

Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning

Pedro Hermosilla
Ulm University

Tobias Ritschel
University College London

Timo Ropinski
Ulm University

Abstract

We show that denoising of 3D point clouds can be learned unsupervised, directly from noisy 3D point cloud data only. This is achieved by extending recent ideas from learning of unsupervised image denoisers to unstructured 3D point clouds. Unsupervised image denoisers operate under the assumption that a noisy pixel observation is a random realization of a distribution around a clean pixel value, which allows appropriate learning on this distribution to eventually converge to the correct value. Regrettably, this assumption is not valid for unstructured points: 3D point clouds are subject to total noise, i. e., deviations in all coordinates, with no reliable pixel grid. Thus, an observation can be the realization of an entire manifold of clean 3D points, which makes a naïve extension of unsupervised image denoisers to 3D point clouds impractical. Overcoming this, we introduce a spatial prior term, that steers converges to the unique closest out of the many possible modes on a manifold. Our results demonstrate unsupervised denoising performance similar to that of supervised learning with clean data when given enough training examples - whereby we do not need any pairs of noisy and clean training data.

1. Introduction

While the amount of clean 3D geometry is limited by the manual effort of human 3D CAD modelling, the amount of 3D point clouds is growing rapidly everyday: our city’s streets, the interior of everyday buildings, and even the goods we consume are routinely 3D-scanned. Regrettably, these data are corrupted by scanner noise and as such not accessible to supervised learning that requires pairs of noisy and clean data. Consequently, it is desirable to be able to denoise the acquired noisy 3D point clouds by solely using the noisy data itself.

Two necessary recent developments indicate that this might be possible: deep learning on 3D point clouds [20] and unsupervised denoising of images [18, 16, 2].

Unfortunately, these two methods cannot be combined naïvely. To learn our unsupervised 3D point cloud denoisers (Fig. 1), we need to overcome two main limitations: the

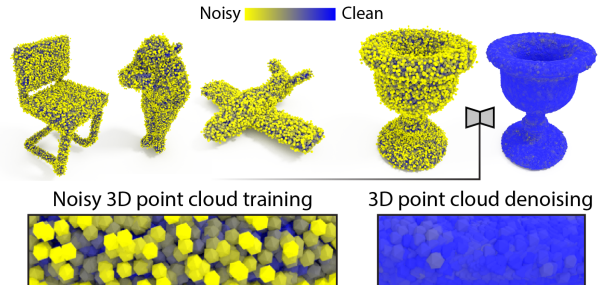


Figure 1. We learn 3D point cloud cleaning (right), unsupervised, from noisy examples alone (left).

practical obstacle to have a pair of two noisy scans of the same object and the theoretical difficulty that noise in 3D point clouds is *total*.

We refer to noise as ‘total’ (Fig. 2) when distortions are not confined to the range (pixel values) while the domain is clean (as pixel positions are), but to the more challenging setting where both domain and range are affected by noise. The name is chosen in analogy to total least squares [10], which dealt with simultaneous noise in domain and range, but for a linear, non-deep setting.

This paper’s evaluation shows for simulated noise of different kind, as well as for real point clouds, how our unsupervised approach nonetheless outperforms a supervised approach given enough, and in some cases even when given the same magnitude of, training data, while it runs efficiently in a single pass on large point clouds.

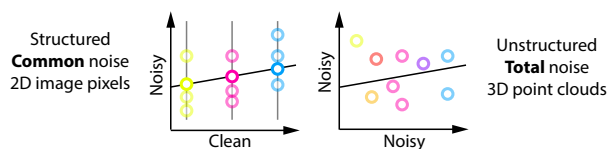


Figure 2. To learn denoising of 3D point clouds, we need to extend from common noise that is clean in one part of the signal, to a total setting, where all parts of the signal are noisy. This example shows three realizations of common noise (left) and total noise (right) for three samples (colors). Please note, how total noise is “more noisy” as both axis are corrupted.

2. Related Work

Image denoising. Denoising images is one of the most basic image manipulation operations. The most primitive variants are based on linear filters such as the Gaussian filter, eventually with additional sharpening [14]. While non-linear filters, such as median, bilateral [25] or non-local means [3] are frequently used in practice, state-of-the-art results are achieved by optimizing for sparsity [6]. Recently, it has become popular to learn denoising, when pairs of clean and noisy images are available [4].

Lehtinen et al. [18] proposed a method to learn denoising with access to only two noisy images, instead of a clean-noisy pair. Taking it a step further, Noise2Void [16] and Noise2Self [2] are two extensions that remove the requirement to have two copies of one image corrupted with noise and instead work on a single image. In both cases, this is achieved by regressing the image from itself. This is done by creating a receptive field with a “blind spot”, and a network regresses the blind spot from its context. We will detail the theory behind those papers [18, 16, 2] in Sec. 3.

3D point cloud denoising. 3D point clouds capture fine spatial details but remain substantially more difficult to handle than images, due to their irregular structure [19].

As for images, linear filters can be applied to remove noise [17], but at the expense of details. As a remedy, image operators such as bilateral [8, 5], non-local means [22] or sparse coding [1] have been transferred to point clouds.

With the advent of PointNet [20], deep learning-based processing of point clouds has become tractable. Four notable deep methods to denoise 3D point clouds were suggested. The first is PointProNet, that denoises patches of points by projecting them to a learned local frame and using Convolutional Neural Networks (CNN) in a supervised setup to move the points back to the surface [23]. However, the accuracy of the method is determined by the accuracy of the local frame estimation, which results in artifacts in extreme sharp edges. The second approach by Rakotosaona et al. [21] uses PCPNet [12] (a variant of PointNet [20]) to map noisy point clouds to clean ones. Third, Yu et al. [31] learns to preserve edges, that dominate man-made objects. Finally, Yifan et al. [30] define a clean surface from noisy points by upsampling.

All these deep denoising approaches are supervised, as they require pairs of clean and noisy point clouds, which in practice are produced by adding noise to synthetic point clouds. Our approach does not require such pairs.

Noise and learning. Noise is an augmentation strategy used in denoising auto-encoders [26]. These are however not aiming to denoise, but add noise to improve robustness. Also is their target not noisy, but noise is in the input or

added to internal states.

3. Denoising Theory

Based on denoising in the regular domain, i. e., images, with or without supervision, we will establish a formalism that can later also be applied to derive our unstructured 3D case.

3.1. Regular Domains

Pixel noise. An observation \mathbf{y}_i at pixel i in a noise corrupted image is a sample of a noise distribution $\mathbf{y}_i \sim p(\mathbf{z}|\mathbf{x}_i)$ around the true value \mathbf{x}_i . This is shown in Fig. 3, a). The black curve is the true signal and pixels (dotted vertical lines) sample it at fixed positions i (black circles) according to a sampling distribution $p(\mathbf{z}|\mathbf{x}_i)$ (yellow curve) around the true value (pink circle).

Supervised. In classic supervised denoising, we know both a clean \mathbf{x}_i and a noisy value $\mathbf{y} \sim p(\mathbf{z}|\mathbf{x}_i)$ for pixel i and minimize

$$\arg \min_{\Theta} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{z}|\mathbf{x}_i)} l(f_{\Theta}(\mathbf{y}), \mathbf{x}_i),$$

where f is a tunable function with parameters Θ , and l is a loss such as L_2 . Here and in the following, we omit the fact that the input to f comprises of many \mathbf{y} that form an entire image, or at least a patch. We also do not show an outer summation over all images (and later, point cloud) exemplars.

Unsupervised, paired. Learning a mapping from one noisy realization of an image to another noisy realization of the same image is achieved by Noise2Noise [18]. It has been shown, that learning

$$\arg \min_{\Theta} \mathbb{E}_{\mathbf{y}_1 \sim p(\mathbf{z}|\mathbf{x}_i)} \mathbb{E}_{\mathbf{y}_2 \sim p(\mathbf{z}|\mathbf{x}_i)} l(f_{\Theta}(\mathbf{y}_1), \mathbf{y}_2),$$

converges to the same value as if it had been learned using the mean / median / mode of the distribution $p(\mathbf{z}|\mathbf{x})$ when l is L_2 / L_1 / L_0 . In most cases, i. e., for mean-free noise, the mean / median / mode is also the clean value. We refer to this method as ‘paired’, as it needs two realizations of the signal, i. e., one image with two realizations of noise.

Unsupervised, unpaired. Learning a mapping from all noisy observations in one image, except one pixel, to this held-out pixel is achieved by Noise2Void [16] and Noise2Self [2]:

$$\arg \min_{\Theta} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{z}|\mathbf{x}_i)} l(f_{\Theta}(\mathbf{y}), \mathbf{y}),$$

Here, f is a special form of \mathcal{J} -incomplete [2] maps that have no access to pixel i when regressing it, i. e., a ‘blind

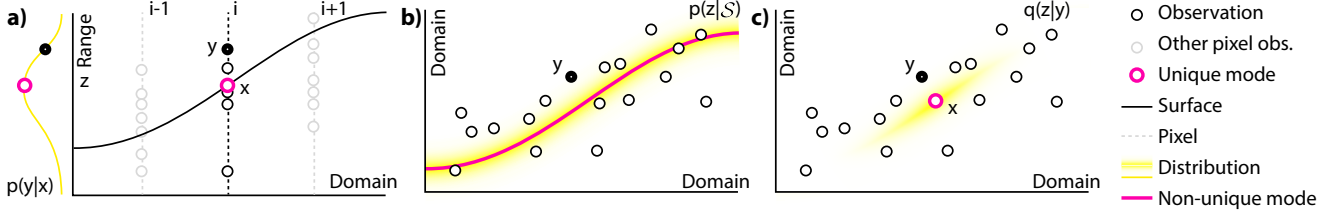


Figure 3. Substantial differences exist when denoising structured and unstructured data. (a) For structured data, each pixel value follows a sampling distribution $p(z|x_i)$ (yellow curve) around the true value (pink circle). (b) For unstructured data, the distribution $p(z|S)$ has a manifold of modes (pink line). (c) By using the proposed proximity-appearance prior, a unique mode closest to the surface is determined.

spot’. The same relation between mean / median / mode and loss as in Noise2Noise applies. Note that this formulation does not require two images, and we, therefore, refer to it as ‘unpaired’.

Discussion. All three methods described above, work under the assumption that, in a structured pixel grid, the range (vertical axis in Fig. 2, left and Fig. 3, a) axis i and the domain z (horizontal axis) have different semantics. The noise is only in the range: it is not uncertain *where* a pixel is, only what its true *value* would be.

3.2. Unstructured Domains

Point noise. As for pixels, we will denote clean points as x , noisy points as y and the noise model as p . All points in our derivation can be either positional with XYZ coordinates, or positional with appearance, represented as XYZRGB points.

To our knowledge, deep denoising of colored point clouds has not been proposed. We will not only show how our technique can also be applied to such data but moreover, how color can help substantially to overcome challenges when training unsupervised learning of a point cloud denoiser. Surprisingly, this benefit can be exploited during training, even when no color is present at test time. If available, it will help, and we can also denoise position and appearance jointly.

Supervised. Denoising a point cloud means to learn

$$\arg \min_{\Theta} E_{y \sim p(z|S)} l(f_{\Theta}(y), S),$$

the sum of the losses l (e. g., Chamfer) between $f_{\Theta}(y)$ and the surface S of the 3D object. Such supervised methods have been proposed, but they remain limited by the amount of training data available [23, 21], as they require access to a clean point cloud.

4. Unsupervised 3D Point Cloud Denoising

We will first describe why a paired approach is not feasible for unstructured data before we introduce our unpaired, unsupervised approach.

4.1. Inapplicability of ‘Paired’ Approaches

Learning a mapping $f_{\Theta}(\mathcal{Y}_1) = \mathcal{Y}_2$ from one noisy point cloud realization \mathcal{Y}_1 to another noisy point cloud realization \mathcal{Y}_2 that both have the same clean point cloud \mathcal{X} and where the i -th point in both point clouds is a realization of the i -th ground truth value, would be a denoiser in the sense of Noise2Noise [18]. Regrettably, Noise2Noise cannot be applied to unsupervised learning from unstructured point clouds for two reasons.

First, this paired design, same as for images, would require supervision in the form of two realizations of the same point cloud corrupted by different noise realizations. While this is already difficult to achieve for 2D image sensors, it is not feasible for 3D scanners.

Second, it would require a network architecture to know which point is which, similar as it is given by the regular structure of an image that explicitly encodes each pixel’s identity i . This is never the case for total noise in points. Opposed to this, modern convolutional deep point processing [20, 13] is exactly about becoming invariant under re-ordering of points.

In order to overcome this problem in a supervised setting, Rakotosaona et al. [21] simulated such pairing by selecting, for each noisy observation, the closest point in the clean point cloud as the target for the loss. However, this is just an approximation of the real surface whose accuracy depends on the quality of the sampling of the clean data. Fortunately, we can show that a pairing assumption is not required, such that our approach operates not only unsupervised but also unpaired, as we will detail next.

4.2. Unpaired

Learning a mapping from a noisy realization to itself $f_{\Theta}(\mathcal{Y}) = \mathcal{Y}$ is an unsupervised and unpaired denoiser in the sense of Noise2Void [16] or Noise2Self [2]. Defining \mathcal{I} incompleteness in a point cloud is no difficulty: just prevent access of f to point y itself when learning point y from the neighbors of y . Thus, essentially, we train a network to map each point to itself without information about itself. Unfortunately, there is the following catch with total noise.

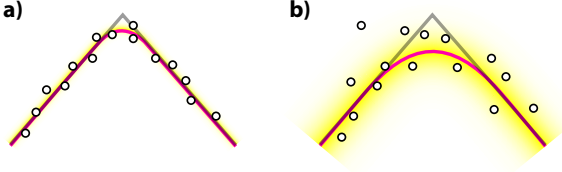


Figure 4. Comparing small (*left*) and large noise (*right*) we see the modes (*pink*) deviate from the GT surface (*black*).

Problem statement. Different from observing pixels at index i in an image (dotted line Fig. 3, a), which tell us that \mathbf{y} is a realization of a hidden value \mathbf{x}_i to infer, it is unknown which hidden surface point is realized when observing a point in an unpaired setting. A noisy point observation \mathbf{y} , can be a realization of $p(\mathbf{z}|\mathbf{x}_1)$ in the same way as it could be a realization of $p(\mathbf{z}|\mathbf{x}_2)$. Consequently, the distribution $p(\mathbf{z}|\mathcal{S})$ has a manifold of modes (pink line in Fig. 3, b). Learning a mapping from a noisy realization to itself will try to converge to this multimodal distribution, since, for the same neighborhood, the network will try to regress different points from this distribution at the same time.

We, therefore, have to look into two questions. First, what can be said about the similarity of this manifold of modes and the clean surface? And second, how can we decide which of the many possible modes is the right one? Answering the second, and deriving bounds for the first question are the key contributions of this paper, enabling unsupervised 3D point cloud denoising.

4.3. Manifold of Modes vs. Clean Surface

Concerning the first question, the manifold of modes is close to the surface, but not identical. Fig. 4, a), shows a clean surface as a black line, with a small amount of noise, where most samples are close to the clean surface. In this condition, the learning converges to a solution identical to a solution it would have converged to, as when trained on the pink line, which is very similar to the clean surface. With more noise, however, it becomes visible in Fig. 4, b) that this manifold is not identical to the surface.

We note, that the mode surface is the convolution of the true surface and the noise model p . We cannot recover details removed by this convolution. This is different from supervised NN-based deconvolution, which has access to pairs of convolved and clean data. In our case, the convolution is on the limit case of the learning data and we never observe non-convolved, clean data.

It is further worth noting, that not all noise distributions lead to a manifold that is a surface in 3D or would be a connected path in our 2D illustrations. Only uni-modal noise distributions, such a scanner noise, have no branching or disconnected components. Our solution will not depend on the topology of this mode structure.

4.4. Unique Modes

As explained above, the naïve implementation of unsupervised unpaired denoising will not have a unique mode to converge to. Therefore, we regularize the problem by imposing the prior $q(\mathbf{z}|\mathbf{y})$ that captures the probability that a given observation \mathbf{y} is a realization of the clean point \mathbf{z} .

We suggest using a combination of spatial and appearance proximity

$$q(\mathbf{z}|\mathbf{y}) = p(\mathbf{z}|\mathcal{S}) * k(\mathbf{z} - \mathbf{y}) \quad (1)$$

$$k(\mathbf{d}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathbf{W}\mathbf{d}\|_2^2}{2\sigma^2}\right), \quad (2)$$

where σ is the bandwidth of k and $\mathbf{W} = \text{diag}(w)$ is a diagonal weight matrix trading spatial and appearance locality. We use a value $w = 1/\alpha r$, r being 5% of the diameter of the model and α a scaling factor. In the case of point clouds with appearance, we use $w = \beta$ in the appearance rows/columns, otherwise, we only consider proximity. For more details about the values for such parameters please refer to the supplementary material.

This results in convergence to the nearest (in space and appearance) mode when optimizing

$$\arg \min_{\Theta} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{z}|\mathcal{S})} \mathbb{E}_{\mathbf{q} \sim q(\mathbf{z}|\mathbf{y})} l(f_{\Theta}(\mathbf{y}), \mathbf{q}), \quad (3)$$

The effect of this prior is seen in Fig. 3, c). Out of many modes, the unique closest one remains.

Note, that our choice of a Gaussian prior q is not related to a Gaussianity of the noise model p , which we do not assume. The only assumption made here is that out of many explanations, the closest one is correct. We experimented with other kernels such as Wendland [28] and inverse multi-quadratic but did not observe an improvement.

Appearance to the rescue As mentioned above, 3D point clouds that come with RGB color annotation are a surprising opportunity to further overcome the limitations of unsupervised training. Otherwise, in some cases, the spatial prior cannot resolve round edges. This is not because the network f is unable to resolve them, but because unsupervised training does not ‘see’ the sharp details. Fig. 5 details how colors resolve this: without RGB, the corners are rounded in Fig. 5, a). When adding color, here red and blue (Fig. 5, b), the points become separated (Fig. 5, c). The sampling of the prior $q(\mathbf{z}|\mathbf{y})$ on a red point, will never pick a blue one and vice-versa. Consequently, the learning behaves as if it had seen the sharp detail.

Thus, using color in the prior reinforces some of the structure, which was lost when not relying on a regular pixel grid. We do not know which noisy point belongs to which measurement, but we have a strong indication, that something of different color, is not a noisy observation of

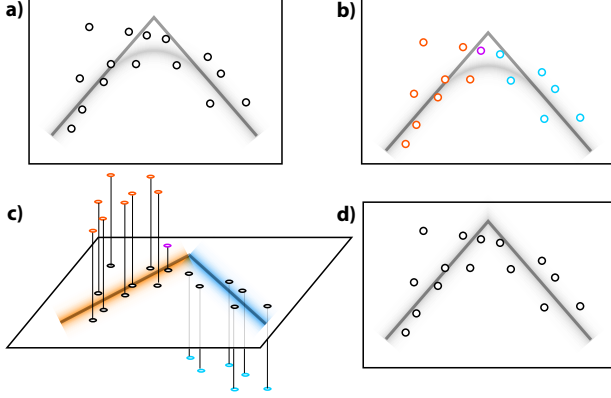


Figure 5. Bilaterality: The manifold of modes of the distribution of a 2D point cloud without color can be curved for strong noise (a). Different appearancea, denoted as red and blue points in b, can be used to establish bilateral distances, lifting points to 3D. The resulting manifold of modes (d) now preserves sharp appearance edges.

the same point. Of course, it is possible, that two observations y_1 and y_2 appear to be from a different point, but happen to be measurements of the same clean surface point x , whereby range noise affects the color. Fortunately, such spurious false negatives are less problematic (they create variance) than the permanent false positives, that lead to rounding (bias). Symmetrically, and maybe more severe, a difference in color is not always a reliable indicator of a geometric discontinuity either. It is, if color is dominated by shading, but texture and shadow edges may lead to a high false negative distance. Note, that the color is only required for training and never used as input to the network.

4.5. Converging to the mode

We train the network to converge to the mode of the prior distribution $q(z|y)$ by using the approximation of the L_0 loss function proposed by Lehtinen et al. [18], $(|f_\Theta(y) - q| + \epsilon)^\gamma$, where their $\epsilon = 10^{-8}$, and their γ is annealed from 2 to 0 over the training.

Thus, our unsupervised training converges to the same as training supervised to find the closest – in XYZ space and RGB appearance, when available – mode on the distribution $S * p$ resulting from convolving the clean surface S and the noise model p .

4.6. Implementation

Prior. To minimize Eq. 3 we need to draw samples according to the prior q which is implemented using rejection sampling: we pick a random point \hat{q} from \mathcal{Y} within r from y , and train on it only if $k(\hat{q} - y) > \xi$ for a uniform random $\xi \in (0, 1)$. In practice, a single sample is used to estimate this inner expected value over $q(z|y)$.

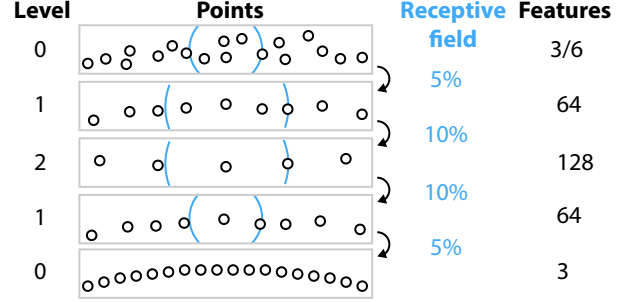


Figure 6. Architecture overview: We start from a noisy point cloud in the top and perform two levels of unstructured encoding, that reduce the receptive field, followed by two levels of decoding using transposed unstructured convolutions.

Architecture. We implement f using an unstructured encoder-decoder based on Monte Carlo convolution [13] (Fig. 6). Such an architecture consumes the point cloud, transforms spatial neighborhoods into latent codes defined on a coarser point set (encoder), and up-sample these to the original point resolution (decoder). The effective receptive field, so the neighborhood from which the NN regressed points are considered, is 30 % of the diameter of the model. In particular, we perform two levels of encoding, the first with a receptive field of 5 %, the second at 10 %. The Poisson disk radii for pooling in Level 1 and Level 2 are half the size of the receptive fields.

This architecture is fast to execute, allowing to denoise in parallel 800 K points in 13 seconds on a single machine with a GeForce RTX 2080. Moreover, this architecture is composed of only 25 K trainable parameters, orders of magnitude smaller than other networks (0.8 million for PointNet or 1.4 million for PointNet++).

Training. Besides these benefits, our method is also easy to implement as seen in Alg. 1. Here, \mathcal{Q} denotes a set of prior samples q for all points in a point cloud. All operations are defined on batches that have the size of the point cloud. We use an ADAM optimizer [15] with an initial learning rate of .005, which is decreased during training.

Algorithm 1 Unsupervised point cloud denoiser training

- 1: **for** all noisy point clouds \mathcal{Y} **do**
 - 2: $\Xi \leftarrow \text{RANDOMUNIFORMBATCH}(0, 1)$
 - 3: $\mathcal{Q} \leftarrow \text{SAMPLEPRIORBATCH}(\mathcal{Y}, \Xi)$
 - 4: $\Theta \leftarrow \text{MINIMIZEBATCH}(\|f_\Theta(\mathcal{Y}) - \mathcal{Q}\|_0)$
 - 5: **end for**
-

Iteration. Similar as in previous work [21], our results improved if the output of the network is fed as input again. However, this introduces two problems: clustering

of points, and shrinking of the point cloud after several iterations. We address these problems in a similar way as Rakotosaona et al. [21]. In order to prevent clustering we introduce the following regularization term that enforces a point cloud with equidistant samples:

$$L_r = \arg \min_{\Theta} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{z}|\mathcal{S})} \max_{\mathbf{y}' \in n(\mathcal{Y}, \mathbf{y})} \|f_{\Theta}(\mathbf{y}), f_{\Theta}(\mathbf{y}')\|_2$$

where $n(\mathcal{Y}, \mathbf{y})$ is the set of points from the noisy point cloud within a patch centered at \mathbf{y} . To prevent shrinking we remove low-frequency displacements before translating the noisy points. The supplemental materials show the effect of these iterations.

5. Evaluation

Our experiments explore the application, both to synthetic (Sec. 5.2) and to real data (Sec. 5.3). For synthetic data, we know the answer and can apply different metrics to quantify the performance of our approach, while we do not know the ground truth for real data and results are limited to a qualitative study.

5.1. Setup

Data set. We have collected 15 different classes with 7 different polygonal models each (5 for training and 2 for testing) from ModelNet-40 [29] and sampled the surface with points as explained next. As we optionally use RGB appearance, it is computed using Lambertian shading from 3 random directional lights.

Sampling. We simulate different forms of noise to corrupt the clean data of the synthetic data set.

In the SIMPLE noise model, we sample each mesh using Poisson Disk sampling [27] to obtain clean point clouds within the range of 13 K and 190 K points each, resulting in 22 million points for training and 10 million points for testing. Then, we add Gaussian noise with a standard deviation of .5%, 1%, and 1.5% of the bounding box diagonal.

The ADVANCED sampling emulates true sensor noise making use of Blendsor [11], a library to simulate sensor noise. In particular, we choose to emulate a Velodyne HDL-64E 3D scan. These devices introduce two types of noise in the measurements, a distance bias for each laser unit and a per-ray Gaussian noise. In our data, we use a standard deviation of .5% of the diagonal of the bounding box for the distance bias and three different levels of per-ray noise, .5%, 1%, and 1.5%. This generates point clouds within the range of 3 K and 120 K points each, resulting in 12 million points for training and 5 million points for testing.

We investigate levels of distortion where a surface is still conceivable. More severe corruptions with uneven sampling or outliers are to be explored in future work.

Metric. We use the Chamfer distance from Fan et al. [7],

$$d(\mathcal{Y}, \mathcal{S}, \mathcal{X}) = \frac{1}{N} \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{y} - \mathbf{s}\|_2 + \frac{1}{M} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{y} - \mathbf{x}\|_2$$

where less is better. The first term measures the average distance between the predicted points to their closest point in a polygonal surface \mathcal{S} . The second term measures how the points are distributed in the ground truth surface.

Since the clean point clouds \mathcal{X} follow a Poisson Disk distribution, by measuring their distance to the closest predicted point we are able to determine if the surface is equally covered by our predictions.

The metric is applied in the test data with three different realizations of noise and averaged over two complete trainings to reduce variance in the estimate of the metric.

Methods. We compare our unsupervised approach with classical methods as well as supervised machine learning approaches. To obtain insights regarding the effectivity of the individual subparts, we investigate ablations with and without the spatial and/or the appearance prior.

The classic baselines are MEAN and BILATERAL [5], which are also unsupervised. Their parameters are chosen to be optimal on the training set.

As supervised learning-based denoisers, we use the same architecture, as we have employed for the unsupervised setting, whereby we use the training algorithm proposed in PointCleanNet [21]. While this means, that we do not use the original PointCleanNet network architecture, which is based on PointNet, we believe that our evaluation is more insightful with a unified architecture – especially since the architecture employed by us has outperformed PointNet on a range of other tasks [13].

Finally, we study three variants of our approach. The first one is with NO PRIOR. The second we denote as NO COLOR, which is our prior but only based on proximity. The last one is FULL which includes all our contributions. More precisely, we use XYZ point clouds for NO PRIOR and NO COLOR and XYZRGB point clouds for FULL. Again, color is only used to sample the prior, not as an input to the network during training or testing.

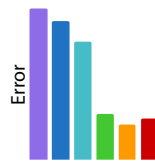
5.2. Quantitative Results

Denoising performance. We start with SIMPLE noise and look into ADVANCED later. A comparison of the average error across the test set for different methods is shown in Tbl. 1, whereby each column represents one method. All methods are trained with the same amount of training exemplars, that is, 22 million points.

As can be seen, our full method (orange) performs best. We even outperform the supervised competitor (red), likely

Table 1. Error (less is better) per method on SIMPLE noise.

Ours					
Mean	Bilat.	No p.	No c.	Full	Sup.
.598	.592	.582	.547	.542	.545



Method	Error
Mean	.598
Bilateral	.592
No p.	.582
No c.	.547
Full	.542
Supervised	.545

because the network has to find more valid generalizations and is less prone to over-fitting. As can also be seen, the other non-learned methods like mean (violet) and bilateral (blue) are not competitive, even when tuned to be optimal on training data. We further see a clear distinction between ablations of our method and the full approach. When training without a spatial prior (cyan), the method is much worse than supervised and only slightly better than mean. A method not using color for training (green) – but including the spatial prior – can achieve almost full performance, but only adding color will outperform supervised.

This comparison is on the same amount of data. However, in most real-world scenarios, the number of noisy point clouds can be assumed to be much higher than the amount of clean ones. We will study this relation next.

Supervision scalability. We will now study, how supervised method scale with the number of clean point clouds and our method with the amount of noisy point clouds.

The outcome is seen in Tbl. 2 where different methods are columns and different amounts of training data are rows. The plot to the right shows the relation as a graph. We show the logarithmic fit to the individual measurements shown as points. The color encoding is identical for all plots. The difference between the methods is measured wrt. the number of total training points ranging from .5 to 22 millions.

Not unexpected, we see all methods benefit from more training data. We see that our method performs better than supervised across a range of training data magnitudes. At around 22 million points, the highest we could measure, the red and orange lines of our full model and supervised cross. That only means, that after this point, our unsupervised method needs more training data to achieve the same performance as a supervised method.

We further see that the ablations of our method without

Table 2. Error (less is better) for different amount of supervision.

Train data	Ours				Sup.
	No p.	No c.	Full		
.5 M	.587	.557	.558	.574	
1 M	.584	.550	.557	.563	
4 M	.584	.553	.543	.546	
22 M	.582	.547	.542	.545	

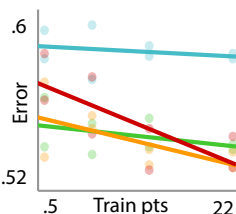
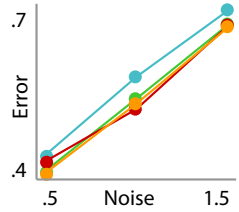


Table 3. Error (less is better) for different levels of SIMPLE noise.

Noise Levels	Ours			
	No p.	No c.	Full	Sup.
1.5 %	.734	.698	.691	.695
1.0 %	.578	.534	.525	.515
0.5 %	.435	.411	.408	.426



a prior and without color do not only perform worse, but also scale less favorable, while ours (orange) is similar to supervised (red). Admittedly, supervised scales best.

Amount of noise. While we have studied the average over three levels of noise in the previous plots, Tbl. 3 looks into the scalability with noise levels in units of scene diameter percentages. We find that error is increasing as expected with noise, but all methods do so in a similar fashion. In two cases, we win over supervised, in one case supervised wins, resulting in the improved average reported above.

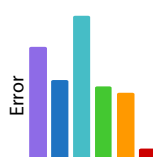
Denosing performance. While we have studied SIMPLE Gaussian noise, we now relax this assumption and explore ADVANCED simulated scanner noise generated as explained in Sec. 5.1. Contrary to real scanned data, it has the benefit that the ground truth is known.

Tbl. 4 shows the error of different methods for this type of noise. We see, that in this case, our full method (orange) performs better than any other unsupervised method, such as mean or bilateral (violet and blue). A supervised method can perform better than other methods for this noise at the same amount of training data input, 12 million points. Finally, we also see that ablations without the suggested prior (cyan and green) have a higher error, indicating the priors are equally relevant for this type of noise, too.

Upgrading Notably, we can upgrade any supervised denoiser in a code-transparent, architecture-agnostic fashion to become unsupervised. Consider a supervised denoiser, which takes clean-noisy pairs instead of noisy ones only, as we do. To apply our method, all we do is to resample the point using our spatial and / or color prior, and “pretend” this to be the clean point cloud.

Table 4. Error (less is better) per method on ADVANCED noise.

	Ours					
Mean	Bilat.	No p.	No c.	Full	Sup.	Error
.378	.362	.393	.359	.356	.329	



Method	Error
Mean	0.378
Bilateral	0.362
No p.	0.393
No c.	0.359
Full	0.356
Supervised	0.329

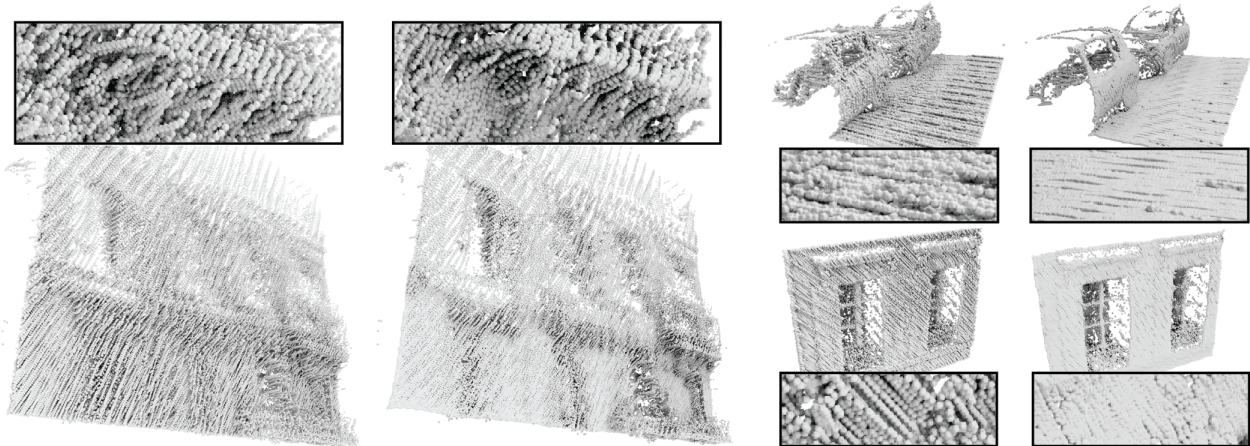


Figure 7. Multiple real world pairs of noisy input scans (*left*) and the result of our denoiser (*right*), accompanied by zoomed insets.

We have done so for Point-CleanNet [21] and evaluated on their dataset. We see even without modifying their architecture or supervision, we still slightly outperform theirs.

	PCNet [21]	
	Noisy Sup.	Our
	4.54 1.36	1.34

5.3. Qualitative Results

Here, we repeat the above experiments, on real world noisy point clouds from a Mobile Laser Scanning setup based on a Velodyne HDL32 3D scanner. We used the Paris-rue-Madame data set [24] which is composed of 20 million points. We subdivide the model into parts of ca. 150 K points each, resulting in 17 million points used during training and 3 million points for testing. Note that in this setting, noise is part of the data and does not need to be simulated. Furthermore, and most important, no clean ground truth is available. Consequentially, the error cannot be quantified and we need to rely on human judgment.

We see in Fig. 7, how our method removes the noise and produces a clean point cloud without shrinking, with uniform sampling as well as details. We cannot apply a visualization of the error, as the ground truth is unknown. We instead provided point cloud renderings by representing the point clouds as a mesh of spheres with shading.

5.4. Ablation

No prior When not using the prior in space (green), the denoiser learned across different types of noise (Tbl. 1), magnitudes of noise (Tbl. 3) and amounts of training data (Tbl. 2) is consistently worse and not much better than Gaussian or bilateral. This indicates it is essential.

No appearance Making use of appearance consistently improves the outcome across the aforementioned three axes

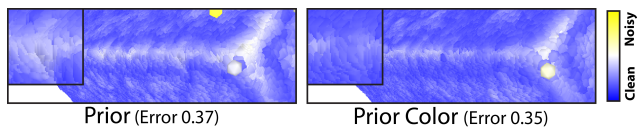


Figure 8. Including and not including color in the prior.

of variations in Tbl. 1 (and Tbl. 4), Tbl. 2 and Tbl. 3, either taking the quality beyond supervision or very close to it.

Effect of color Fig. 8 shows a sharp edge with two different colors to be denoised. Including the color, slightly reduces the error (less high-error yellow, more blue).

6. Conclusions

We have presented an unsupervised learning method to denoise 3D point clouds without needing access to clean examples, and not even noisy pairs. This allows the method to scale with natural data instead of clean CAD models decorated with synthetic noise. Our achievements were enabled by a network that maps the point cloud to itself in combination with a spatial locality and a bilateral appearance prior. Using appearance in the prior is optional, but can improve the result, without even being input to the network, neither at test nor at training time. Denoising with color as input, as well as joint denoising of color and position, remains future work. Our results indicate we can outperform supervised methods, even with the same number of training examples.

Acknowledgements This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG), grant RO 3408/2-1 (ProLint), and the Federal Ministry for Economic Affairs and Energy (BMWi), grant ZF4483101ED7 (VRReconstruct). We acknowledge Gloria Fackelmann for the supplementary video narration.

References

- [1] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. l_1 -sparse reconstruction of sharp point set surfaces. *ACM Trans. Graph.*, 29(5):135, 2010. [2](#)
- [2] Joshua Batson and Loïc Royer. Noise2Self: Blind denoising by self-supervision. *CoRR*, abs/1901.11365, 2019. [1](#), [2](#), [3](#)
- [3] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *CVPR*, pages 60–5, 2005. [2](#)
- [4] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, pages 2392–2399, 2012. [2](#)
- [5] Julie Digne and Carlo de Franchis. The Bilateral Filter for Point Clouds. *Image Processing On Line*, 7:278–287, 2017. [2](#), [6](#)
- [6] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Trans. Image Processing*, 15(12):3736–45, 2006. [2](#)
- [7] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016. [6](#)
- [8] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph.*, 22(3):950–3, 2003. [2](#)
- [9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. [10](#), [11](#)
- [10] Gene H Golub and Charles F Van Loan. An analysis of the total least squares problem. *SIAM J Numerical Analysis*, 17(6):883–893, 1980. [1](#)
- [11] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Bensor: blender sensor simulation toolbox. In *Int. Symposium on Visual Computing*, pages 199–208, 2011. [6](#)
- [12] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNET: learning local shape properties from raw point clouds. *CoRR*, abs/1710.04954, 2017. [2](#)
- [13] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vazquez, Alvar Vinacua, and Timo Ropinski. Monte Carlo convolution for learning on non-uniformly sampled point clouds. *ACM Trans. Graph.*, 37(6), 2018. [3](#), [5](#), [6](#)
- [14] Anil K Jain. *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall., 1989. [2](#)
- [15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. [5](#)
- [16] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2Void - learning denoising from single noisy images. *CoRR*, abs/1811.10980, 2018. [1](#), [2](#), [3](#)
- [17] In-Kwon Lee. Curve reconstruction from unorganized points. *Computer aided geometric design*, 17(2):161–177, 2000. [2](#)
- [18] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2Noise: Learning image restoration without clean data. *ICML*, 2018. [1](#), [2](#), [3](#), [5](#)
- [19] Marc Levoy and Turner Whitted. *The use of points as a display primitive*. UNC Chapel Hill Technical Report, 1985. [2](#)
- [20] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *CoRR*, abs/1612.00593, 2016. [1](#), [2](#), [3](#)
- [21] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J. Mitra, and Maks Ovsjanikov. POINTCLEAN-NET: learning to denoise and remove outliers from dense point clouds. 2019. [2](#), [3](#), [5](#), [6](#), [8](#)
- [22] Guy Rosman, Anastasia Dubrovina, and Ron Kimmel. Patch-collaborative spectral point-cloud denoising. In *Computer Graphics Forum*, volume 32, pages 1–12, 2013. [2](#)
- [23] Riccardo Roveri, A. Cengiz Öztireli, Ioana Pandele, and Markus H. Gross. Pointpronets: Consolidation of point clouds with convolutional neural networks. *Comput. Graph. Forum*, 37(2):87–99, 2018. [2](#), [3](#)
- [24] Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. Paris-rue-madame database - a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *ICPRAM*, 2014. [8](#), [10](#), [13](#)
- [25] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, page 839, 1998. [2](#)
- [26] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J Machine Learning Res.*, 11:3371–408, 2010. [2](#)
- [27] Li-Yi Wei. Parallel poisson disk sampling. *ACM Trans. Graph.*, 27(3):20:1–20:9, 2008. [6](#)
- [28] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics*, 4(1):389–96, 1995. [4](#)
- [29] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. [6](#)
- [30] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3D point set upsampling. In *CVPR*, 2019. [2](#)
- [31] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *ECCV*, pages 386–402, 2018. [2](#)

Supplementary material

A. Error distribution

We studied the distribution of points wrt. the distance to the underlying surface in our test dataset. Fig. 9 (left) presents a histogram with the results of such study. Here we can see how the distribution of points has its maximum at distance 0.0 from the real surface and decreases with distance, following the same distribution as in the supervised setting.

B. Convergence

Fig. 9 (right) illustrates the evaluation error during training for two networks trained on the smallest dataset (.5 Million points). One network is trained with our algorithm (FULL) and the other one with supervised training. We can see that in both cases the loss decreases during training similarly. However, our algorithm converges to a lower error than supervised learning. With more training data, this gap is reduced until the curves cross.

C. Prior Kernel

We experimented with different kernels for our prior: Gaussian (k_g), Wendland (k_w), and Inverse Multi-Quadric (k_i). All kernels were adjusted to fit in the range $[-1, 1]$:

$$k_g(\mathbf{d}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathbf{W}\mathbf{d}\|_2^2}{2\sigma^2}\right),$$

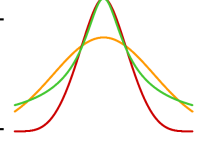
$$k_w(\mathbf{d}) = (1 - \|\mathbf{W}\mathbf{d}\|_2^2)^4 (1 + 4\|\mathbf{W}\mathbf{d}\|_2^2),$$

$$k_i(\mathbf{d}) = \frac{1}{\sqrt{1 + (5\|\mathbf{W}\mathbf{d}\|_2^2)^2}},$$

where $\mathbf{W} = \text{diag}(w)$ and $w = 1/\alpha r$. We performed an ablation study for different values of α and for each kernel independently. For the Gaussian kernel we used $\alpha_1 = .3$, $\alpha_2 = .5$, and $\alpha_3 = .7$, and for the Wendland and Inverse Multi-Quadric kernel we used $\alpha_1 = .8$, $\alpha_2 = 1.$, and $\alpha_3 = 1.2$. Tbl. 5 shows that the best performance was obtained by the Gaussian kernel.

Table 5. Error obtained for different functions used as our prior.

Prior	α_1	α_2	α_3
Gauss. (k_g) ●	.567	.548	.556
Wedland (k_w) ●	.595	.560	.561
Inv. MQ (k_i) ●	.577	.577	.582



D. Noise levels

Tbl. 3 on the paper tests one neural network trained for all noise levels at the same time. In this experiment, we have trained our network using only one level of noise and report its test error for all levels. We can see from the table, that performance drops only marginally.

Train	Test		
	0.5%	1.0%	1.5%
0.5%	.435	.576	.738
1.0%	.431	.551	.708
1.5%	.467	.594	.741
All	.411	.534	.698

E. Iterative refinement

Since our approximation of the surface during training is not exact, the results improve by applying our network several times iteratively. Fig. 10 presents a visual encoding of the distance of each point to the real surface after applying each denoising step.

F. Additional Qualitative Results

We trained our network with the Kitti dataset [9] in order to evaluate the robustness of our method with sparse point clouds. Fig. 11 presents some qualitative results. We can see that our network is able to remove the noise successfully.

Moreover, Fig. 12, and Fig. 13 provide more qualitative results on all the datasets used in the evaluation. In Fig. 12 we can see how our algorithm obtains a quality similar to the network trained with supervised data. Fig. 13 illustrates the result of applying our method to models from the Paris-rue-Madame Database [24] composed of 375 K-800 K points.

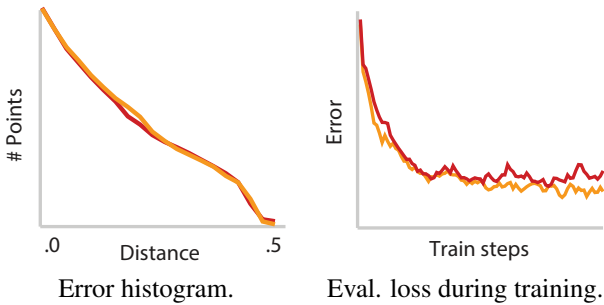


Figure 9. Ours (FULL) ● vs Supervised ●

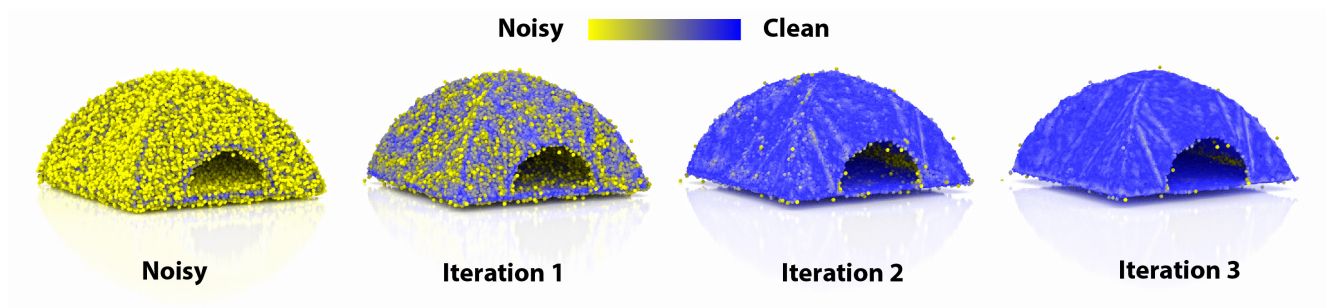


Figure 10. Resulting point cloud of applying our network several times iteratively on a noisy point cloud.

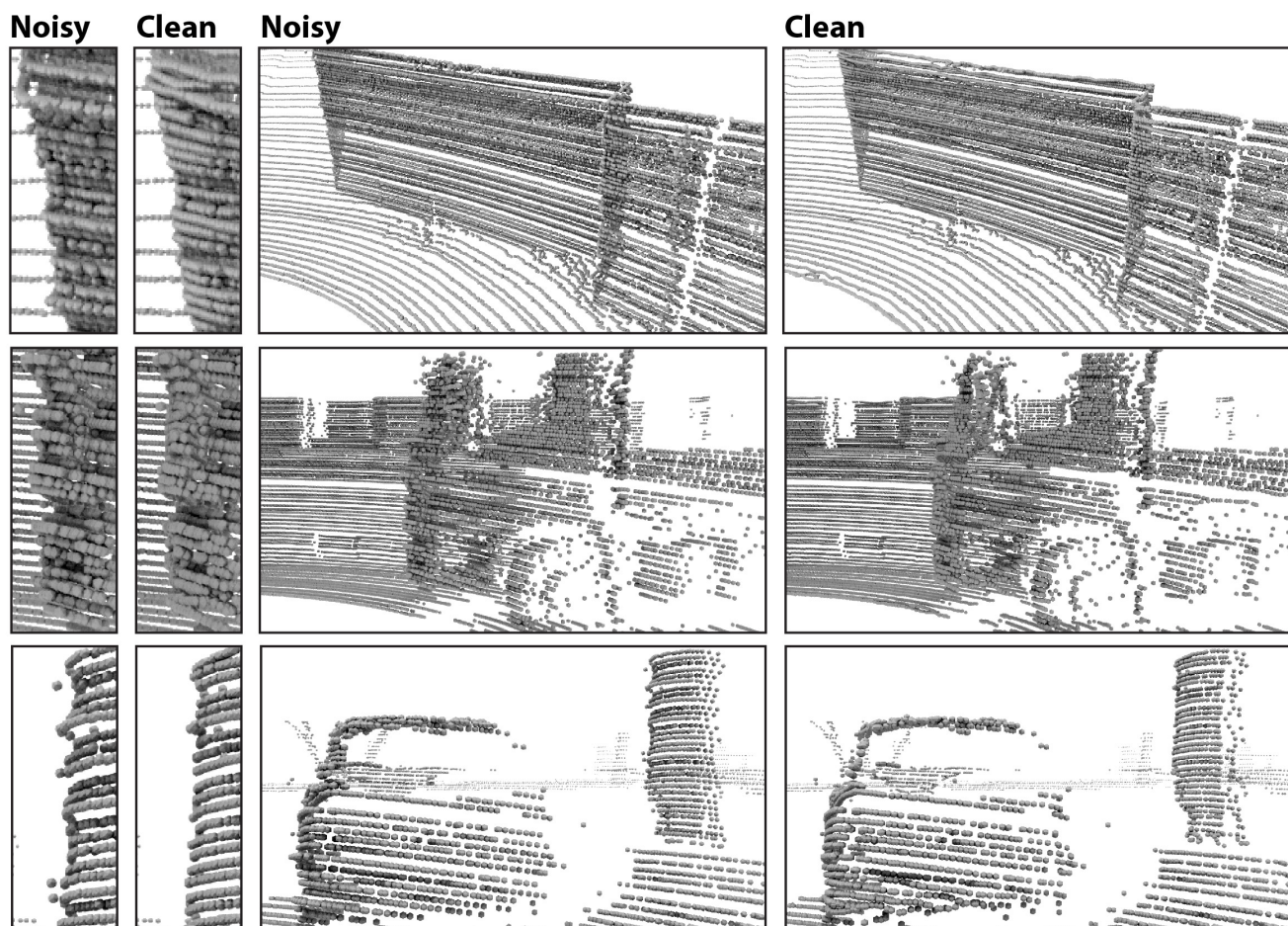


Figure 11. Qualitative results of our network trained on the Kitti dataset [9]. Note how our network is able to remove the noise in sparse point clouds as the one illustrated in the bottom image.

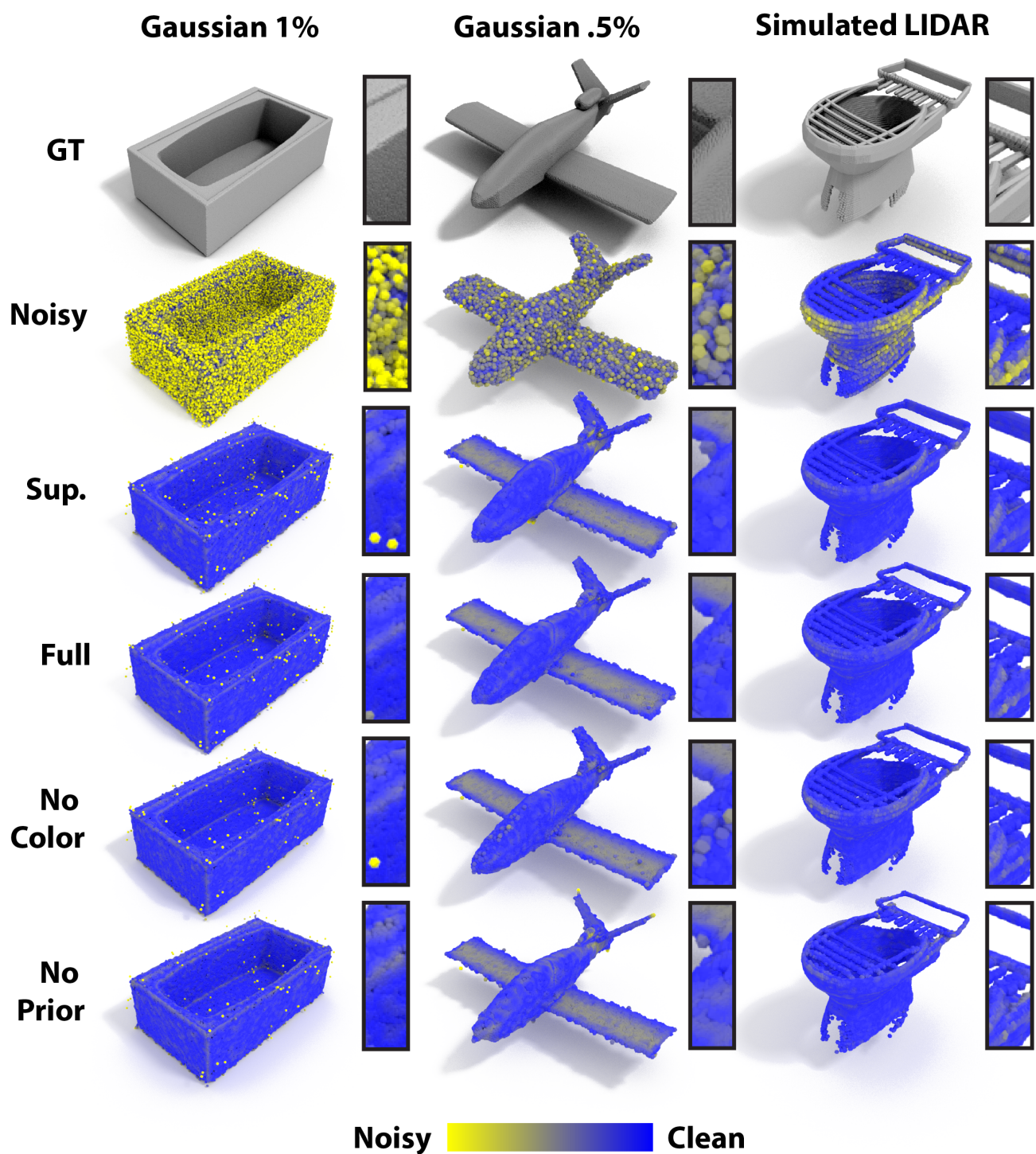
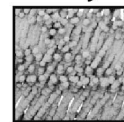


Figure 12. Visual encoding of the distance from each point to the underlying ground truth surface. Here blue indicates zero distance to the surface and yellow 2% of the diagonal of the model. We can see that for the two simulated datasets with different levels of noise, our algorithm achieves (**Full**) almost the same quality as a network trained with supervised data (**Sup.**). Moreover, we evaluated two variants of our training procedure: a prior without color information (**No Color**), and **No Prior**.

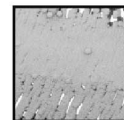
Noisy

Clean

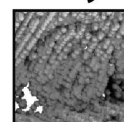
Noisy



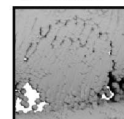
Clean



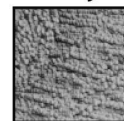
Noisy



Clean



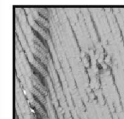
Noisy



Clean



Noisy



Clean

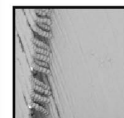


Figure 13. Comparison of our denoising algorithm (**right**) with the noisy scanned data (**left**) from the Paris-rue-Madame Database [24]. The number of points for each model are, from top to bottom, 800 K, 450 K, 450 K, and 375 K points. Our network is able to process each model in parallel in a workstation with a single Nvidia RTX 2080.