Structured Low-Rank Matrix Factorization for Point-Cloud Denoising

Kripasindhu Sarkar[1,2], Florian Bernard[3,4], Kiran Varanasi[1], Christian Theobalt[3,4], and Didier Stricker[1,2]

[1]DFKI Kaiserslautern, [2]TU Kaiserslautern, [3]MPI Informatics, [4]Saarland Informatics Campus

Abstract

In this work we address the problem of point-cloud denoising, where we assume that a given point-cloud comprises (noisy) points that were sampled from an underlying surface that is to be denoised. We phrase the point-cloud denoising problem in terms of a dictionary learning framework. To this end, for a given point-cloud we (robustly) extract planar patches covering the entire point-cloud, where each patch contains a (noisy) description of the local structure of the underlying surface. Based on the general assumption that many of the local patches (in the noise-free point-cloud) contain redundant information (e.g. due to smoothness of the surface, or due to repetitive structures), we find a low-dimensional affine subspace that (approximately) explains the extracted (noisy) patches. Computationally, this is achieved by solving a structured low-rank matrix factorization problem, where we impose smoothness on the patch dictionary and sparsity on the coefficients. We experimentally demonstrate that our method outperforms existing denoising approaches in various noise scenarios.

1. Introduction

Point-clouds are a ubiquitous representation of 3D data that are obtained as the output of a 3D scanner (e.g. range scanners, Kinect etc.), or are the end-result of reconstruction algorithms (e.g. structure-from-motion, or KinectFusion [30]). A well-known issue with sensor-based data acquisition is that the obtained measurements may be noisy, so that they do not faithfully reproduce the measured real-world object. In order to deal with this, various denoising methods are designed for different kinds of data. A common assumption of many methods is that one deals with Gaussian noise—an assumption which does not hold in most real-world applications. Moreover, albeit the fact that point-clouds describe geometric data, they lack a regular structure or topology, which makes their denoising more challenging than for structured data that is defined over a regular grid, such as images.

Learning-based denoising methods have been demonstrated to be extremely successful in numerous settings, particularly for image data [17]. Many of them are based on convolutional neural networks (CNNs) (e.g., [22, 43]). They require that the data that is to be processed is amenable to convolution operations, i.e. the data has a well-defined topology which ideally is regular. Since images exhibit these desirable properties, applying convolutions to images is straightforward. However, defining such operations on general geometrical objects is more challenging and is an active area of research (*e.g* for meshes [8, 29] or for point-clouds [32, 33, 5]). Due to the lack of a well-defined topology, applying such techniques to point-clouds is difficult, so that often the spatial neighborhood information is not used.

In contrast, we propose a simple method that exploits the spatial neighborhood in point-clouds and does not require any training data. We perform point-cloud denoising based on a matrix factorization formulation, where we pay particular attention to finding a suitable *fixed-length and regular representation* of the point-cloud, such that the above-described problems regarding irregular structure and the lack of a topology are circumvented. To this end, similarly to [16, 36], we represent a given point-cloud by a collection of (local) displacement maps, each of which encode the local structure of the point-cloud within a small patch. Our novel contribution is a strategy for extracting such patches that is robust to noise. In contrast to the prior works, we do not require the patches to be too small [16], or require a semi-automatic mesh quadrangulation step [36]. We learn a dictionary that is able to denoise the extracted displacement maps, such that the denoised point-cloud is obtained from the reconstruction of the denoised local patches. An important and desirable property of our approach is that it works under a wide range of different noise characteristics, as the dictionary learning procedure implicitly deals with the noise model present in the data.

Main Contributions: The technical contributions of our method are as follows:

- Given an input 3D point-cloud, we present a method to *robustly* extract a collection of patches covering the entire point-cloud, where each patch describes the local structure. Unlike previous works, we do not require the patches to be exceedingly small [16] or require an explicit meshing and quadrangulation step [36].

- We propose a formulation of *structured matrix factorization* for these 3D patches, by extending it to handle missing data, where the patches may contain unoccupied bins.
- We handle a wide range of noise characteristics including real world 3D scanner noise, unlike previous methods which are limited to normally distributed noise. We demonstrate superior denoising results as compared to other methods.

2. Related Work

Mesh Denoising: The problem of denoising is directly connected to building a signal processing framework over a domain on which the signal is defined [38, 39]. Zhong et al. [44] propose a method for 3D shape completion by imposing sparsity on the Laplacian eigenbasis of the shape. The more recent approaches for denoising use machine learning models such as CNNs that are trained on a pre-registered dataset of meshes [9]. However, to be effective, they require training data of meshes that are pre-registered to a common topology, which is a challenging problem in its own right.

Global models for point-cloud denoising: The alternative is to consider the 3D point-clouds directly, with no structure and grid pattern given for the domain on which this signal is defined. Unlike with the mesh input, these methods cannot make use of a surface normal for denoising. Nevertheless, it is possible to learn denoising models from unstructured point-clouds [32, 33, 1]. These models are sensitive to a global transformation of the point-cloud, so the 3D alignment problem needs to be addressed by a dedicated spatial transformer network [32], which can be further enhanced by learning local features [33]. Despite their versatility, these methods ignore the fact that most real world 3D point-clouds are sampled from objects which can be represented as manifold surfaces at the local level. Alternatively, a dynamic graph CNN model can be learned that adapts the topology over time [40], but this requires a dynamic graph update of a global model. In contrast to these point-based methods which try to estimate a global model for the entire point-cloud, our method estimates only local models (patches) around points, which are simpler to learn. Moreover, local patch models are well-suited for denoising, as noise affects the shape in a local manner.

Local methods for point-cloud denoising: A good strategy of denoising point-clouds is to fit a smooth target surface using the local neighborhood and project points on this smooth surface for obtaining the denoised point-cloud [21, 14, 24, 28, 3, 4]. Out of them, the most popular strategy is the use of a weighted least squares measure biased towards the local point of computation to fit a continuous function—also known as Moving Least Squares methods [3, 4, 21, 24]. On a similar line, [13, 15] use Voronoi-based computations for the surface reconstruction, while [10] uses octree decomposition followed by a modified 'meshless' Laplacian smoothing for the purpose of denoising. All these methods perform very well on denoising point-clouds, but often remove important details of the underlying surface. This is mostly because of the local nature of the problem being solved, which does not consider the information coming from different regions of the surface. This motivates us to use a non-local method on local feature descriptors (patches), which is further discussed in the subsequent paragraphs.

Patch-based methods: Patch-based methods have been developed first for 2D image processing, where it is assumed that the 2D image has a locally sparse representation in the transformed domain. These algorithms can be categorised into dictionary based [2, 27, 26] and BM3D (Block-matching and 3D filtering) based [11, 12, 25] methods. In BM3D, similar 2D patches are further grouped into 3D groups, which enhances sparsity. Rosman et al. [34] apply this idea towards the denoising of 3D shapes by matching the patches on a *collaborative patch* representation. Our method does not require this *a priori* matching step.

Digne et al. [16] proposed a method for point-cloud compression by learning a dictionary of circular patches. They assume that the local patches are sufficiently small (where the shape is parameterizable to a unit disc). In contrast to this work, (i) we address the problem of denoising instead of compression which makes the patch extraction procedure more challenging, (ii) our local reference frame for patches is robust to noise and outliers (in comparison to simple PCA), (iii) our dictionary learning framework handles missing data, and (iv) as a result of aforementioned differences, our patch size is much larger in order to have a meaningful patch description in the presence of noise.

Sarkar et al. [36] are able to address larger patches (in comparison to [16]), by requiring a 3D mesh input along with the point-cloud, and computing local patches based on quadrangulation of its low resolution mesh. Their method learns a sparse dictionary for 3D shape completion and denoising. In a later work [37], they learn a denoising autoencoder neural network model by relying on the same mesh quadrangulation for computing and orienting the patches. Both of these approaches are limited by the assumption of mesh quadrangulation, which may not be possible to compute on general point-clouds.

A fundamental problem with applying patch-based methods to point-clouds is that the patch placement and orientation can be ambiguous and cannot always be judged based on the local structure of the point-cloud alone. The previous approaches worked around this problem by assum-

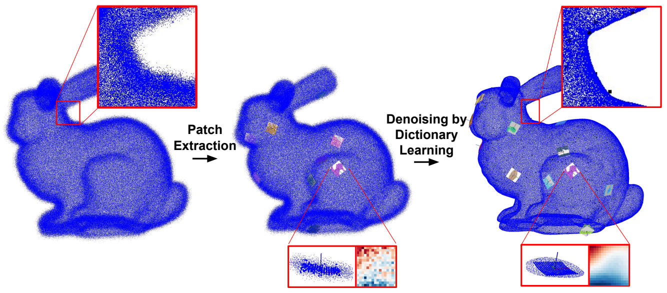

Figure 1. Overview of our pipeline. Given a noisy point-cloud (left), we first extract local patches that cover the entire point-cloud (center), where each local patch encodes a displacement map (for illustration purposes we only show a few patches). Subsequently, we denoise the displacement maps in order to obtain the clean displacement maps based on dictionary learning, as illustrated in Fig. 3. After reconstructing the denoised point-cloud based on the clean displacement maps, we obtain the clean point-cloud (right).

ing the patches to be very small, or requiring a prior matching step, or a mesh quadrangulation step. In this paper, we propose an alternative by the combination of (i) robust patch computation and (ii) structured matrix factorization that elegantly handles missing data.

3. Point-Cloud Denoising

Overview: We consider the task of denoising a point-cloud that comprises points that are noisy samples of an (unknown) surface. In order to process a given point-cloud, we represent the entire point-cloud by means of (small) planar patches, where each patch describes the local structure of the point-cloud in terms of a displacement along the patch normal. By associating with each patch a rigid-body pose, i.e. a location and an orientation, one can reconstruct the point-cloud from its patch-based representation. As such, in order to perform denoising, we first extract patches from the noisy point-cloud, we then denoise the patches using dictionary learning, and eventually reconstruct the point-cloud based on the denoised patches (Fig. 1).

3.1. Notation

For a given integer $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. Let A be an $m \times n$ matrix, and let $i \in [n]$ and $j \in [m]$. By $A_{:,i}$ we denote the m-dimensional column vector that is formed by the i-th column of A, for which we also use the short-hand A_i. Moreover, we denote by $A_{j,:}$ the n-dimensional row vector that is formed by the j-th row of A. The operator $\odot$ denotes the Hadamard product. For a point $x \in \mathbb{R}^3$, we use $\mathcal{N}_\epsilon(x) := \{y \ : \ \|x - y\| < \epsilon\}$ to denote the ϵ-neighborhood (or ϵ-ball) of x, where $\|\cdot\|$ is the ℓ_2-norm. For a given matrix $Z \in \mathbb{R}^{3\times z}$ containing z points in 3D, we define $\mathcal{N}_\epsilon^Z(x) := \{Z_i \ : \ i \in [z], \|x - Z_i\| < \epsilon\}$ to denote the ϵ-neighborhood of x among the points in Z.

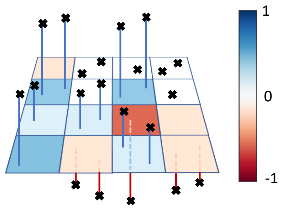

Figure 2. Illustration of a displacement map that is defined over a square patch. The displacement values are colour-coded, as shown in the legend. The points of the point-cloud are shown as black crosses. The position of each point is described relative to an anchor point on the patch by means of a displacement along the patch normal.

3.2. Patch-based Point-Cloud Representation

In this section we describe the patch-based point-cloud description, which is also illustrated in Fig. 2. Let $X \in \mathbb{R}^{3\times n}$ be a 3D point-cloud comprising n points, which we assume to be (noisy) samples of some (unknown) underlying surface. For the patch-based point-cloud representation, we consider a collection of planar patches that are evenly distributed so that they cover the entire point-cloud. Each patch is oriented such that, ideally, the patch normal and the normal of the underlying surface coincide.

3.2.1 Seed Point Selection

We select a subset of points from the point-cloud X, written in the matrix $S \in \mathbb{R}^{3\times s}$, which forms an evenly distributed and sparse representation of X. The s points of S are used as *seed points* for the computation of the patches, which is based on voxel-based downsampling method with grid cell length of d. In total, this results in $s \ll n$ seed points, which are then stored in the matrix S. The parameter d implicitly controls the density of the patches that are to be computed. It should be chosen (together with the neighborhood radius r) in such a way that all points in the point-cloud are contained in the union of the We r-neighboorhood of the seed points. Formally, this can be expressed as

$$\forall\, j \in [n]\ \exists\, i \in [s] : X_j \in \mathcal{N}_r^X(S_i)\,. \qquad (1)$$

While the condition $d < \sqrt{2}r$ ensures that property (1) holds, in practice we choose d to be have a similar value as r in order to achieve a denser covering with the patches. The parameter choices are explained in Section 4.

3.2.2 Robust Patch Extraction

Next, we describe how to robustly extract patches after the seed points have been selected. For the sake of a simple explanation we use *square* patches (note that disk-shaped patches could also be considered). To this end, for each seed point $S_i \in \mathbb{R}^3$ we extract the patch P_i that covers a square with side length $l = \sqrt{2}r$. The patch $P_i = (t_i, R_i, Y_i)$ is represented as a 3-tuple that describes the local structure of the point-cloud within the r-neighbourhood $\mathcal{N}_r(S_i)$ of S_i. Here, t_i is the location of the center of the patch, R_i is its orientation, and $Y_i \in \mathbb{R}^{m\times m}$ is the *displacement map*. For reasons that will become clear shortly, the seed point location S_i does not necessarily coincide with patch center t_i. A significant advantage of the displacement map representation compared to the original point-cloud is that now the Y_i constitute a *fixed-length* matrix representation of the local surface structure, such that they are amenable to simple linear algebra operations.

Robust selection of orientation and location: In order to obtain a patch that aligns well with the underlying surface—even under significantly noisy point-clouds—we propose to use a robust strategy that makes use of a RANSAC-like sampling strategy [18], which is outlined in Alg. 1. In order to determine whether a point is an inlier for a given patch, we check whether this point is within a given distance to the square *patch*. By considering the *point-to-patch* distance (line 8), in contrast to the *point-to-plane* distance, one can avoid the problem that a patch is fitted in such a way that it cuts through the object that is described by the underlying surface (cf. Fig. 5). After the best patch, i.e. the one that leads to the largest number of inliers, is determined, the patch orientation is computed based on principal component analysis (PCA), and the patch center is set to the mean of the inlier points. If the number of inliers are less than a threshold, the seed point and the neighborhood are discarded, making our method naturally robust to noisy outliers.

Displacement map computation: Given the computed position t_i and the orientation R_i of the patch, we compute the fixed-length displacement map from all neighboring points of the seed point. We represent the r-neighborhood $\mathcal{N}_r^X(S_i)$ of the seed point S_i in the reference frame defined by (t_i, R_i), and project all points onto the square patch, and then sample the patch on an $m \times m$ grid of size $l = \sqrt{2}r$. The displacement value for each bin is then computed as the median of all “z-coordinates” of the points that fall in that particular bin. For all $p, q \in [m]$, the “z-coordinates” denote the length of displacements from the center point $a_{pq}^i \in \mathbb{R}^3$ of the bin at position (p, q) along the patch normal (cf. Fig. 2). In addition, for each point X_j of the point-cloud, we keep track of the information to which patch and to which bin of each displacement map it falls during the patch computation. This correspondence information is used for the reconstruction of the shapes from the patches. In order to keep track of this correspondence information, for each $j \in [n]$ we define the list of $k_j \in \mathbb{N}$ index triplets $I_j \in \mathbb{R}^{3\times k_j}$, where each column $(I_j)_\ell =: [i, p, q]^T \in [s] \times [m] \times [m]$ for $\ell \in [k_j]$ contains the patch index $i \in [s]$ in the first row, and the corresponding bin index $[p, q]^T \in [m] \times [m]$ in the second and third rows. Note that a point X_j can belong to multiple patches

```
Input: point-cloud $X$, seed points $S$, neighborhood $r$, side length $l$
Output: patch orientations $R_1, \dots, R_s$ and locations $t_1, \dots, t_s$
1  foreach $i \in [s]$ do
       // random sampling to find inlier points
2      $\mathcal{I}_i \leftarrow \emptyset$
3      foreach random sample do
4          randomly select three unique points $x, y, z$ from $\mathcal{N}_r^X(S_i)$
5          fit a plane $\mathcal{L}_{\text{put}}$ with normal $n$ through $x, y, z$
6          compute the putative patch center $c_{\text{put}} \leftarrow \frac{1}{3}(x + y + z)$
7          define square patch based on $l$, $\mathcal{L}_{\text{put}}$ and $c_{\text{put}}$
8          compute the set of inlier points $\mathcal{I}_{\text{put}}$ (point-to-patch
             distance)
9          if $|\mathcal{I}_{put}| > |\mathcal{I}_i|$ then
10             $\mathcal{I}_i \leftarrow \mathcal{I}_{\text{put}}$
       // determine the orientation
11     obtain orientation $R_i$ based on PCA of $\mathcal{I}_i$
       // determine the patch center
12     $t_i \leftarrow \text{mean}(\mathcal{I}_i)$
```
Algorithm 1: Algorithm for robust patch extraction.

so that $k_j \geq 1$. In overall, given the point-cloud X as an input, along with the parameters d, r and m, we compute the patch set $\{P_i \ : \ i \in [s]\}$, and the set of correspondence information $\{I_j \ : \ j \in [n]\}$. Note that some of the patch bins may be unoccupied, i.e. there is no point X_j that falls into such a bin, which particularly occurs in areas of edges and corners.

3.2.3 Point-Cloud Reconstruction

Next, we briefly describe how we can reconstruct the point-cloud given the patch set $\{P_i = (t_i, R_i, Y_i) \ : \ i \in [s]\}$ and the set of correspondence information $\{I_j \ : \ j \in [n]\}$. To this end, for all m^2 bins we compute the global bin position $a_{pq}^i \in \mathbb{R}^3$ in all s displacement maps, which we also refer to as *anchor points*. Then, we obtain each of the $\ell \in [k_j]$ reconstructed points as $a_{pq}^i + (Y_i)_{pq} n_i$, where $n_i \in \mathbb{R}^3$ is the normal of the i-th patch (which is determined by R_i), and $[i, p, q]^T = (I_j)_\ell$ for all $\ell \in [k_j]$.

3.3. Denoising by Dictionary Learning

In this section we describe the dictionary learning procedure that is used for denoising the collection of displacement maps $\mathcal{Y} = \{Y_1, \dots, Y_s\}$. On overview of the approach is shown in Fig. 3. A theoretical motivation for using such formulation is provided in the *supplementary material*.

General matrix factorization: Let $y_i := \text{vec}(Y_i) \in \mathbb{R}^{m^2}$ be the vectorization of Y_i, and let $\mathbf{Y} = [y_1, \dots, y_s] \in \mathbb{R}^{m^2 \times s}$ be the matrix that contains the vectorized displacement maps. With that, finding a low-dimensional subspace that is a good approximation of the noisy $\mathbf{Y}$ can be phrased as the general matrix factorization problem

$$\min_{\Phi, A} \ \ell(\mathbf{Y}, \Phi A) + \Omega(\Phi, A)\,, \tag{2}$$

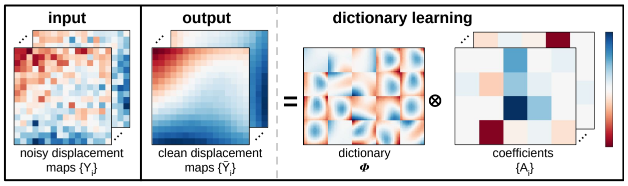

Figure 3. Overview of the dictionary learning method. Given noisy displacement maps $\{Y_i\}$ as input, the clean displacement maps $\{\bar{Y}_i\}$ are obtained by learning a sparse dictionary, such that the clean displacement maps $\bar{Y}_i$ are given by a weighted sum of the dictionary atoms Φ based on the coefficients $\{A_i\}$.

where $\Phi \in \mathbb{R}^{m^2 \times r}$ is the *factor matrix* that contains r dictionary atoms in its columns, and $A \in \mathbb{R}^{r \times s}$ contains the *coefficients* in order to reconstruct $\mathbf{Y}$ using the dictionary Φ. Here, the function $\ell(\cdot)$ is the loss function that measures how well the factorization ΦA approximates the given $\mathbf{Y}$, and $\Omega(\cdot)$ is a regularizer that has the purpose to impose desirable properties upon the factors Φ and the coefficients A. In the following we will further specify the loss function and the regularizer.

Matrix Factorization for Denoising: Due to appealing theoretical properties regarding optimality guarantees, as well as a promising performance in various applications (e.g. [6]), for tackling Problem (2) we build upon the *structured low-rank matrix factorization* framework by Haeffele et al. [20]. For the sake of notational convenience, and w.l.o.g., we assume that $\mathbf{Y}\mathbf{1}_s = \mathbf{0}_{m^2}$, i.e. the column-mean of $\mathbf{Y}$ is zero (if this is not the case, we simply subtract the column-mean from $\mathbf{Y}$). Moreover, let $M \in \{0,1\}^{m^2 \times s}$ be a binary matrix that masks out unobserved data in the displacement map matrix $\mathbf{Y}$. We define the loss function as

$$\ell(\mathbf{Y}, \Phi A) := \|M \odot (\mathbf{Y} - \Phi A)\|_F^2\,, \qquad (3)$$

such that the error when approximating $\mathbf{Y}$ using the factorization ΦA is measured in a *weighted* least-squares sense. Here, the difference to the work in [20] is that we only compute the least-squares error for those elements in $\mathbf{Y}$ that are available. We emphasize that the possibility of handling missing data is absolutely essential for point-cloud denoising, since, due to edges and corners of the underlying surface, some bins in the displacement maps of the extracted patches may be unoccupied.

The purpose of the regularization term $\Omega(\cdot)$ is twofold: on the one hand, it shall impose sparsity on the coefficients A, such that each y_i is reconstructed from only few dictionary atoms, i.e. the columns $\Phi_1, \dots, \Phi_r$ of Φ. Moreover, in addition to sparse coefficients, we impose spatial smoothness as well as ℓ_2-regularization upon each dictionary atom Φ_i for $i \in [r]$. In the context of learning linear shape deformations with spatially localized support, a similar set of regularizers has been used by Bernard et al. [6]. Our regularizer is given by

$$\Omega(\Phi, A) = \lambda \sum_{i=1}^{r} \|\Phi_i\|_\phi \|A_{i,:}^T\|_a\,, \qquad (4)$$

where for $y \in \mathbb{R}^{m^2}$ and $z \in \mathbb{R}^s$ the norms $\|\cdot\|_\phi$ and $\|\cdot\|_a$ are defined as

$$\|y\|_\phi = \lambda_2\|y\|_2 + \lambda_E\|Ey\|_2, \text{ and} \qquad (5)$$

$$\|z\|_a = \lambda_1\|z\|_1\,. \qquad (6)$$

The scalars $\lambda, \lambda_2, \lambda_E$ and λ_1 are non-negative weights that are used to specify the relative importance of the overall regularization term, the ℓ_2-norm of the factors, the smoothness of the factors, and the sparsity of the coefficients, respectively. The matrix E is the incidence matrix of the 4-neighbourhood graph of the $m \times m$ grid of the patch, such that $\|E \cdot\|_2$ can be seen as a graph-based (semi)-norm that takes the spatial connectivity of the patch bins into account. Its main purpose is to ensure that the learned dictionary atoms $\Phi_1, \dots, \Phi_r$ are spatially smooth. We point out that if spatial smoothness is undesirable, one can set λ_E=0.

Once we have found Φ and A, we obtain the estimated matrix of clean displacement maps as $\bar{\mathbf{Y}} = \Phi A \in \mathbb{R}^{m^2 \times s}$, which are then used to reconstruct the denoised point-cloud. Please refer to the *supplementary material* for the motivation for using this particular form of the regularizer in (4).

Optimization: In order to optimize Problem (2), we use a block-coordinate descent approach [20]. To this end, the variables Φ and A are updated alternatingly, where for each variable a gradient descent step of the loss function is conducted first, followed by a proximal step for the regularizer. The computation of the proximal operators of the regularizer is described in [6].

4. Experiments

4.1. General Denoising Settings

Dataset: For evaluating our denoising algorithm, we use the models from [36], which include common meshes like *Bunny*, *Fandisk*, *Milk Bottle*, *Baseball* and several meshes with fine surface detail such as shoe soles (*Supernova*, *Terrex*, *Wander*, *Leather-Shoe*) and *Brain*.

Patch computation: To work with similar parameters across all models, we scale the models so that the bounding box corresponds to a unit cube and sample 1000k points from the surface and add different types of noise. We use a neighborhood radius $r = 0.03$ (so that the patch length is $l \approx 0.042$) and $m = 16$ as patch dimension for all the models. The value for the radius was selected in such a way that the patches are significantly larger compared to the noise level (e.g. Gaussian noise with $\sigma = 0.0075$). Also we have chosen the seed radius d the same as r to obtain overlapping patches. Ideally, the patch length should be chosen such that the r-neighbourhood contains only points within a topological disk on the surface, which restricts it to be less than the

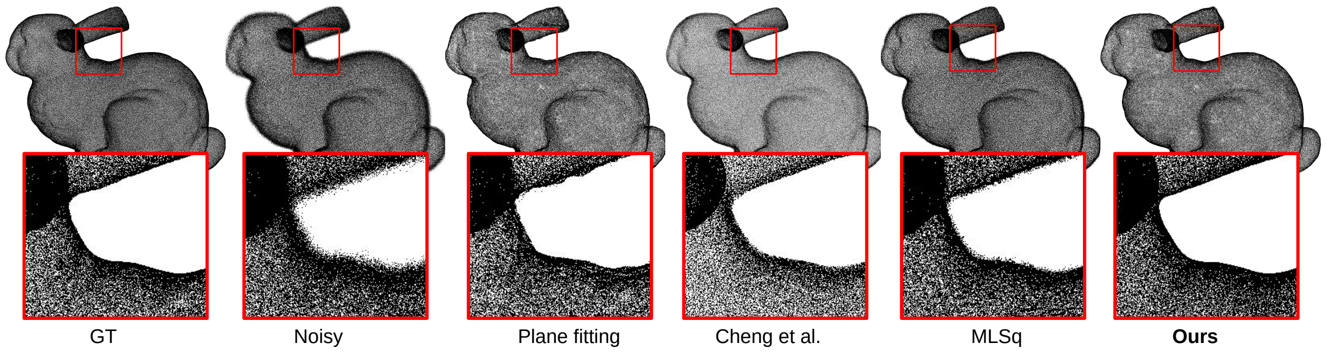

Figure 4. Qualitative results of our method and a comparison with other denoising methods under Gaussian noise. From left to right we show the following point-clouds: ground truth, noisy, denoised by plane-based method, denoised by [10], denoised by [4], and denoised by our method. In the magnification of the high-curvature ear region of the bunny it can be seen that qualitatively our denoising method achieves the sharpest results that appears closest to the ground truth. In addition, as shown in Table 1, quantitatively our method outperforms the other approaches.

noisy	plane	[10]	[4]	Ours
4.93e-03	3.21e-03	6.46e-03	2.40e-03	**1.60e-03**

Table 1. Summary of the results of denoising 9 shapes (Section 4.1 - Eg. Bunny, Fandisk, Supernova, etc.). The values represent the mean of RMS point-to-mesh error (relative to bounding box) of different algorithms for Gaussian noise with $\sigma = 0.0025$, 0.005 and 0.075, for 10 shapes. For exact result of each shape with different noises, please refer to the supplementary document. A qualitative result of one such shape (Bunny) is provided in Figure 4.

distance between surface points and the shape medial axis. However, because of our robust patch fitting and the handling of missing data during matrix factorization, we can generously extend this limit. We have empirically found the chosen radius of $r = 0.03$ to perform well and thus we used it in all the experiments with the meshes from [36].

Matrix factorization: For all point-clouds we normalize the parameters of the matrix factorization such that they are invariant to the problem size. With that, we set the parameters to $\lambda = 6.4\frac{m^2 s}{r}, \lambda_2 = 10^{-5}\frac{1}{m^2}, \lambda_E = 2{\cdot}10^{-5}\frac{1}{m^2}$ and $\lambda_1 = 45\frac{1}{s}$. The number of dictionary atoms r is set to 50 for all the point-clouds. Figure 4 shows the qualitative results of our method under Gaussian noise. Experiments are described later with more details.

4.2. Evaluation of Design Choices

Robust patch computation: Fig. 5 illustrates, qualitatively, how our patch computation method outperforms both (i) simple PCA based approach (left), and (ii) RANSAC plane fitting approach (middle). Obtaining local reference frames for patches using simple PCA results in considering the entire cloud neighborhood which does not align the patch well to the underlying surface in noisy and sharp neighborhoods, whereas our approach aligns it well. Also, robust *plane fitting* can accommodate outliers far away from the patch region which is not desired. Our method is resistant to both of the unwanted characteristics.

When the amount of noise is small and the patch size

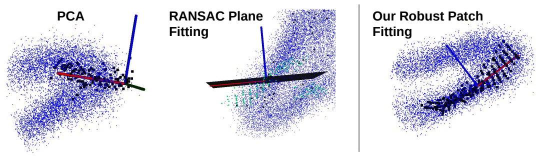

Figure 5. Comparison of patch orientation and location based on PCA (left), RANSAC-based *plane* fitting (right) and our robust patch fitting method (right) on similar structures (ear region of the *bunny* model).

is also sufficiently small, the commonly used PCA-based patch fitting is reasonable. In this case, with a sufficiently large amount of random samples, our method becomes close to such a PCA-based approach, which we also demonstrate experimentally: With an added noise with $\sigma = 0.0025$, for the bunny model the error with the PCA-based patch computation results in an RMS error of 6.61e-04, in comparison to 7.17e-04 with our method. However, for more severe noise or larger patches, the assumption that the points within a neighborhood are (approximately) normally distributed breaks, and hence the PCA-based approach is not suitable anymore. Experimentally, we found that with a larger noise of $\sigma = 0.005$, the error increases to 1.52e-03 with the PCA-based approach, in comparison to 1.15e-03 with our method. Hence, our proposed patch extraction method is particularly well-suited for computing patches in highly noisy areas and complicated shapes.

Evaluation of matrix factorization method: We have also evaluated the matrix factorization method without our generalization to handle missing data, cf. (3). In this case, for learning the dictionary we replace each missing value in the input matrix $\mathbf{Y}$ by the average value taken over all patches. On the bunny model, this results in an RMS of 8.42e-04 for $\sigma = 0.0025$ (compared to 7.17e-04 for our implementation) and for $\sigma = 0.005$ in an error of 1.15e-03 (similar as our implementation 1.15e-03).

Moreover, in Table 1 (and the corresponding table in *supplementary material*) the column 'plane' corresponds to

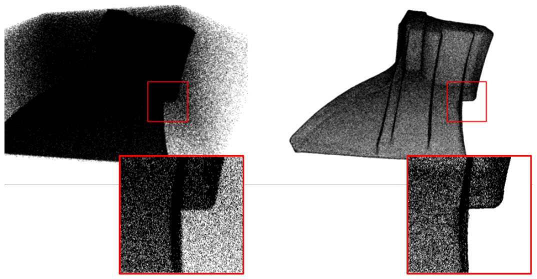

Figure 6. Result of our denoising algorithm with uniform noise (50% of input points) in addition to Gaussian noise ($\sigma = 0.0025$). For the model *Fandisk* the RMS error after denoising is $8.05e{-}04$ (in comparison to $8.12e{-}04$ in the experiment without white noise). Hence, our denoising algorithm is resistant to outliers.

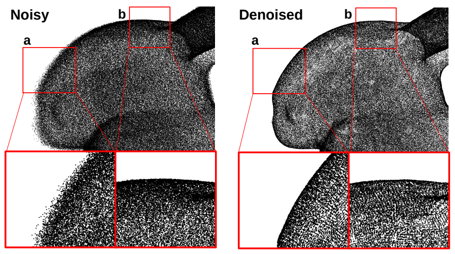

Figure 7. Result of our denoising algorithm when the noise is added in the direction of camera. The left part shows the noisy data and the right part shows the denoised output. It can be seen that our method consistently removes the noise from the surface perpendicular to the camera direction (where the maximum amount of noise occurs, see **region a**) and that it does not alter much when the surface is parallel to the camera direction (places where the surface is mostly unaffected by the added noise, see **region b**).

simply running our robust patch extraction method without any dictionary learning. It can be understood as fixing the displacement maps to contain the constant values 0. When comparing this to the result of running our full approach (with matrix factorization), we can see a significant improvement when using the matrix factorization.

4.3. Comparison to Other Methods

To get a better understanding of the denoising performance of our method, we perform extensive experiments with the simple i.i.d. Gaussian noise in different point-clouds, and compare our results with the denoising algorithm of (i) Cheng et al. [10], (ii) moving least-squares [4], and (iii) the plane-based approach described in the previous paragraph (i.e. our method without matrix factorization). For [4] we use the implementation in Point Cloud Library (PCL) [35]. The recent work of [10] uses octree decomposition followed by a modified 'meshless' Laplacian smoothing and claim to be better, in terms of details retained, as compared to Voronoi based algorithms such as Cocone and RobustCocone [13, 15]. We could not compare our results with another non-local based method [34] because of the lack of the availability of the source code.

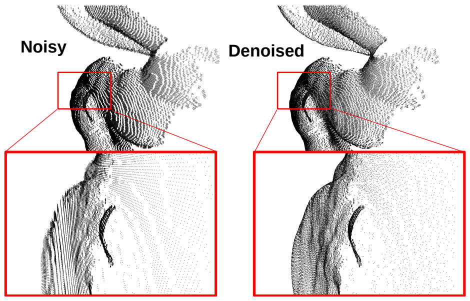

Figure 8. Top: Denoising of a point-cloud with noise obtained by a Kinect simulator [7] (parameters: $r = 0.05$, $m = 16$). Note the removal of typical fringe patterns in both the raw depth scan (zoomed version). See supplementary material for a larger image.

Since points are sampled from the meshes, we use Root Mean Square (RMS) point-to-mesh error (relative to the bounding box as evaluation metric, which is reported in Table 1 (and the corresponding table in *supplementary material*). It can clearly be seen that quantitatively our method performs better than both [10] and [4] on both the set of common models (*Bunny*, *Fandisk*, *Milk bottle*, etc.) and the shoe soles model with high surface detail. A visual comparison of the results is provided in Figure 4.

Uniform noise: Due to the employed robust patch fitting procedure, our method naturally deals with outliers without the requirement of any additional preprocessing or outlier removal steps. We perform further experiments to validate this, where, in addition to Gaussian noise, we randomly add a total number of 50% of the initial number of points from a uniform distribution to the point-cloud (within the bounding box). We run our denoising algorithm on such noisy data and provide our results in Figure 6. Experimentally we have found that adding a significant amount of uniformly distributed outliers does not impact our method.

4.4. Evaluation of Different Noise Models

Many of the existing point-cloud denoising techniques are closely tied to a reconstruction algorithm itself [42, 41], where the main objective is to achieve a better reconstruction. In contrast, in this work we provide a generic method for processing and denoising point-clouds. Hence, in order to get a better understanding of our algorithm with respect to these regards, in the previous sections we have extensively evaluated it based on i.i.d. Gaussian noise and compared it with other methods that solve the same problem. To complement these evaluations, in this section we provide additional experiments with different types of noise in order to see how well our method generalizes to varying scenarios. Here, our aim is to analyze the general performance of our method under different types of noise commonly observed in real-world applications (e.g. data of different scanner types).

Noise perpendicular to the surface: In this experiment

	noisy	[10]	[4]	Ours
Bunny	4.99e-03	1.19e-01	2.56e-03	**1.28e-03**
Fandisk	4.94e-03	7.76e-02	2.55e-03	**1.45e-03**

	noisy	[10]	[4]	Ours
Bunny	1.71e-03	8.92e-04	7.49e-04	**4.68e-04**
Fandisk	2.08e-03	9.51e-04	1.31e-03	**5.66e-04**

Table 2. Comparison of RMS error (relative to bounding box) for denoising methods under different noise models. Left: Noise perpendicular to the surface. Right: Noise in the direction of the camera.

we add i.i.d. Gaussian noise in the direction of the surface normal, and then use our method to denoise the noisy point-cloud. Since our method denoises values that are displacement maps along the normal of the computed patch center, this can be considered as the simplest noise model for our method. The quantitative result for this noise model with $\sigma = 0.005$ is provided in Table 2 (left).

Noise towards camera: Analyzing the noise that arises in Kinect scans is an important topic of research in the vision community because of the immense popularity of Kinect for capturing depth maps [31, 23]. It has been found that in Kinect data the axial noise (noise in the depth measurement towards the camera direction) varies significantly and to a large extent with the depth, in comparison to the lateral noise (noise perpendicular to the camera direction) [31]. Motivated by this we investigate a noise model where we add i.i.d. Gaussian noise in the direction of a virtual camera. To make the setup more realistic we scale the shape *Bunny* and *Fandisk* so that they have a bounding box of size 50cm, and place the virtual camera at 1.5 meters from the center. Following [31], we added Gaussian noise of standard deviation $\sigma = 0.003$ meters in the direction of the camera to all the points. We apply our denoising algorithm along with the two other methods and show the quantitative and qualitative results in Table 2 (right) and Figure 7 respectively.

Kinect simulator: Directional noise provides a good approximation of the axial noise from Kinect depth maps, but it does not provide several other typical artifacts of Kinect, e.g. occlusion boundaries, noise due to distance between IR projector and IR camera, projection patterns etc. For this we use the Kinect simulator provided by [7] and produce a point-cloud for the bunny mesh. We then use our algorithm for denoising and show the result in Figure 8. Even though this noise is far from Gaussian, our method handles it well.

4.5. Real-World Applications

Denoising point-clouds from a range scanner: We use our method to denoise the data obtained from a structured light scanner (provided by [34]), for which we show qualitative results in Figure 9. It can be seen that the result of our method visually looks better compared to the other methods and the input raw point-cloud.

Denoising for reconstruction: Denoising methods are often targeted specifically to improve the quality of reconstruction from a noisy point-cloud [10]. Hence, in this experiment we reconstruct a point-cloud after denoising with our method and show the qualitative result in Figure 10. As a proof-of-concept, these results show that our method can also be used as a preprocessing step for increasing the quality of reconstruction.

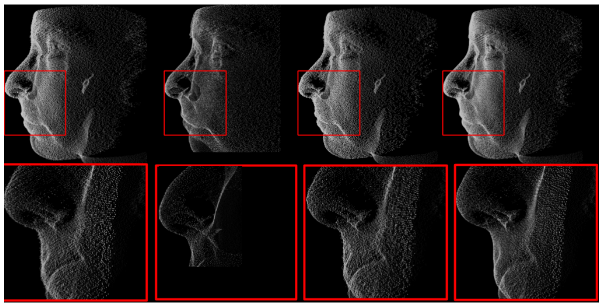
Figure 9. Denoising result from a range scanner. From left to right we show raw scanned data, denoising results from Cheng et al. [10], MLS [4] and ours. Parameters: r=0.25, m=16. A zoomed version is provided in supplementary material.

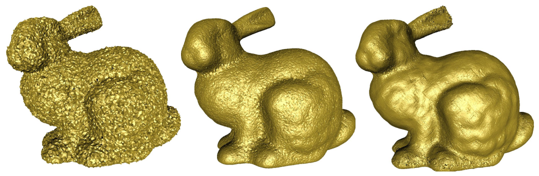
Figure 10. Qualitative reconstruction result by the Ball Pivoting method implemented in [19] for noisy cloud (Left), denoised cloud by [10] (Middle) and denoised cloud from our method (Right). It can be seen that our method preserves surface details very well.

5. Conclusion

In this work we have presented a method for denoising point-clouds based on dictionary learning. To this end, we robustly extract local planar patches (that represent fixed-length displacement maps) from a given point-cloud. In order to denoise the point-cloud, we denoise the displacement maps based on dictionary learning, and then reconstruct the point-cloud from the denoised patches. Our dictionary learning generalizes the recent structured low-rank matrix factorization [20] so that it can also handle displacement maps with missing data, which is crucial for the proposed denoising method. A particular strength of our method is that it learns a new point-cloud-specific dictionary fo each denoising task. It therefore works under a wide range of different types of noise, which we have also confirmed experimentally. Overall, we have shown that our point-cloud denoising framework handles various common types of noise better then previous methods. In the future, we also plan to explore further point-cloud processing tasks within this framework, such as point-cloud inpainting.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. Representation learning and adversarial generation of 3d point clouds. *CoRR*, abs/1707.02392, 2017. 2

[2] M. Aharon, M. Elad, and A. Bruckstein. K -svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, Nov 2006. 2

[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society. 2

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, Jan 2003. 2, 6, 7, 8

[5] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018. 1

[6] F. Bernard, P. Gemmar, F. Hertel, J. Goncalves, and J. Thunberg. Linear shape deformation models with local support using graph-based structured matrix factorisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5629–5638, 2016. 5

[7] J. Bohg, J. Romero, A. Herzog, and S. Schaal. Robot arm pose estimation through pixel-wise part classification. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3143–3150, May 2014. 7, 8

[8] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016. 1

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *Arxiv Preprint arxiv:1611:08097 (IEEE Signal Processing Magazine)*, 2017. 2

[10] S. Cheng and M. Lau. Denoising a point cloud for surface reconstruction. *CoRR*, abs/1704.04038, 2017. 2, 6, 7, 8

[11] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 2

[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Bm3d image denoising with shape-adaptive principal component analysis. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009. 2

[13] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Computational Geometry*, 35(1):124 – 141, 2006. Special Issue on the 20th ACM Symposium on Computational Geometry. 2, 7

[14] T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association. 2

[15] T. K. Dey and L. Wang. Voronoi-based feature curves extraction for sampled singular surfaces. *Computers & Graphics*, 37(6):659–668, 2013. 2, 7

[16] J. Digne, R. Chaine, and S. Valette. Self-similarity for accurate compression of point sampled surfaces. *Computer Graphics Forum*, 33(2):155–164, 2014. 1, 2

[17] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006. 1

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987. 4

[19] L. Giaccari. Surface reconstruction toolbox 0.2, 2011. 8

[20] B. Haeffele, E. Young, and R. Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 2007–2015, 2014. 5, 8

[21] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. 28, 12 2009. 2

[22] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009. 1

[23] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012. 8

[24] R. Kolluri. Provably good moving least squares. *ACM Trans. Algorithms*, 4(2):18:1–18:25, May 2008. 2

[25] M. Lebrun. An analysis and implementation of the bm3d image denoising method. *Image Processing On Line*, 2:175–213, 2012. 2

[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM. 2

[27] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, Jan 2008. 2

[28] B. Mederos, L. Velho, and L. H. de Figueiredo. Robust smoothing of noisy point clouds. In *Proc. SIAM Conference on Geometric Design and Computing*, volume 2004, page 2, 2003. 2

[29] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, volume 1, page 3, 2017. 1

[30] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. 1

[31] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In

2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission, pages 524–530, Oct 2012. 8

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 1(2):4, 2017. 1, 2

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017. 1, 2

[34] G. Rosman, A. Dubrovina, and R. Kimmel. Patch-Collaborative Spectral Point-Cloud Denoising. *Computer Graphics Forum*, 2013. 2, 7, 8

[35] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 7

[36] K. Sarkar, K. Varanasi, and D. Stricker. Learning quadrangulated patches for 3d shape parameterization and completion. In *International Conference on 3D Vision 2017*, 2017. 1, 2, 5, 6

[37] K. Sarkar, K. Varanasi, and D. Stricker. 3d shape processing by convolutional denoising autoencoders on local patches. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. 2

[38] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd International ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 351–358, 1995. 2

[39] B. Valiant and B. Levy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 27(2):251–260, 2008. 2

[40] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. E. Solomon. Dynamic graph cnn for learning on point clouds. *Arxiv preprint arxiv:1801:07829*, 2018. 2

[41] O. Wasenmller, M. Meyer, and D. Stricker. Augmented reality 3d discrepancy check in industrial applications. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–134, Sept 2016. 7

[42] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung, and A. Sorkine-Hornung. Point cloud noise and outlier removal for image-based 3d reconstruction. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 118–127, Oct 2016. 7

[43] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012. 1

[44] M. Zhong and H. Qin. Surface inpainting with sparsity constraints. *Computer Aided Geometric Design*, 41:23 – 35, 2016. 2