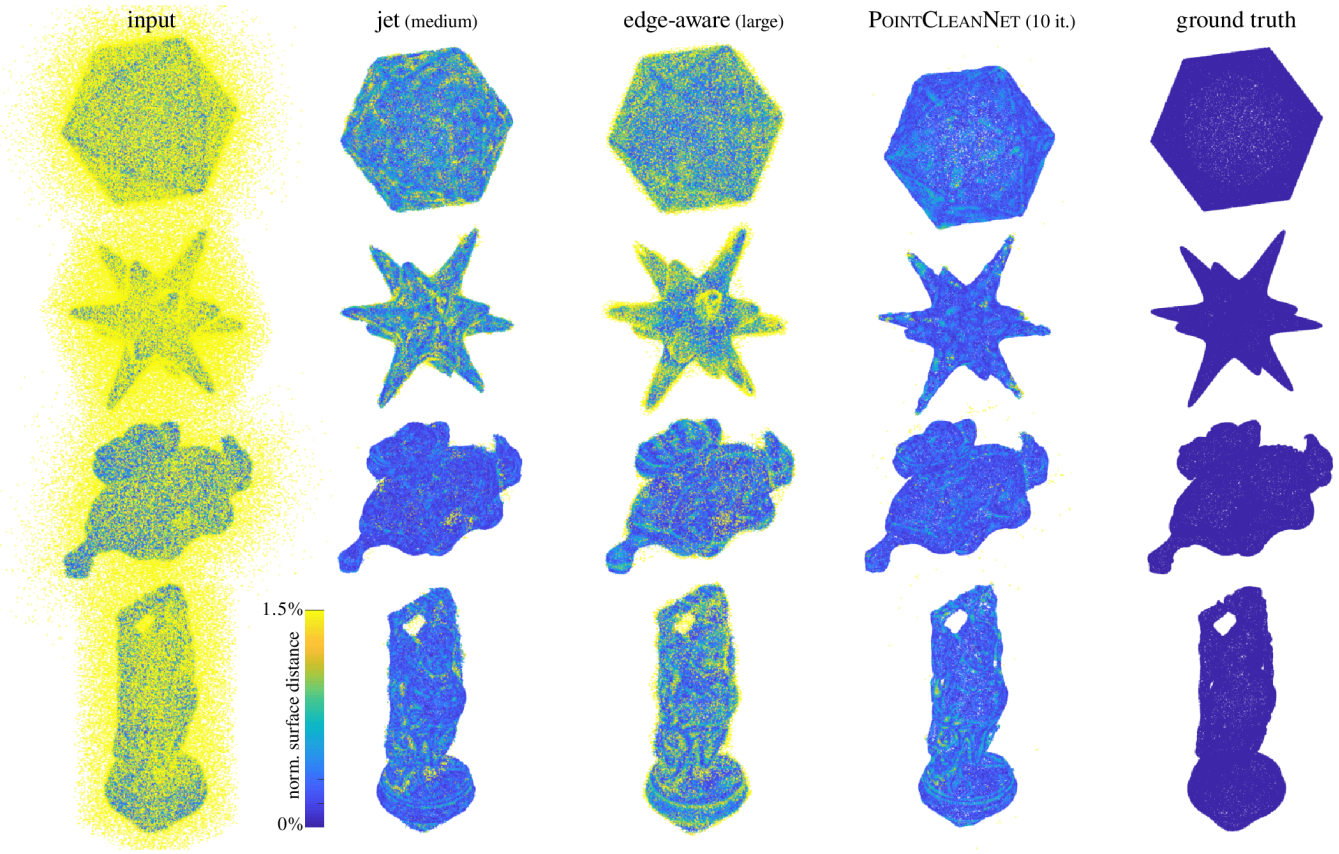

Figure 10: *Qualitative comparison on our outlier test set. Here the task is to remove points that are not part of the original surface. Note that that analytic methods with a large setting for the radius (third column) fail to remove outliers hidden inside small details, such as the arms of the statue, while a smaller setting (second column) results in a lot of residual noise. Since* POINTCLEANNET *can learn to adapt to the feature to produce a result with less noise.*

using an outlier classification technique and then denoised the point clouds from our test set. We show the results for different noise levels from zero to 2.5% of the shape bounding box diagonal. Finally, we make two observations: first, POINTCLEANNET outperforms edge-aware and jet-fitting techniques with outlier removal and denoising on medium to large noise levels; and second, on smaller noise levels, our model still outperforms a few of the different tuning variations of the related techniques. Recall that our model does not require parameter tuning by the user.

Figure 10 also shows qualitative results for outlier removal on our test set compared to the related techniques mentioned before. We observe that edge-aware filtering performs worse around highly detailed regions and edges, while jet-fitting does not manage to clean the remaining outliers at scattered points. The result highlights the consistent performance of POINTCLEANNET across different shapes and varying level of details, contrary to the other methods which produce less consistent distances to the underlying ground truth shapes.

6.5. Performance under Different Noise Types

6.5.1. Directional noise

We evaluated POINTCLEANNET on a synthetic dataset simulating 3D data acquisition via depth cameras. To do so we created a dataset with structured noise levels to simulate depth uncertainty of depth reconstructions. Specifically, we added noise using an anisotropic Gaussian distribution with constant covariance matrix aligned along the scanning direction. The results are shown in Figure 11. Note that our network was *not* re-trained for this specific model.

In this setting, the non-data-driven methods, such as jet-fitting and edge-aware filtering, perform well since they are not specialized to any noise model. Even though our method was never trained on this type of noise, it still performs on par with the best methods on low to medium noise settings, and is only outperformed on high noise settings by non-data-driven methods with parameters tuned for the given noise strength.

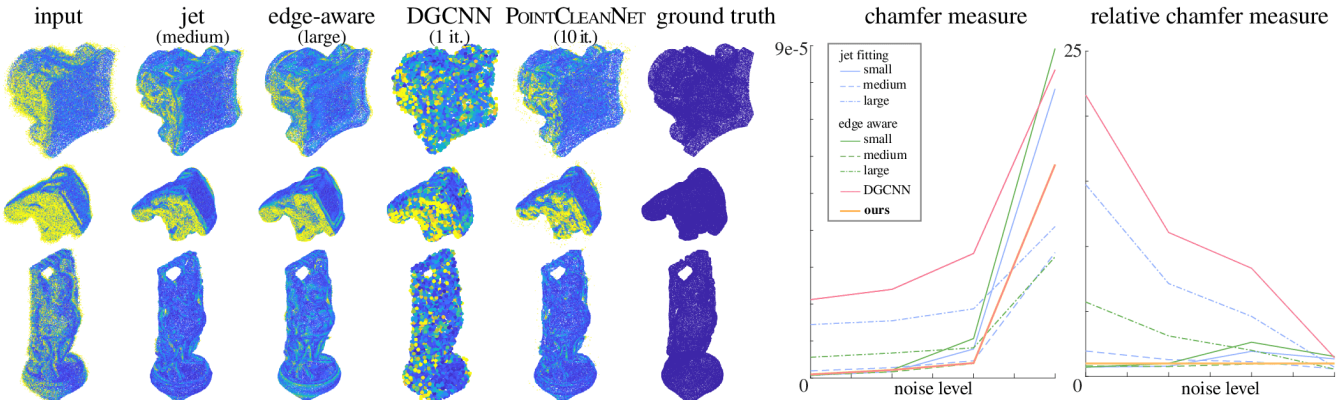

Figure 11: *Qualitative comparison on the directional noise test set. We show that* POINTCLEANNET *can adapt to different kinds of noise models. Here we added anisotropic noise to the shapes. Qualitative results are o the left, and quantitative plots of the absolute and relative Chamfer measure over different noise levels on the right. Even though our method was not trained on isotropic noise only, it still performs on par with the state of the art on low and medium noise levels.*

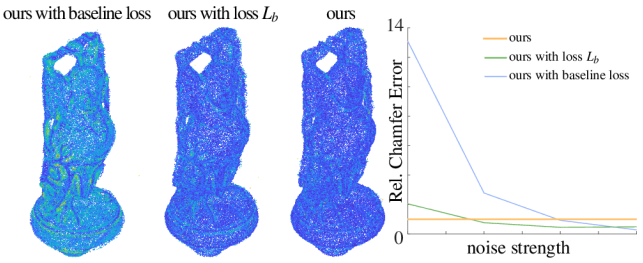

Figure 12: *Comparison of our two-term loss L_a with the alternative loss L_b and the baseline loss L_c. Our loss results gives large benefits over the baseline loss, and performs somewhat better than the alternative loss as well, due to being less constrained.*

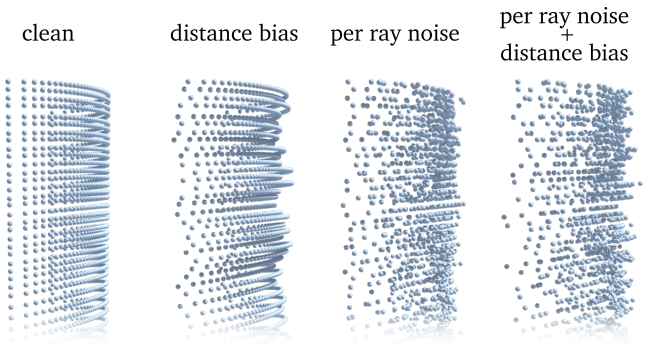

Figure 13: *Simulated noise of a Velodyne HDL-64E scanner. Here we show an example of the two noise types introduced by this scanner: distance bias (db) adds an error to the depth of scan lines, while ray noise (rn) adds an error to the depth of each point. Note that the points in the ground truth (left) are not distributed uniformly, so we use the distance of each point to the surface as error measure for experiments with this dataset.*

6.5.2. Structured noise

We evaluated our method on a simulated LIDAR dataset (*Velodyne*) generated using BlenSor [GKUP11], which models various types of range scanners. We chose to simulate a rotating LIDAR, in particular a Velodyne HDL-64E scanner. BlenSor implements two types of sensor specific error for this scanner: first, a distance bias for each laser unit; and second, a per-ray Gaussian noise. The different effects of the noise types can be observed in Figure 13. We use the shapes in POINTCLEANNET dataset for this experiment. After being normalized, each shape is scanned from $\theta \in [0^\circ, 180^\circ]$ with distance bias with standard deviation 0%, 0.5% and 1% and a per-ray noise of standard deviation 0%, 0.5% and 1%.

Quantitative numbers are presented in Table 1. Here, we use the distance to the surface instead of the Chamfer measure, since this type of scanner naturally produces non-uniform point clouds (even in the ground truth). In bold, we highlight the two best performing methods. We evaluate two versions of our method: one version trained on the unstructured noise described in Section 6.1, and one version re-trained on the *Velodyne* dataset. The re-trained version significantly outperforms all other methods, while the non-retrained version still performs competitively. In Figure 17, we show a qualitative evaluation of our results compared to the two other best performing methods. We show the denoised scans for two shapes: cylinder and dragon, with distance bias (first two rows) and with distance bias and per-ray noise (last two rows). Both jet and bilateral methods preserve an amount of structured noise (see the cylinder shape). Jet medium produces artefact points in areas of high details especially on the dragon shape.

Table 2 and Figure 14 evaluate our model on the *Kinect v1* dataset introduced in [WLT16]. We trained the model on the scans of the shapes David, big-girl and pyramid from this dataset, and tested on a subset of the scans from boy, girl and cone with a radius of 2.5% of the shapes' bounding box diagonals. Our trained network achieves the best performance. In Figure 14, we show the Poisson reconstructions [KH13] and distances from the ground

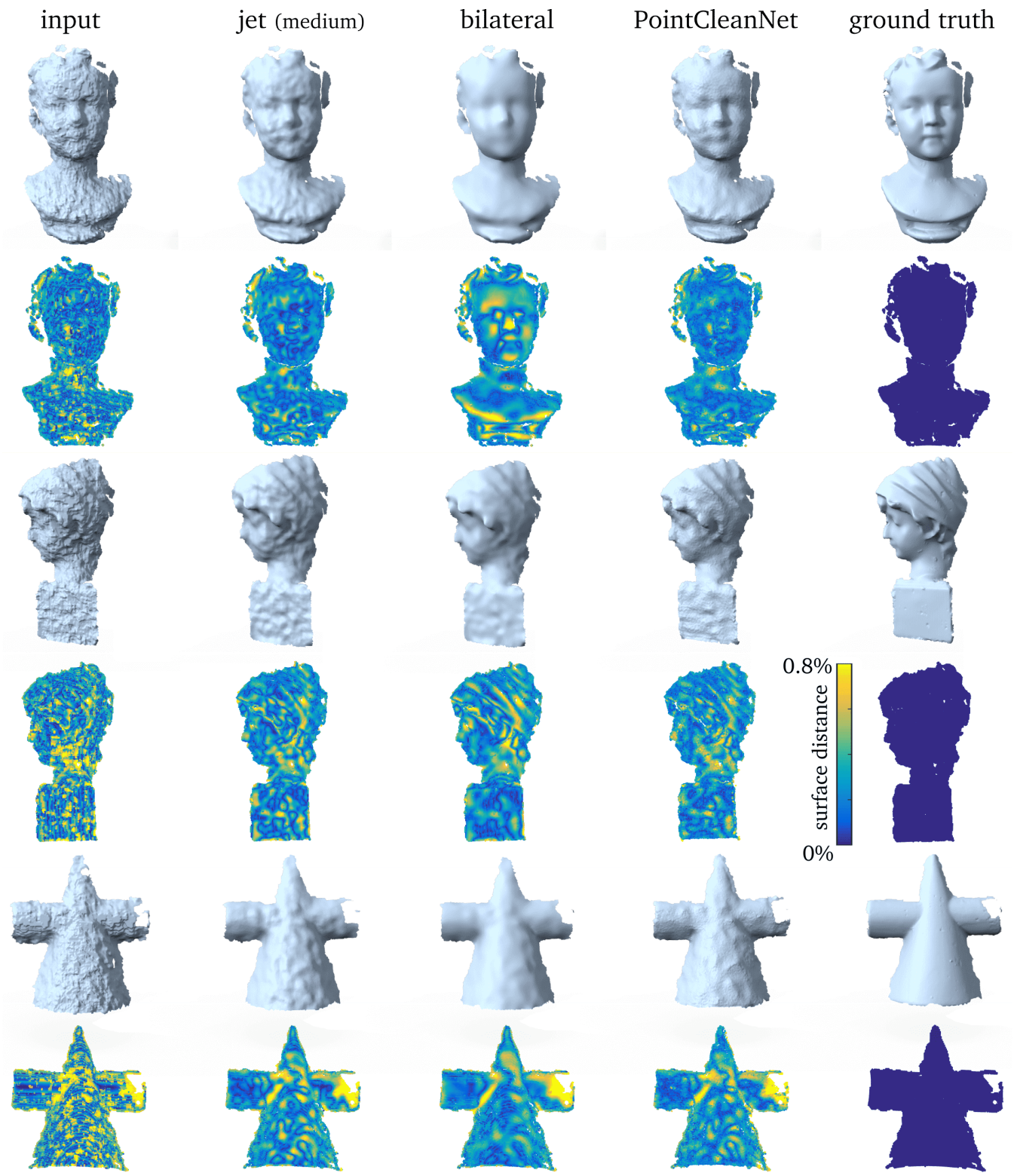

Figure 14: *Qualitative comparison of the three best performing methods on the Kinect v1 dataset. For each shape, we compare a Poisson reconstruction and the normalized distance to the ground truth surface of the denoised point sets computed by each method.*

Table 1: *Comparison with state-of-the-art methods on the Velodyne datasets. We show the root mean square distance-to-surface (RMSD) for each method. The first column evaluates the methods on a dataset with only distance bias as noise and the second column with added per-ray noise (see Figure 13 for examples of the noise types). PointProNets was re-trained on the dataset, and for our method we show both a re-trained version, and a version trained on the original dataset (Section 6.1).*

	Velodyne (db)	*Velodyne* (db+rn)
jet small	5.46	5.78
jet medium	4.91	5.18
jet large	9.68	9.67
edge-aware small	5.50	6.36
edge-aware med.	5.48	5.77
edge-aware large	11.31	11.53
bilateral	**4.53**	**4.99**
PointProNets	17.47	22.02
ours	5.83	7.03
ours re-trained	**4.07**	**4.27**

truth for the three best performing methods: jet medium, bilateral and ours. The normals where computed using the normal estimation tool from Meshlab. Note that jet-fitting method tends to preserve the structured noise while the bilateral method tends to oversmooth shape. We also re-trained POINTCLEANNET on the *Kinect v2* dataset described in [WLT16]. As shown in Table 2 our method performs well compared to other methods.

6.5.3. Misaligned data

We evaluated our method on a dataset containing misaligned scans. For this, we generate scans of each mesh from the POINTCLEANNET dataset by simulating a rotating Velodyne HDL-64E scanner using BlenSor and scan each shape from 6 equally spaced angles between 0° and 360° without noise. To simulate the artefacts of misaligned scans, we apply a rotation centered at the scanner position and of angle uniformly sampled in $[-\theta, \theta]$ to each scan. Our misaligned scans dataset was built with θ being valued at 1° and 2°. Figure 15 shows a cross section of two shapes from the misaligned scans dataset.

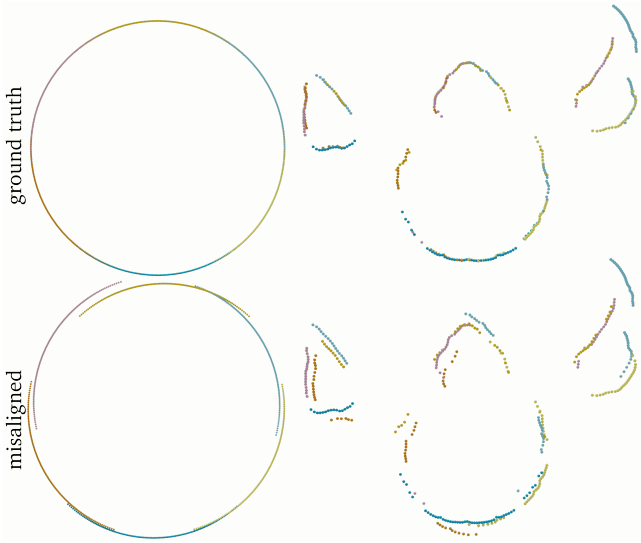

Figure 15: *Cross section of cylinder (left) and armadillo (right) misaligned scans with $\theta = 2^\circ$. Each color corresponds to a scan from a single angle.*

We present a quantitative evaluation in Table 3. Similar to the *Velodyne* dataset, we only evaluate the distance to the surface, since the produced scans are not uniform. We evaluate a model that we re-trained on this dataset, as well as the model that was only trained on the basic POINTCLEANNET dataset. We compare both to state-of-the-art methods. Note that our re-trained model has the best performance. In Figure 16 we also show a qualitative comparison of our method with the bilateral filtering method [DDF17]. We observe that POINTCLEANNET is able to merge different scans while bilateral filtering keeps some artefacts from the misaligned scans (first row) and even introduces noise (second row).

Table 2: *Comparison with state-of-the-art methods on the Kinect v1 and Kinect v2 datasets. We show the chamfer measure for each method. PointProNets was re-trained on each dataset, and for our method we show both a re-trained version, and a version trained on the original dataset (Section 6.1).*

	Kinect v1	*Kinect v2*
jet small	5.10	6.36
jet medium	**4.69**	**6.16**
jet large	5.40	8.63
edge-aware small	5.21	6.38
edge-aware med.	4.78	6.53
edge-aware large	6.85	13.10
bilateral	4.72	**6.04**
PointProNets	7.39	12.81
ours	5.02	6.42
ours re-trained	**4.57**	6.26

Table 3: *Comparison with state-of-the-art methods on the misaligned scans dataset. We report the root mean square distance-to-surface (RMSD). We consider both a re-trained version and a version trained on the original dataset of* POINTCLEANNET .

	misaligned scans
jet small	5.10
jet medium	**4.74**
jet large	7.52
edge-aware small	5.43
edge-aware med.	4.85
edge-aware large	6.62
bilateral	5.08
ours	5.23
ours re-trained	**4.35**

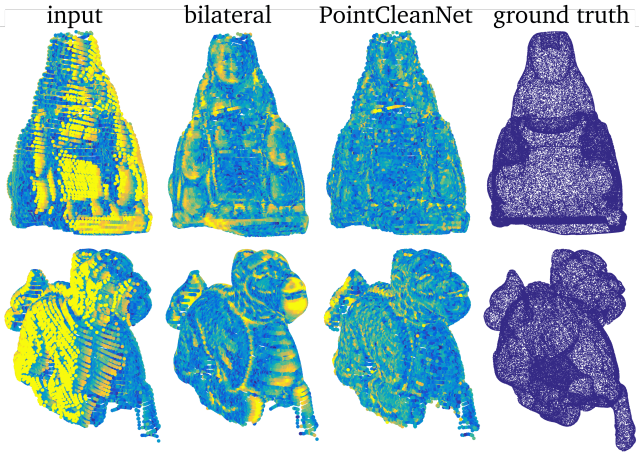

Figure 16: *Qualitative comparison with bilateral filtering [DDF17] on the misaligned scans dataset. We show the normalized distance to the ground truth surface.*

6.5.4. Generalization to real-world data.

Figure 1 and the first two results in Figure 9 (left) show the result of our approach on real data obtained with the plane swift algorithm [WKZ*16], an image-based 3D reconstruction technique. Statue, torch, and scarecrow input point clouds each contain 1.4M points. Since in this case no ground truth is available, we only show the qualitative results obtained using our method. Note that although trained on an entirely different dataset, POINTCLEANNET still produces high quality results on this challenging real-world data. The next three shapes in Figure 9 show results on other external raw point clouds, while the last one shows a shape with sharp edges.

7. Conclusion, Limitations and Future Work

We presented POINTCLEANNET, a learning-based framework that consumes noisy point clouds and outputs clean ones by removing outliers and denoising the remaining points with a displacement back to the underlying (unknown) scanned surface. One key advantage of the proposed setup is the simplicity of using the framework at test time as it neither requires additional parameters nor noise/device specifications from the user. In our extensive evaluation, we demonstrated that POINTCLEANNET consistently outperforms state-of-the-art denoising approaches (that were provided with manually tuned parameters) on a range of models under various noise settings. Given its universality and ease of use, POINTCLEANNET can be readily integrated with any geometry processing workflow that consumes raw point clouds. Note that in our current framework, we still need paired noisy-clean data to train POINTCLEANNET. An exciting future direction would be learn denoising directly from unpaired data. As a supervised learning method, our approach is also unlikely to succeed when noise characteristics during training are very different from the ones of the test data.

While we presented a first learning architecture to clean raw point clouds directly, several future directions remain to be explored: (i) First, as a simple extension, we would like to combine the outlier removal and denoising into a single network, rather than two separate parts. (ii) Further, to increase efficiency, we would like to investigate how to perform denoising at a patch-level rather than per-point level. This would require designing a scheme to combine denoising results from overlapping patches. (iii) Although POINTCLEANNET already produces a uniform point distribution on the underlying surface if the noisy points are uniformly distributed, we would like to investigate the effect of a specific uniformity term in the loss function (similar to [YHCOZ18]) to also produce a uniform distribution for non-uniform noisy points. The challenge, however, would be to restrain the points to remain on the surface and not deviate off the underlying surface. (iv) Additionally, it would be interesting to investigate how to allow the network to upsample points, especially in regions with insufficient number of points, or to combine it with existing upsampling methods such as [YLF*18] This would be akin to the 'point spray' function in more traditional point cloud processing toolboxes. (v) Finally, we would like to investigate how to train a point cloud cleanup network *without* requiring paired noisy-clean point clouds in the training set. If successful, this will enable directly handling noisy point clouds from arbitrary scanning setups without requiring explicit noise model or examples of denoised point clouds at training time. We plan to draw inspiration from related *unpaired* image-translation tasks where generative adversarial setups that have been successfully used.

8. Acknowledgements

Parts of this work were supported by the KAUST OSR Award No. CRG-2017-3426, a gift from the NVIDIA Corporation, the ERC Starting Grants EXPROTEA (StG-2017-758800) and SmartGeometry (StG-2013-335373), a Google Faculty Award, and gifts from Adobe.

References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (2003), 3–15. 2

[ADMG18] ACHLIOPTAS P., DIAMANTI O., MITLIAGKAS I., GUIBAS L.: Learning representations and generative models for 3d point clouds. In *ICLR* (2018). 9

[Agg15] AGGARWAL C. C.: Outlier analysis. In *Data mining* (2015), Springer, pp. 237–263. 1, 2

[ASGCO10] AVRON H., SHARF A., GREIF C., COHEN-OR D.: l_1-sparse reconstruction of sharp point set surfaces. *ACM TOG 29*, 5 (2010), 135. 3

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric deep learning: going beyond euclidean data. *IEEE SPM 34*, 4 (2017), 18–42. 1

[BBZ*18] BENOUINI R., BATIOUA I., ZENKOUAR K., NAJAH S., QJIDAA H.: Efficient 3d object classification by using direct krawtchouk moment invariants. *MTA* (2018), 1–26. 3

[BCM05] BUADES A., COLL B., MOREL J.-M.: A non-local algorithm for image denoising. In *CVPR* (2005), vol. 2, IEEE, pp. 60–65. 3

[BG05] BEN-GAL I.: Outlier detection. In *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 131–146. 2

[BTBW77] BARROW H. G., TENENBAUM J. M., BOLLES R. C., WOLF H. C.: Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI* (1977), pp. 659–663. 9

[BTS*17] BERGER M., TAGLIASACCHI A., SEVERSKY L. M., ALLIEZ P., GUENNEBAUD G., LEVINE J. A., SHARF A., SILVA C. T.: A survey of surface reconstruction from point clouds. In *CGF* (2017), vol. 36, pp. 301–329. 8

[CCC*10] CHAMBOLLE A., CASELLES V., CREMERS D., NOVAGA M., POCK T.: An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery 9*, 263-340 (2010), 227. 3

[CCSM11] CHAZAL F., COHEN-STEINER D., MÉRIGOT Q.: Geometric inference for probability measures. *FOCS 11*, 6 (2011), 733–751. 2

[CM10] CHATTERJEE P., MILANFAR P.: Is denoising dead? *IEEE TIP 19*, 4 (2010), 895–911. 3

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *CAGD 22:2* (2005), 121–146. 1, 2, 9, 10

[CP07] CAZALS F., POUGET M.: *Jet_fitting_3: A generic C++ package for estimating the differential properties on sampled surfaces via polynomial fitting*. PhD thesis, INRIA, 2007. 1, 2, 9

[DCV14] DIGNE J., CHAINE R., VALETTE S.: Self-similarity for accurate compression of point sampled surfaces. In *CGF* (2014), vol. 33, pp. 155–164. 3

[DDF17] DIGNE J., DE FRANCHIS C.: The bilateral filter for point clouds. *Image Processing On Line 7* (2017), 278–287. 2, 9, 14, 15

[DG10] DESCHAUD J.-E., GOULETTE F.: Point cloud non local denoising using local surface descriptor similarity. *IAPRS 38*, 3A (2010), 109–114. 3

[Dig12] DIGNE J.: Similarity based filtering of point clouds. In *CVPR Workshops* (2012), IEEE, pp. 73–79. 1, 3

[DVC18] DIGNE J., VALETTE S., CHAINE R.: Sparse geometric representation through local shape probing. *IEEE TVCG 24*, 7 (2018), 2238–2250. 3

[EA06] ELAD M., AHARON M.: Image denoising via sparse and redundant representations over learned dictionaries. *IEEE TIP 15*, 12 (Dec 2006), 3736–3745. 3

[Ela10] ELAD M.: From exact to approximate solutions. In *Sparse and Redundant Representations*. Springer, 2010, pp. 79–109. 3

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. *ACM TOG 24*, 3 (2005), 544–552. 1, 2

[FSG17] FAN H., SU H., GUIBAS L. J.: A point set generation network for 3d object reconstruction from a single image. In *CVPR* (2017), vol. 2, p. 6. 9

[GCSA13] GIRAUDOT S., COHEN-STEINER D., ALLIEZ P.: Noise-adaptive shape reconstruction from raw point sets. In *SGP* (2013), Eurographics Association, pp. 229–238. 2

[GKOM18] GUERRERO P., KLEIMAN Y., OVSJANIKOV M., MITRA N. J.: PCPNet: Learning local shape properties from raw point clouds. *CGF 37*, 2 (2018), 75–85. 2, 3, 4, 5, 7, 10

[GKUP11] GSCHWANDTNER M., KWITT R., UHL A., PREE W.: Blensor: blender sensor simulation toolbox. In *ISVC* (2011), Springer, pp. 199–208. 12

[GMM13] GUIBAS L., MOROZOV D., MÉRIGOT Q.: Witnessed k-distance. *Discrete & Computational Geometry 49*, 1 (2013), 22–45. 2

[GP11] GROSS M., PFISTER H.: *Point-based graphics*. Elsevier, 2011. 2

[HJW*17] HAN X.-F., JIN J. S., WANG M.-J., JIANG W., GAO L., XIAO L.: A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication 57* (2017), 103–112. 1, 2, 3

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H. R.: Edge-aware point set resampling. *ACM TOG 32*, 1 (2013), 9. 1, 2, 9, 10

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV* (2015), pp. 1026–1034. 7

[JMFU17] JIN K. H., MCCANN M. T., FROUSTEY E., UNSER M.: Deep convolutional neural network for inverse problems in imaging. *IEEE TIP 26*, 9 (2017), 4509–4522. 3

[JSZ*15] JADERBERG M., SIMONYAN K., ZISSERMAN A., ET AL.: Spatial transformer networks. In *NIPS* (2015), pp. 2017–2025. 4, 5

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM TOG 32*, 3 (July 2013), 29:1–29:13. 12

[LN11] LEVIN A., NADLER B.: Natural image denoising: Optimality and inherent bounds. In *CVPR, 2011* (2011), IEEE, pp. 2833–2840. 3

[LOM*18] LESCOAT T., OVSJANIKOV M., MEMARI P., THIERY J.-M., BOUBEKEUR T.: A survey on data-driven dictionary-based methods for 3d modeling. In *CGF* (2018), vol. 37, pp. 577–601. 3

[Mai10] MAIRAL J.: *Sparse coding for machine learning, image processing and computer vision*. PhD thesis, ENS Cachan, 2010. 3

[MBBV15] MASCI J., BOSCAINI D., BRONSTEIN M., VANDERGHEYNST P.: Geodesic convolutional neural networks on riemannian manifolds. In *ICCV workshops* (2015), pp. 37–45. 1

[MC17] MATTEI E., CASTRODAD A.: Point cloud denoising via moving rpca. In *CGFm* (2017), vol. 36, pp. 123–137. 3

[MVdF03] MEDEROS B., VELHO L., DE FIGUEIREDO L. H.: Point cloud denoising. In *SIAM Conference on Geometric Design and Computing* (2003), Citeseer, pp. 1–11. 2

[ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. In *CGF* (2009), vol. 28, pp. 493–501. 2

[Pin95] PINCUS R.: Barnett, v., and lewis t.: Outliers in statistical data. *Biometrical Journal 37*, 2 (1995), 256–256. 2

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR 1*, 2 (2017), 4. 1, 3, 4, 5

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS* (2017), pp. 5099–5108. 3

[RH11] ROUSSEEUW P. J., HUBERT M.: Robust statistics for outlier detection. *WIREs: Data Mining and Knowledge Discovery 1*, 1 (2011), 73–79. 2

[RÖPG18] ROVERI R., ÖZTIRELI A. C., PANDELE I., GROSS M.: Pointpronets: Consolidation of point clouds with convolutional neural networks. *CGF 37*, 2 (2018), 87–99. 3, 8, 9, 10

[SSW15] SUN Y., SCHAEFER S., WANG W.: Denoising point sets via l0 minimization. *CAGD 35* (2015), 2–15. 3

[Tau95] TAUBIN G.: Curve and surface smoothing without shrinkage. In *Computer Vision* (1995), IEEE, pp. 852–857. 6

[WHC*16] WEI L., HUANG Q., CEYLAN D., VOUGA E., LI H.: Dense human body correspondences using convolutional networks. In *CVPR* (2016), pp. 1544–1553. 1

[WKZ*16] WOLFF K., KIM C., ZIMMER H., SCHROERS C., BOTSCH M., SORKINE-HORNUNG O., SORKINE-HORNUNG A.: Point cloud noise and outlier removal for image-based 3d reconstruction. In *3D Vision* (2016), IEEE, pp. 118–127. 2, 15

[WLT16] WANG P.-S., LIU Y., TONG X.: Mesh denoising via cascaded normal regression. *ACM TOG 35*, 6 (2016), 232. 3, 12, 14

[WSL*18] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829* (2018). 1, 3, 8, 9, 10

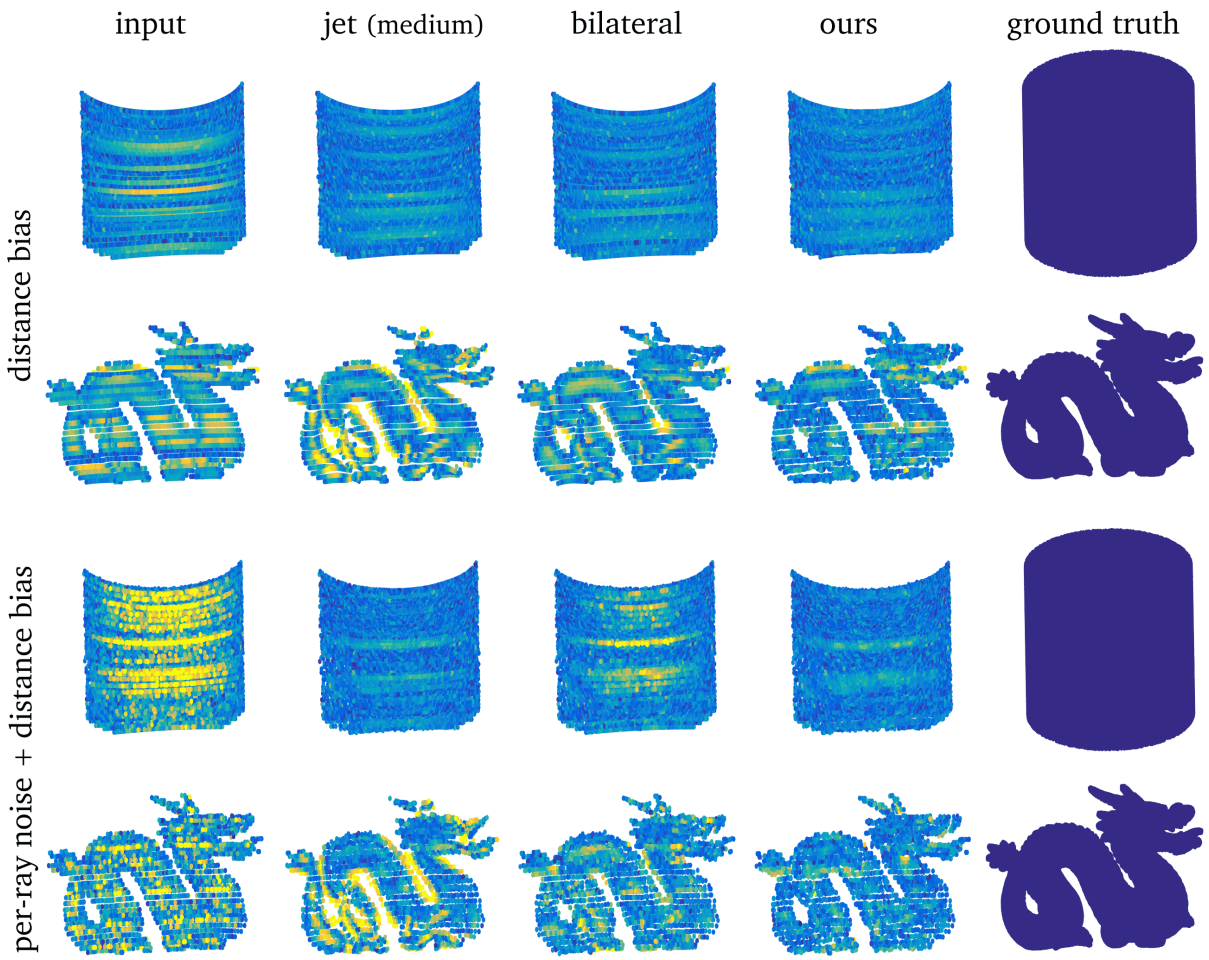

Figure 17: *Qualitative comparison with state-of-the-art methods on the Velodyne dataset. We display the normalized distance to the ground truth surface. The two top rows are evaluated on a dataset with only distance bias as noise and the two bottom rows with added per-ray noise. The simulated scanner noise has a high spatial correlation along the horizontal scan-lines, and lower correlation vertically across scan-lines. In this setting, jet fitting introduces significant error in detailed surface regions, while bilateral denoising has high residual error in the examples that have both noise types.* POINTCLEANNET *successfully learns the noise model, resulting in lower residual error.*

[YHCOZ18] YIN K., HUANG H., COHEN-OR D., ZHANG H.: P2p-net: Bidirectional point displacement net for shape transform. *ACM TOG 37*, 4 (July 2018), 152:1–152:13. 3, 15

[YLF*18] YU L., LI X., FU C.-W., COHEN-OR D., HENG P.-A.: Pu-net: Point cloud upsampling network. In *CVPR* (2018). 3, 15

[YLÖ*16] YOON Y.-J., LELIDIS A., ÖZTIRELI A. C., HWANG J.-M., GROSS M., CHOI S.-M.: Geometry representations with unsupervised feature learning. In *Big Data and Smart Computing* (2016), IEEE, pp. 137–142. 3

[ZCN*18] ZENG J., CHEUNG G., NG M., PANG J., YANG C.: 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model. *arXiv preprint arXiv:1803.07252* (2018). 3

[ZSW*10] ZHENG Q., SHARF A., WAN G., LI Y., MITRA N. J., COHEN-OR D., CHEN B.: Non-local scan consolidation for 3d urban scenes. *ACM TOG 29*, 4 (2010), 94–1. 3

[ZZC*17] ZHANG K., ZUO W., CHEN Y., MENG D., ZHANG L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE TIP 26*, 7 (2017), 3142–3155. 3

[ZZGZ17] ZHANG K., ZUO W., GU S., ZHANG L.: Learning deep cnn denoiser prior for image restoration. In *CVPR* (2017), vol. 2. 3

Appendix A:

Figure 17 shows a qualitative comparison with state-of-the-art methods on the *Velodyne* dataset described in Section 6.5.2.