EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

PointProNets: Consolidation of Point Clouds with Convolutional Neural Networks

Riccardo Roveri[1], A. Cengiz Öztireli[1], Ioana Pandele[1] and Markus Gross[1]

[1]ETH Zürich

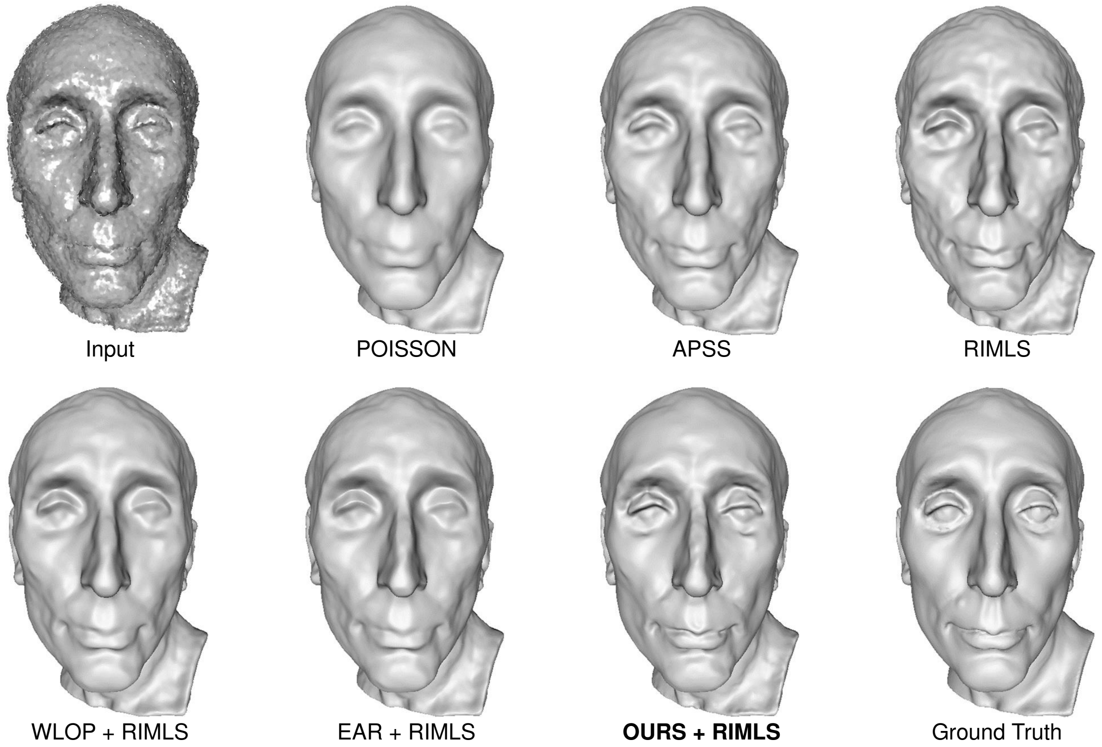

Figure 1: *Surface reconstruction from a noisy and sparse point cloud is an ill-posed problem with infinitely many possible reconstructed surfaces. Our technique consolidates an input point cloud by learning local maps from input to output geometry patches to enhance reconstructions with accurate geometric features and details. This leads to significant improvements for the resulting surfaces.*

Abstract

With the widespread use of 3D acquisition devices, there is an increasing need of consolidating captured noisy and sparse point cloud data for accurate representation of the underlying structures. There are numerous algorithms that rely on a variety of assumptions such as local smoothness to tackle this ill-posed problem. However, such priors lead to loss of important features and geometric detail. Instead, we propose a novel data-driven approach for point cloud consolidation via a convolutional neural network based technique. Our method takes a sparse and noisy point cloud as input, and produces a dense point cloud accurately representing the underlying surface by resolving ambiguities in geometry. The resulting point set can then be used to reconstruct accurate manifold surfaces and estimate surface properties. To achieve this, we propose a generative neural network architecture that can input and output point clouds, unlocking a powerful set of tools from the deep learning literature. We use this architecture to apply convolutional neural networks to local patches of geometry for high quality and efficient point cloud consolidation. This results in significantly more accurate surfaces, as we illustrate with a diversity of examples and comparisons to the state-of-the-art.

CCS Concepts

•***Computing methodologies*** → ***Point-based models;***

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

1. Introduction

Capturing 3D geometries is becoming commonplace thanks to the abundance of affordable and lightweight sensors and advancing algorithms. The captured geometries can then be used for various applications ranging from 3D printing to photography. A main challenge for 3D capture systems, however, is that noise and sparseness in point cloud data typically obscure important geometric features and details. Recovering those details can be very difficult or impossible for many cases.

Given a noisy and sparse point cloud depicting an object boundary, i.e. surface, it is an ill-posed problem to recover such geometric details: there can be infinitely many different geometries that would result in the same sparse and noisy set of sample points. To regularize the problem, we thus need prior beliefs on the global or local structure of the geometry to be reconstructed [BTS*17]. For resolving fine features and details, most methods rely on local priors such as locally piece-wise smooth surfaces with sharp features [OGG09, HWG*13]. This has led to many successful *consolidation*, i.e. synthesizing a new point set that accurately samples the underlying surface, and surface reconstruction algorithms.

Although these techniques generate plausible surfaces, they cannot recover elaborate geometric features if the artifacts in point cloud data become substantial or the priors do not hold. It is in general a very challenging problem to resolve geometric features and up-sample a point cloud especially when only the raw point cloud without further attributes such as surface normals are provided [WHG*15].

In this paper, we propose a data-driven approach to recover surface features and details by building on the recent very successful class of convolutional neural network (CNN) based deep learning methods. CNN-s have shown exceptional performance for many image processing problems, and are more and more used also for generative tasks where an input image is transformed into a new image with desired properties [XRY*15, IZZE16, YZW*16, GCB*17]. However, modern CNN based architectures require a regular sampling of data, and thus extending these techniques to unorganized point clouds is non-trivial [QSMG17], and so far could only be used for coarse shape completion on voxel grids of relatively low resolution [HLH*17].

We tackle this by exploiting the structure of our problem: the geometric features we target are encoded in local regions, which can be individually parametrized. Our method jointly learns local parametrizations and the locally fitted surfaces. We achieve this by developing a new neural network based generative architecture that can consume and output point clouds. This architecture provides an end-to-end approach where an input raw point cloud is used to generate a new very dense point cloud that accurately samples the underlying surface. We show that this leads to substantial improvements in terms of accuracy of the final surface representations. In summary, we have the following main contributions:

- The first deep learning method for local point cloud processing with a fully differentiable architecture that we call PointProNet. A key component in this architecture is a differentiable points projection layer for converting unordered points to regularly sampled height maps. Although we use the architecture for consolidation, it can also be used for revising further point cloud processing tasks
- An end-to-end data-driven algorithm for consolidation of unorganized point clouds that leads to very accurate surface representations, with significant quantitative and visual improvements over the previous methods.

2. Related Work

Consolidation typically involves denoising, resampling, and surface normal estimation, as well as outlier removal and missing data completion. This is then followed by surface reconstruction to get the final surface. Many reconstruction methods can also be used for resampling, and methods that output dense point sets render the reconstruction problem trivial. Hence, our technique is related to both classes of methods. We review the most relevant techniques below.

Consolidation with Smoothness Priors Consolidation and the subsequent task of reconstruction are ill-posed problems and hence further assumptions are required to generate reconstructed surfaces. A very versatile assumption is local smoothness [ABCO*03]. Smooth reconstructions or resampled point sets can be obtained with radial basis functions [CBC*01], solving a Poisson equation in 3D [KBH06], parametrization-free projections [LCOLTE07, HLZ*09, PMA*14], or moving least squares based local approximations [ABCO*03, SOS04, GG07]. The smoothness assumption breaks, however, for certain classes of real-world surfaces that contain sharp features. Many other methods thus focus on preserving such features by utilizing sparsity inducing norms [ASGCO10, SSW15], dictionary learning [XZZ*14], positional constraints [KH13], dedicated sampling of edges [HWG*13], or robust statistics [OGG09, OAG10], leading to significant improvements especially for man-made objects. All these techniques rely on an input point set only, and hence cannot resolve surface shapes if the input point cloud contains a prohibitive amount of imperfection that makes inferring the underlying surface infeasible. We solve this problem by guiding local fits with priors extracted from existing point cloud data of geometries with similar local structures. This learning based approach resolves ambiguities and steers the reconstructions towards accurate local structures.

Data-driven Geometry Completion and Reconstruction When large portions of geometry are missing, several methods use data-driven priors to complete and reconstruct surfaces from point clouds. This can be achieved by retrieving models [SXZ*12, KMHG13, LDGN15] or model parts [GSH*07, SFCH12, SKAG15] from a database that can also be deformed to match the input point clouds [PMG*05, NXS12, KMYG12]. Although such methods excel at global shape completion, resolving geometric features and details can be challenging due to the limited range of compatible objects or parts in the database, wrong matches, and misalignments [HLH*17]. In contrast, we do not require close matches or alignments between training and test geometries, and focus on learning to recover geometric details. Other data-driven methods regress to parameters of a constructed model, which is typically used for e.g. human body shapes [ASK*05, WBLP11]. However, such parametric models lack

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

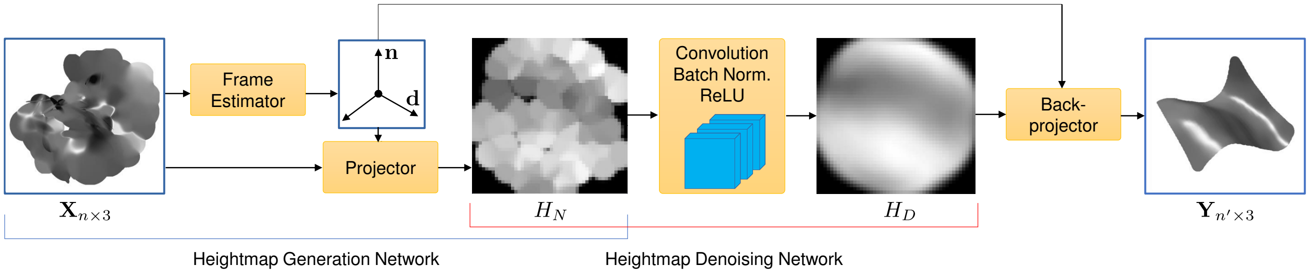

Figure 2: *The network architecture. Each patch* $\mathbf{X}$ *of an input point cloud is processed with this architecture to generate the consolidated point set stored in* $\mathbf{Y}$. *Each component is differentiable and hence allows for end-to-end training.*

the geometric details we target, and are only designed for when the test geometries belong to the specific parametric model constructed.

Geometry Generation and Completion with Deep Learning We propose a new neural network based deep architecture for the consolidation problem. The exceptional performance of deep neural networks on image processing tasks has led to various previous efforts on extending their power to 3D surfaces. Several approaches extend the 2D grids used for image processing to 3D voxels grids [WSK*15, SGF16, VDR*16]. This allows a direct extension of many successful architectures to 3D. However, these only work for relatively low resolution of grids (typically up to 32^3) due to the increased memory and computational requirements in 3D. Even with the state-of-the-art approaches that fuse global and local patches [HLH*17], or utilize octrees [RUG17], the resolution is limited to 256^3. It has thus been so far not possible to directly recover geometric details with the current deep learning architectures [DQN17]. We specifically target such geometric features and details and propose a new dedicated deep architecture for the consolidation problem.

While we do not aim at recovering large missing parts of point clouds, the concurrent work by Han et al. [HLH*17] targets completion of 3D shapes. In addition to a global structure inference network, their deep learning architecture includes a patch-based local geometry refinement network. The latter is built with voxel grids and 3D CNN-s, while we propose a network component to project unordered points to 2D heightmaps. This makes our method memory efficient and suitable to preserve fine details. Even if the general application is different, we leave the comparison with the local geometry refinement network of [HLH*17] for future work.

Sparse Representations for Geometry in Deep Learning There are several ideas in the deep learning literature to handle 3D geometries efficiently via exploiting the sparsity of the data by sparse convolutions [Gra14, Gra15, ERW*17], probing filters with a sparse set of points [LPS*16], mapping inputs to a permutohedral lattice [JKG16], a voting scheme for sliding-window based object detection [WP15], applying convolutional neural networks to images depicting multiple views of a 3D object for classification and recognition [SMKLM15, QSN*16], extracting features in a preprocessing step [FXD*15, GZC15, DJÖ*17], or representing shapes in a spectral domain [BZSL13, MBBV15]. These methods, however, are not designed to handle general large point cloud data.

For analyzing point clouds, Qi et al. [QSMG17] have recently proposed a neural network architecture. This method can deal with the order invariance of points, and can also be nested hierarchically for an understanding of geometric features at multiple scales, similar to a convolutional neural network [QYSG17]. Although it performs well for descriptive tasks on point clouds such as classification and segmentation, these architectures are not designed for generating points representing geometric details for accurate surface reconstruction, which is the focus of our work.

3. Algorithm Overview and Training Data Generation

3.1. Overview

The main idea of our technique is to learn a local mapping that transforms each set of points extracted from a local patch of the input point cloud to its consolidated version, where the output points sample the underlying surface very accurately and densely. The union of all such local output sets give us the final output point cloud. We define a patch as the set of points included in a local neighborhood. In particular, we represent a patch of geometry around a point as an oriented 2D heightmap that stores distances to the sample points in the neighborhood along a given direction. This 2D representation of local patches makes it possible to exploit the strengths of deep learning architectures for image denoising and super-resolution, and extend them to the task of processing unstructured 3D points.

In order to learn the mapping from a noisy patch of points to its consolidated version, we designed a new neural network architecture composed of fully differentiable components, as shown in Figure 2. The first component, *Heightmap Generation Network (HGN)* receives the matrix $\mathbf{X}_{n\times 3}$ that stores the x, y, z-coordinates of n input noisy points in the patch, and generates a noisy heightmap image H_N of resolution $k \times k$. In particular, it first learns a local coordinate frame for projection, projects the points onto the correponding image plane with a projection module, and resamples the resulting heightmap to obtain the regularly sampled image H_N. The second component, *Heightmap Denoising Network (HDN)*, uses image convolutions to transform the noisy heightmap H_N into a denoised version H_D. Finally, by transforming the pixel coordinates of H_D into point locations according to the learned image plane parameters and the stored distance values, the consolidated patch is generated and stored as a list of n' point coordinates $\mathbf{Y}_{n'\times 3}$, $n \leq n' \leq k^2$. In addition to learning the positions of the consolidated points, we propose a simple extension of our network architecture that allows us to learn consolidating their normals as well, if noisy normals of the input $\mathbf{X}$ are supplied or pre-computed.

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

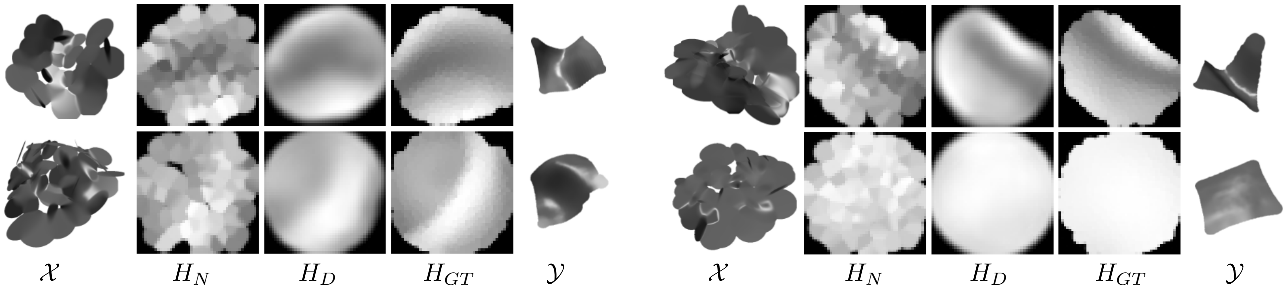

Figure 3: *Given an input point cloud patch $\mathcal{X}$ stored as raw point locations in the matrix* **X***, our network projects and resamples the geometry to convert it into the image H_N, and processes this image to produce H_D (here we also show ground truth H_{GT}'s for reference). This is finally back-projected to get the consolidated point set $\mathcal{Y}$.*

3.2. Training Data Generation

We start with a set of pairs of an input patch, and the corresponding ground truth output patch. These are cut out from input and ground truth output point clouds in spherical neighborhoods of radius r. For each pair, we thus have a set $\mathcal{X}$ of noisy and sparse points, and the corresponding denser set $\mathcal{Y}_{GT}$ of consolidated, i.e. denoised and up-sampled, points. We then extract a ground truth heightmap H_{GT} from $\mathcal{Y}_{GT}$. At training time, the aim of the network is to produce a denoised heightmap H_D that is as similar as possible to H_{GT}, starting from the input set $\mathcal{X}$.

Ground Truth Heightmap Generation The consolidated patch $\mathcal{Y}_{GT}$ is not directly fed to the network, but transformed to a 2D representation: we aim at extracting a ground truth heightmap H_{GT} which best encodes, in a 2D image, the 3D geometry. We thus want to find a normalized vector $\mathbf{n}_{GT}$ of a proper image plane positioned at an offset $-r\mathbf{n}_{GT}$ (to avoid negative distances). Since a heightmap can represent only one layer of geometry, we would like to have the least amount of points from different depth levels projected onto the same image pixel and thus averaged. In practice, we set the vector $\mathbf{n}_{GT}$ as the average of the normals of the points in the consolidated set $\mathcal{Y}_{GT}$. Due to the high density of $\mathcal{Y}_{GT}$, its uniform sampling, and lack of noise, we found this average is robust for capturing local geometries for the patch sizes we consider. Given the image frame defined by $\mathbf{n}_{GT}$, and an orthogonal vector $\mathbf{d}_{GT}$, the consolidated points are projected onto the plane orthogonal to $\mathbf{n}_{GT}$, and transformed into image coordinates. The distances between the original points and the projected ones are interpolated with gaussians to produce a resampled heightmap stored in H_{GT}. The same heightmap generation procedure is performed in a custom module within our network in the HGN component. We thus refer to Section 4.1 for a more detailed explanation of the operations.

Data Augmentation Note that the orientation of $\mathbf{n}_{GT}$ is ambiguous: both $\mathbf{n}_{GT}$ and $-\mathbf{n}_{GT}$ would produce a valid heightmap, even though the resulting images can be substantially different. We thus choose the sign of $\mathbf{n}_{GT}$ randomly for every patch, in order to ensure that we feed the network with varied data. The vector $\mathbf{d}_{GT}$ defines rotation of heightmaps on the image plane. We choose a random $\mathbf{d}_{GT}$ orthogonal to $\mathbf{n}_{GT}$ to make the learned representation invariant to this degree of freedom. When feeding data to the network during training, pairs $(\mathcal{X}, \mathcal{Y}_{GT})$ are randomly extracted from the training point clouds by positioning centers of the neighborhoods at random points in input point clouds. We thus get a dense coverage of each geometry in the database. Finally, we further augment the patch pairs dataset by random resampling of the input point clouds, getting an arbitrary sampling rate for each $\mathcal{X}$. The number of points can then be matched to n by random down-sampling or replication of points in $\mathcal{X}$, as the network expects a fixed-size input matrix **X**.

4. A Network Architecture for Point Cloud Consolidation

Given the training data consisting of pairs $(\mathcal{X}, \mathcal{Y}_{GT})$ that are transformed into $(\mathbf{X}, H_{GT})$ as described above, we would like to design an architecture that can be trained with these pairs at training time, and produce consolidated output points $\mathcal{Y}$ for an arbitrary $\mathcal{X}$ at testing time. In this section, we elaborate on the main components of our network (Figure 2) in more detail, and explain how the output of the network (used in a feed forward manner) serves to produce the final consolidated point cloud.

4.1. Heightmap Generation Network

The goal of our first component, *Heightmap Generation Network (HGN)* in Figure 2, is to estimate an image plane orientation, and to produce a corresponding noisy heightmap by projecting the points stored in **X**. The component is thus divided in two parts: first, a vector **n** and an orthogonal direction **d** are estimated from the input points in $\mathcal{X}$, then, the input points are projected to the image plane, generating a noisy heightmap image H_N.

Frame Estimator In order to estimate **n**, the component *Frame Estimator* (Figure 2) needs to deal with the unordered structure of the input point set given in **X**. To tackle this problem, we utilize the idea of using a symmetric function with respect to ordering of points **X** with a single max pooling layer from a very recent work [QSMG17], and use it to predict **n**. In the original work, the final fully connected layers produce a global descriptor, which is then used for classification or segmentation. In our method, we adopt the same architecture but modify the output of the final fully connected layers to produce the 3D vector **n**. Additionally, it would be beneficial that the learned representation is invariant to translations and rotations of **X**. We achieve this by centering the points in **X** by subtracting the patch center, and by feeding patches with random rotations at training time.

As elaborated on in Section 3.2, due to the ambiguity of the sign of $\mathbf{n}_{GT}$, our dataset contains patches of either orientation. Even without this augmentation, we found out that there can be many similar patches with similar $\mathbf{n}_{GT}$ but with opposite signs. The frame estimator then typically learns to estimate an average, which significantly distorts the learned $\mathbf{n}$ and thus heightmaps. To avoid this averaging, at training time, we snap the orientation of $\mathbf{n}$ to that of $\mathbf{n}_{GT}$ by setting $\mathbf{n} \leftarrow \mathbf{n}(\mathbf{n}^T\mathbf{n}_{GT})$ and normalizing. This ensures that the network is forced to learn the direction of $\mathbf{n}$, and choose either of the two orientations, and not their average. Note that this snapping component is not present at testing time, where the orientation of $\mathbf{n}$ is irrelevant for generating the final consolidated patch. Similarly, at training time, the direction vector $\mathbf{d}$, is kept as close as possible to $\mathbf{d}_{GT}$ and orthogonal to $\mathbf{n}$ at each iteration by setting $\mathbf{d} \leftarrow \mathbf{d}_{GT} - (\mathbf{d}_{GT}^T\mathbf{n})\mathbf{n}$ and normalizing in the component, to ensure rotations on the plane are not averaged. At testing, a random $\mathbf{d}$ orthogonal to $\mathbf{n}$ is sufficient as the learned representation is invariant to rotations on the plane.

Projector The second part of HGN takes the vectors $\mathbf{n}$, $\mathbf{d}$, and the input point set in the form of the matrix $\mathbf{X}$, and renders a 2D heightmap H_N regularly sampled at pixel coordinates. The projector component first projects the 3D points onto the image plane given by the vectors $\mathbf{n}$ and $\mathbf{d}$, and positioned at an offset of $-r$, to avoid negative distances. Hence, for each point $\mathbf{x} \in \mathcal{X}$ (i.e. each row of $\mathbf{X}$), we define

$$\mathbf{p} = \mathbf{x} - (\mathbf{x}^T\mathbf{n} + r)\mathbf{n}, \tag{1}$$

as the projected position of $\mathbf{x}$. For each projected point $\mathbf{p}$, we also store the distance $\|\mathbf{x}-\mathbf{p}\|$. The projected points are then transformed into image coordinates as

$$\mathbf{i} = \frac{k}{2r}\left[\mathbf{p}^T\mathbf{d}+r \quad \mathbf{p}^T(\mathbf{n}\times\mathbf{d})/\|(\mathbf{n}\times\mathbf{d})\|+r\right]^T, \tag{2}$$

where H_N is a $k \times k$ image. We thus get the image coordinates $\mathbf{i}$ and the corresponding distance values $D(\mathbf{i})$. The heighmap image H_N is then generated by interpolating the distances $D(\mathbf{i})$ on the image plane with Gaussian interpolation at pixel centers.

We use a Gaussian with a cutoff such that for a given pixel center $\mathbf{c}$ in image coordinates, only the points $\mathbf{i}$ given by $\mathcal{N} = \{\mathbf{i} \mid \|\mathbf{c}-\mathbf{i}\| < \delta\}$ need to be considered. The value at $\mathbf{c}$ is then given by

$$H_N(\mathbf{c}) = \begin{cases} \frac{1}{W(\mathbf{c})}\sum_{\mathbf{i}\in\mathcal{N}(\mathbf{c})} g(\mathbf{c},\mathbf{i})D(\mathbf{i}), & \mathcal{N}(\mathbf{c}) \neq \emptyset \\ 0, & \mathcal{N}(\mathbf{c}) = \emptyset, \end{cases} \tag{3}$$

where $W(\mathbf{c}) = \sum_{\mathbf{i}\in\mathcal{N}(\mathbf{c})} g(\mathbf{c},\mathbf{i})$, and $g(\mathbf{c},\mathbf{i}) = e^{-\frac{\|\mathbf{c}-\mathbf{i}\|^2}{\sigma^2}}$. We show some examples of generated H_N at testing time in Figure 3. All operations of the projection module are differentiable with respect to the inputs, thus the gradients can be back-propagated through the network.

4.2. Heightmap Denoising Network

Our second network component, *Heightmap Denoising Network (HDN)* in Figure 2, takes the noisy heightmap H_N as input, and generates a denoised version H_D as its output. As this is a mapping between regular images, many previous methods from the image processing literature can be utilized. CNN-based architectures have been successfully adopted for image denoising and super-resolution [KLL15, ZZC*16], obtaining state-of-the-art results. We thus also adopt a deep CNN for this step. HDN is inspired by a recent network architecture [KLL15], consisting of a sequence of 10 convolutional layers with depth 64, and filters of size 7 × 7. After each convolution, batch normalization and a rectified linear unit layer (RELU) are applied.

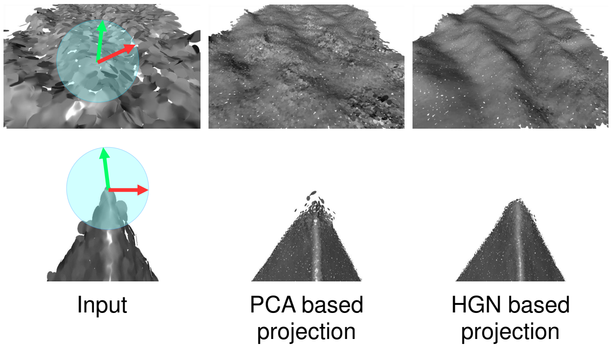

Figure 4: *Estimating consistent local directions for projection that are robust to noise and result in heigtmaps that capture local structure well is difficult with geometric methods such as PCA (shown in red), whereas our architecture generates a robust and propoer direction for heighmap generation (green). This is essential for HDN to generate accurate and consistent results as we show for consolidated point sets with projections estimated by PCA and HGN (middle and right).*

Examples of noisy H_N and corresponding denoised H_D heightmaps obtained at testing time are shown in Figure 3. The network learns a very accurate mapping, leading to H_D very close to the ground truth patch images.

4.3. Training Procedure and Analysis

Training and Loss We first train HDN by using the ground truth plane parameters, thus by substituting $\mathbf{n}$ with $\mathbf{n}_{GT}$ in HGN, and minimizing the loss $\|H_D - H_{GT}\|_2$. This allows us to train the convolutional layers of HDN on patch pairs with perfect projection and resampling. Once the weights of HDN are trained, we fix them and train HGN with the same loss as before. By imposing the same loss, we force HGN to learn the best projection such that the projected heightmap, once denoised, becomes as similar as possible to the ground truth image H_{GT}.

Robustness to Noise and Sampling Adopting a learning based approach to estimate projection and denoising simultaneously makes our local fits robust to noise and sparse data, and consistent with the local geometric features. This is very hard for purely geometric algorithms, such as fitting local planes with PCA. Such methods result in parameters that are overfitted to noise or biased with respect to the patch structure, depending on the size of the neighborhood, noise level, and local geometry.

Figure 4 (top, left) shows an example for a noisy patch (blue circle), where a PCA-based estimation of the normal vector at the patch center is given in red, and the **n** estimated by our network in green. The former is obtained by averaging the normals of the points in the patch, all estimated with a small neigborhood size (one third of the patch size) with PCA. Our estimated **n** generates a better heightmap that is consistent for noisy patches, as it captures the underlying local geometry well. This is clear also in the final consolidated point clouds, as we show for PCA-based projections followed by HDN in Figure 4 (top, center), and our full architecture HGN + HDN in Figure 4 (top, right). Utilizing our full architecture results in a much smoother geometry while preserving important features.

Such local fits with geometric techniques are also problematic when the size of neighborhood considered is large with respect to local geometric structures, which is the case for all our patches, as we need to capture local structure within our networks. In Figure 4 (bottom), the same comparison as in (top) is shown for a sharp feature, but in this case the PCA normals are computed with a size as large as that of the patch. The **n** estimated by our network (green) allows HGN to generate a proper heightmap around the peak, which can then be effectively denoised by HDN (bottom, right). PCA, on the other hand, estimates a vector (red) that is perpendicular to **n**. This results in a heightmap where distances of points are averaged, leading to artifacts in output consolidated point sets (bottom, center).

4.4. Processing Point Clouds at Testing Time

Processing a Single Patch At testing time, given an input point cloud, a spherical patch of radius r around a point is extracted. If the normals of the input point cloud are provided or pre-computed, the patch can be further refined by considering location-wise and normal-wise close points. The patch is first resampled to obtain n points as in data generation for training (Section 3.2), and centered at the origin. It is then fed to the *Frame Estimator* component of HGN (Figure 2) to estimate the normal direction **n** for the plane over which the heightmap H_N is defined. A random direction vector **d** orthogonal to **n** is then computed, and the noisy height map H_N is generated with *Projector*. The H_N is then denoised by HDN to produce H_D, which is finally converted into a point cloud by *Backprojector* with the same frame that *Projector* uses. Each pixel center with the corresponding depth given by H_D projects into a 3D point. Pixels with zero values are not projected, as they do not represent geometry (the resulting positions fall out of the patch sphere due to the offset we use as explained in Section 4.1), but are rather placeholders for no geometry. The resulting consolidated set of points is then translated to the original position of the input patch. We show examples of consolidated sets **Y** in Figure 3. As compared to the input noisy and sparse set $\mathcal{X}$, we get a denoised and much denser output set $\mathcal{Y}$.

Reprojection Density The above process is repeated independently for patches around every point of the input point cloud. The generated point sets are all retained in a final set representing the consolidated point cloud, without any further processing such as averaging of point locations.

In order to introduce overlaps between patches and hence produce a dense output, we evaluate a patch around each input noisy point. Less overlaps can be introduced for efficiency, at the cost of quality due to sparseness of the output. An example of output point cloud with less overlapping is shown in Figure 5 (left), where patches were extracted only around one quarter of the input noisy points. Compared to the denser version in Figure 5 (right), it is smoother and contains several small holes.

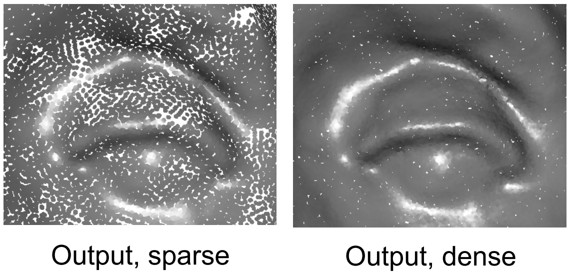

Figure 5: *The output consolidated point cloud, obtained by evaluating only patches around one quarter of the input noisy points (left), and around every input noisy point (right).*

The number of new points n' sampled on H_D further defines the density of the final consolidated point cloud. For example, reprojecting a single point corresponding to the central pixel of H_D would produce a denoised point cloud with the same number of points as the input, thus possibly losing the ability to preserve fine details. On the other hand, reprojecting a point for every pixel of H_D could lead to artifacts at the borders of the patch, nearby the zero value pixels which are placeholders for no geometry. The convolutions, indeed, may introduce smooth transitions between the zero values pixels and the ones representing geometry. We found projecting the pixels in a central part of H_D for each patch produces best results. After H_D is computed, we thus reproject only the pixels that fall into a square of size m around the patch center.

4.5. Extension for Point Normals

As we get a dense and denoised point set as the consolidated output, surface normals at the output points can simply be computed with existing geometric methods such as PCA. However, for cases where there are sharp features to be preserved, we might still not get the exact expected sharpness for normals as we are limited by the resolution of the intermediate image-based representation H_D. For such cases, we thus propose an additional network component for denoising point normals. The idea is to denoise, instead of a single-channel noisy heightmap H_N as before, a three-channel noisy normal map N_N, generated from the input normals. The architecture is the same as before with a few modifications. First, HDN processes images N_N of three channels, each representing a component of normal vectors. Second, each channel j of the normal map N_N is generated as in Equation 3, by interpolating the j^{th} component of the surface normals denoted by $N(\mathbf{i}, j)$ for projected points $\mathbf{i}$ as

$$N(\mathbf{c}, j) = \frac{1}{W(\mathbf{c})} \sum_{\mathbf{i} \in \mathcal{N}(\mathbf{c})} g(\mathbf{c}, \mathbf{i}) N(\mathbf{i}, j). \tag{4}$$

Here, $N(\mathbf{c}, j)$ is the j^{th} component of the normal vector stored at the pixel center $\mathbf{c}$, expressed in image plane coordinates as before.

At testing time, given an input patch $\mathcal{X}$, the maps H_N and N_N are

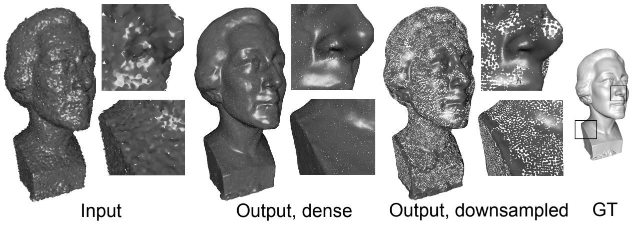

Figure 6: *From an input noisy point cloud (Input), our method produces a dense, consolidated version (Output, dense). Prior to surface reconstruction, this can be optionally adaptively downsampled (Output, downsampled) to speed up reconstruction algorithms. The ground truth mesh (GT) is shown as reference.*

generated with respect to the estimated frame, and their denoised versions H_D and N_D are obtained via two separate HDN's. We can then obtain the point locations from H_D by back-projection as before, and normals from N_D by changing the coordinate system according to the same estimated frame.

5. Experiments and Analysis

5.1. Network Implementation and Parameters

For all our experiments, we set the patch radius r to 5 times the average spacing between the input points, and resample the patches to $n = 100$ points (Section 3.2). We use images of size $k = 48$, and set $\sigma = 1/r$ for generating H_N, and $\sigma = 1/2r$ for generating the training dense heightmaps H_{GT}, with $\delta = 2.5\sigma$. We back-project points from the heighmap image H_G in a square of size $m = 24$. The whole architecture is trained with the Adam Optimizer with an initial learning rate of 0.0001 lowered by 10 times every 30k steps. We feed the patches in batches of size 8. For the PointNet components [QSMG17], we use the default parameters and their basic code for handling unordered point sets as input. The network was implemented in TensorFlow.

5.2. Pipeline For Surface Reconstruction

A key application of point cloud consolidation is to serve as a preprocessing step to surface reconstruction algorithms. These algorithms are affected by noise and sparseness of the input data. Thus, providing a consolidated, dense point cloud is critical for improving the reconstructed surfaces. We start by applying our method to a noisy input point cloud and generating a consolidated dense version. Since the resulting point cloud is very dense, we can easily downsample it in an adaptive fashion, keeping a high density of points in proximity of the features. This step considerably speeds up the reconstructions without loosing quality. In particular, we use a simple and efficient clustering algorithm [PGK02], where downsampling is obtained by grouping points in local clusters. In order to keep a denser sampling near the features, the size of the clusters is adapted to the local variation of the point set. We use 30 as a maximum cluster size and 0.02 as maximum surface variation (for the *flags* dataset, see below, 20 and 0.03).

Figure 6 illustrates an input point cloud (of about 50k points) from our *sculptures* dataset (see below), our dense consolidated point cloud (about 1.6M points), our subsampled point cloud (about 150k points), and the ground truth mesh from which the noisy input was sampled. The generated dense point cloud does not present noise and preserves detailed features such as the nostril and the edges of the base. Those features are also preserved in the downsampled point cloud, while reducing the overall sample count.

If not learned through our point normals network extension, we estimate the point normals of the consolidated point cloud using PCA of local neighborhoods and a Riemannian graph for their global orientation. Due to the high density and quality of the output consolidated point cloud, this simple approach already obtains high quality results. We compute point normals with PCA on 50 nearest neighbor points.

Finally, we reconstruct the underlying surface by extracting the iso-surface of the *RIMLS* [OGG09] using the marching cubes algorithm. We refer to our surface reconstruction results as *OURS-R*, and our output dense point clouds as *OURS*. We use a spatial low pass filter of 7 to 10 times the local spacing of output points for RIMLS. The RIMLS sharpness parameter σ_n is set to 0.75.

5.3. Datasets

For our experiments, we built three datasets: two with synthetic data and different levels of noise (*sculptures* and *flags*), and one with real-data from a sensor (*Kinect v2*), each composed of objects with similar features and separated in a training and a testing subset. For testing our point normals network extension, we built an additional synthetic dataset containing multiple geometric sharp features (*geometric shapes*). The test point clouds only contain point locations, without point normals or any additional attributes.

For training, we have three ground truth models for the *sculptures*, three for *flags*, four for *geometric shapes*, and four for *Kinect v2*, from which many training patches are generated. The ground truth point clouds of *sculptures*, *geometric shapes* and the noisy and ground truth point clouds of *Kinect v2* were extracted from the models also used in recent works [WLT16]. For the Kinect models represented as meshes, we remove the connectivity information and just retain the vertex locations. The meshes of the *flags* dataset were generated by animating a waving flag mesh and randomly selecting frames. While the *flags* dataset is very specialized (each model has wrinkles of similar shapes and sizes), the other datasets are more general. The ground truth models of our datasets are shown in Figure 7.

For each model in the training sets, ground truth point sets are twice as dense as the input point sets, and are generated by Poisson disk sampling for equal distribution of points. Synthetic Gaussian noise was dynamically added to the input points at training time, as the training patches were generated. For the *sculptures* dataset, three training sessions were performed, each with noise of a different standard deviation ($\sigma_1 = 0.037r$, $\sigma_2 = 0.075r$ and $\sigma_3 = 0.15r$), while for the *flags* and the *geometric shapes* datasets, $\sigma = 0.075r$.

Our testing sets contains six models for *sculptures*, ten for *flags*, two for *geometric shapes*, and 14 scans of three models for *Kinect v2*. For each model, an input noisy and sparse point cloud was sampled (except for *Kinect v2*, where we already have noisy scans) from the

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

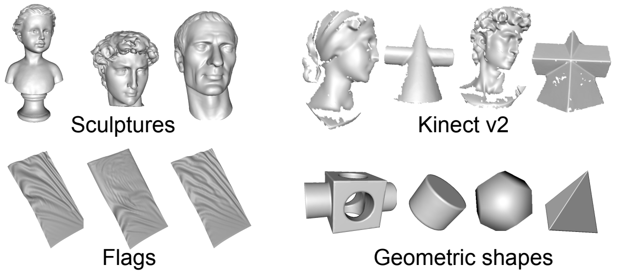

Figure 7: *The ground truth meshes for our four training datasets.*

ground truth model with the same conditions as in the corresponding training dataset. The input point clouds of the *sculptures* dataset consist of about 70k points on average, while the other datasets come with around 15k points for testing models.

5.4. Comparisons

We compare numerically (reconstructions) and visually (point clouds and reconstructions) to five common and state-of-the-art methods for point consolidation and surface reconstruction: Poisson Surface Reconstruction [KBH06], APSS [GG07], RIMLS [OGG09], WLOP [HLZ*09], and EAR [HWG*13]. While Poisson, APSS, and RIMLS directly produce an iso-surface, WLOP and EAR generate a resampled point cloud. For comparing our surface reconstruction results, we thus apply RIMLS to the output of WLOP and EAR, and utilize marching cubes to extract the final surface for all methods. We refer to these combinations as WLOP-R and EAR-R.

In order to numerically compare the mesh reconstruction results, we adopt the Hausdorff distance between the reconstructed meshes and the ground truth ones. As we have models that are not closed, we used the one-sided Hausdorff distance from a ground truth mesh to the reconstructed one, in order to avoid including errors due to extra surface parts around the boundaries in the reconstructed mesh. The Hausdorff distance is normalized with respect to the diagonal of the bounding box of the mesh, and multiplied by 10^4. For every dataset, we compute the average Hausdorff distance for all testing models.

We exhaustively search for the best parameters for the other methods by running an extensive test for each model. For APSS, RIMLS, WLOP-R and EAR-R, the spatial low pass filter parameter is tuned separately for each dataset and each method, by testing a set of values varying between three to ten times the local point spacing, and choosing the best result. The RIMLS sharpness parameter σ_n is set to 0.75 for all methods. For Poisson Surface Reconstruction, an octree depth parameter of 14 is used. The WLOP and EAR neighborhood radius parameter is set to 8 times the average spacing of the input point set, and the EAR sharpness parameters to an angle of 30 with edge sensitivity 0.05. For methods that require surface normals, we estimate them with PCA again by using optimized values for each case.

5.5. Experiments

For all experiments, we observed a significant visual and numerical improvement over the existing methods when using our technique.

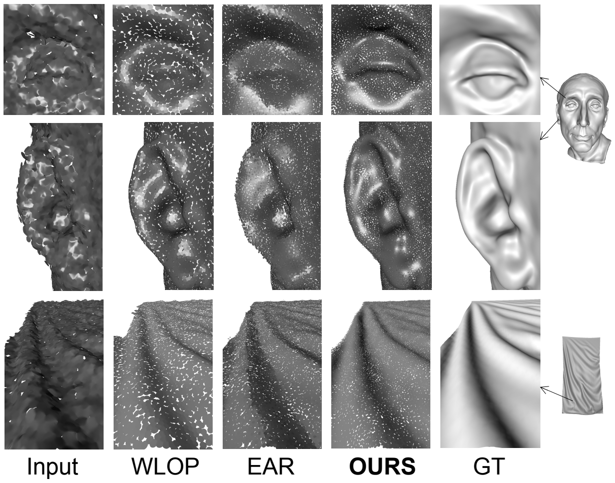

Figure 8: *Consolidated point clouds on the noisy input Nicolo (from the sculptures dataset with σ_2), and on a model from the flags dataset. Our method captures local structures of the ground truth (GT) model accurately.*

We show example input and consolidated output point clouds using our technique as well as others in Figure 8. While WLOP oversmoothes the details of the model Nicolo and the wrinkles of the flag, the dense point clouds of EAR deform the geometry by creating extra sharp edges that are not present in the original models, e.g. at the border of the ear or at the peak of the flag wrinkle. Our dense point clouds reproduce the ground truth local structures more faithfully, e.g. we get a realistically rounded eyelid without turning it into a sharp edge.

We show visual comparisons of reconstructions in Figures 1, 9, 10, and 11. We observe that Poisson, APSS and WLOP-R tend to generate oversmoothed surfaces, as can be seen for many surface features, e.g. for the eyes of Nicolo in Figure 1, the ear and hair of Bimba in Figure 10, or the flag in Figure 11. The oversmoothing effect is confirmed by the last row of Figure 9, displaying the distances from the ground truth to the closest point on the reconstructed meshes. In the detailed ear and hair regions, these methods have high errors. On the other hand, these methods can also produce noisy results depending on the input, as for the buste and neck of the Boy model in Figure 11. Our technique produces faithful reconstructions for all cases, avoiding over- or under-smoothing of local structures.

RIMLS and EAR are designed to preserve sharp features. Indeed, EAR-R produces accurate results in geometric shapes with clear edges such as the base of Eros in Figure 9. However, it fails to correctly preserve more organic, detailed features such as the face of the same model or the ear and hair of Bimba, as shown in Figure 10 and in the distance maps in Figure 9. Similarly, RIMLS performs better on sharp features, but produces bumpy results in smooth regions, such as the buste of Bimba, and overall cannot capture delicate structures such as the eyes of Nicolo in Figure 1, or the wrinkle profile in Figure 11. Instead of sharpening details, our method outputs high quality structures that more faithfully reproduce the underlying geometry, thanks to the learned representation.

These visual results are confirmed by quantitative comparisons

© 2018 The Author(s)

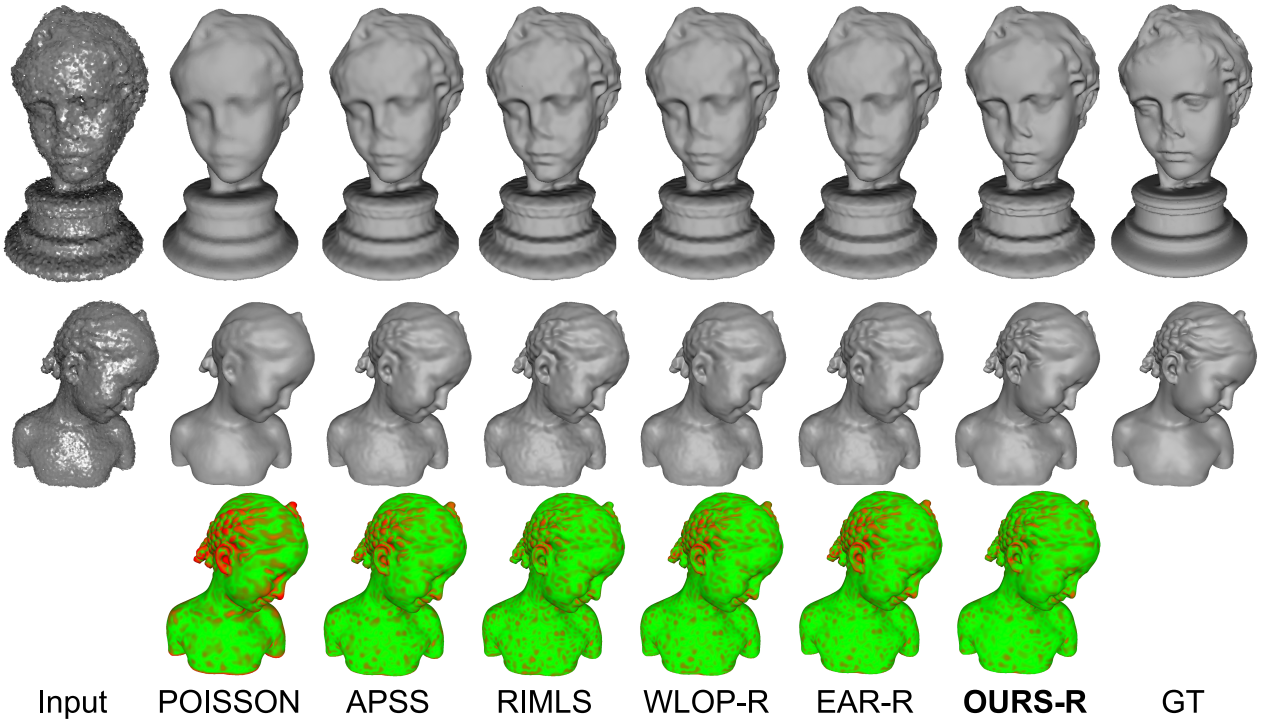

Figure 9: *Surface reconstructions for the test models Eros (top) and Bimba (bottom) from the sculptures dataset with σ_2, with ground truth meshes (GT) for reference. For Bimba, the distances from the ground truth mesh to the reconstructed meshes are also displayed (red encodes large values).*

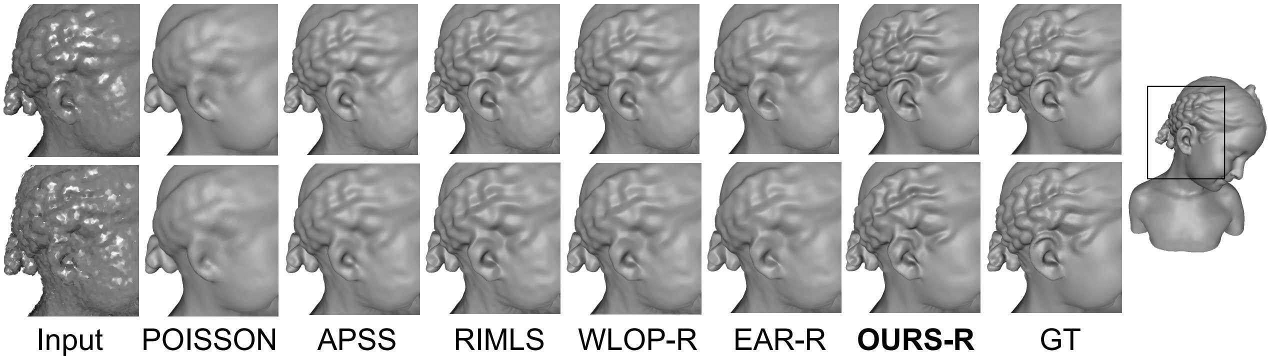

Figure 10: *Close-ups of the reconstruction of Bimba, with two different levels of noise σ_1 and σ_2. In both cases, our method (OURS-R) produces the most accurate ear and hair features, while keeping the cheek smooth as in the ground truth (GT).*

in Table 1. The best two results for every dataset are highlighted in bold. Our method obtains the smallest Hausdorff distance for every dataset.

Cross-Training In Figure 12, we analyze the importance of training datasets for accurate structure recovery. We show consolidated point clouds of the Nicolo model by using the ground truth plane normal $\mathbf{n}_{GT}$ and direction $\mathbf{d}_{GT}$ for each patch, and varying the heightmap denoising procedure. In particular, in the first column, the H_D's are generated by simply smoothing the H_N with Gaussian interpolation using a small σ, in the second column the same smoothing is applied but with a larger σ, in the third column we use

Dataset	APSS	RIMLS	WLOP-R	EAR-R	**OURS-R**
Scu. σ_1	2.85	**2.62**	2.99	3.6	**2.57**
Scu. σ_2	**4.11**	4.27	4.26	4.56	**3.70**
Scu. σ_3	**6.28**	7.37	6.54	6.74	**6.14**
Flags	**5.69**	6.03	**5.69**	5.93	**5.55**
Kin.v2	17.58	19.40	**17.24**	17.83	**17.10**

Table 1: *The Hausdorff distances averaged over the testing models in each dataset for each method. The best two performing methods are highlighted for each dataset. Our method (OURS-R) outperforms the others in every dataset.*

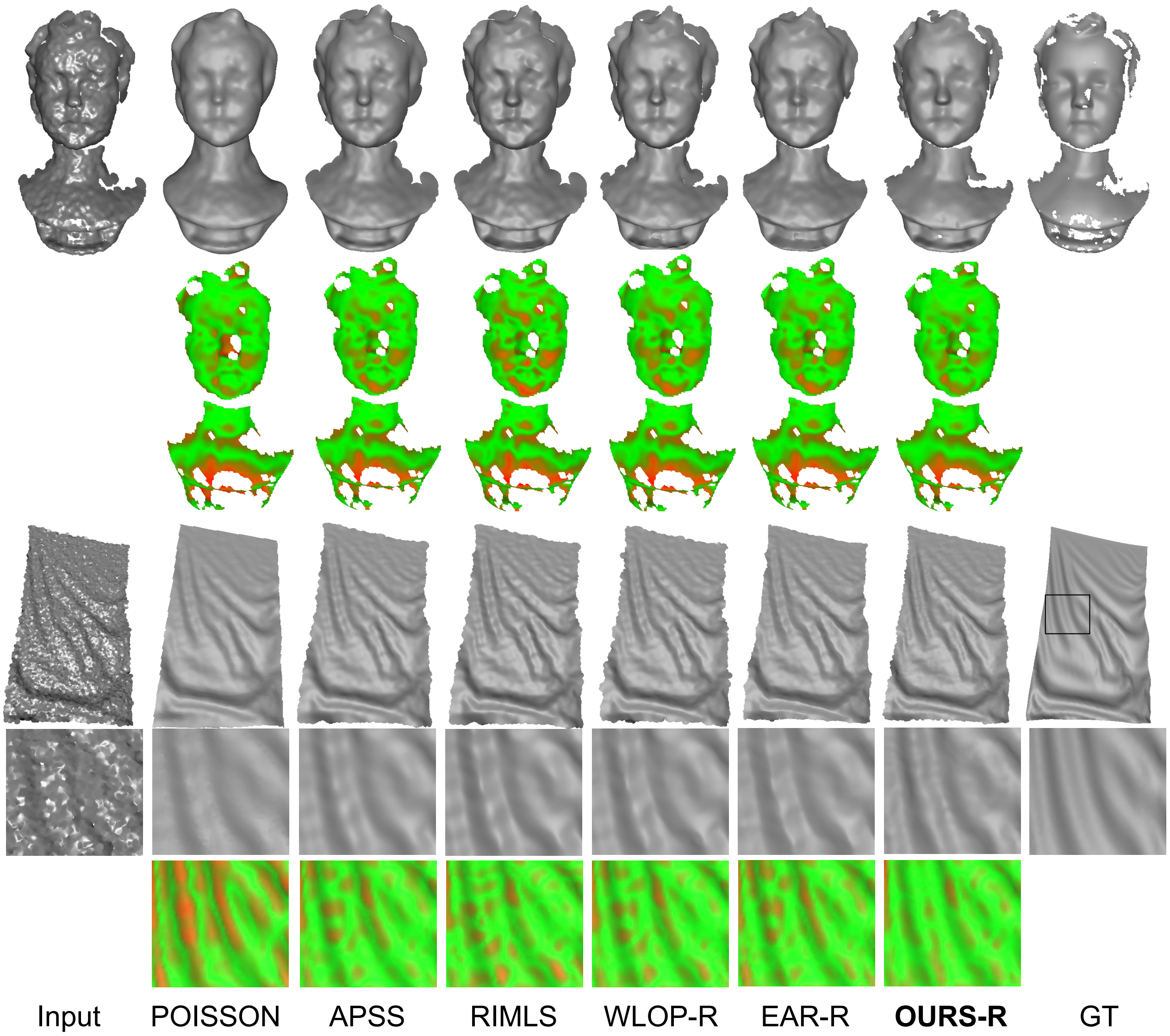

Figure 11: *Reconstructed surfaces for the testing models Boy (top) and a flag (bottom) from the Kinect v2 and flags datasets, respectively. The distances from the ground truth (GT) mesh to the reconstructed meshes are also plotted for the flag.*

HDN to generate the H_D but using the weights trained on the *flags* dataset, and in the fourth column we use the weights trained on the *sculptures* dataset. The last column is the ground truth, and every column contains the generated point cloud and four example H_D's. Simpler image denoising techniques produce considerably worse results, and by specializing the training with models of the same class as the testing models, we can obtain substantially better results than with more general datasets.

Point Normals While our dense output point clouds allow for accurate estimation of surface normals with PCA, we found our extension for denoising normals to be useful for preservation of sharp features for geometric objects, as elaborated on in Section 4.5. In this case, the input normals to generate N_N were estimated from the noisy point clouds with PCA, and oriented with a Riemannian graph. In Figure 13, we show reconstruction results on the *geometric shapes* testing dataset (Fandisk and Icosahedron). The normals on the consolidated point clouds, color-coded in this figure, are estimated with PCA, or learned with the normal estimation network for comparison. The close-ups of the point clouds (top), and the reconstructed surfaces (bottom) illustrate that the learned normals better preserve the sharp features. In Figure 14, we further illustrate some noisy normal maps N_N, denoised versions N_D, and the ground truth N_{GT} for the Fandisk model. We can observe that our N_D contains very sharp edges.

Timing All trainings were performed for 200k steps, lasting on average 5 hours. The total time required to denoise an input patch is about 0.013 seconds, out of which about 44% is for preprocessing the input, 15% for estimating H_N, 33% to denoise it to H_D, and 7%

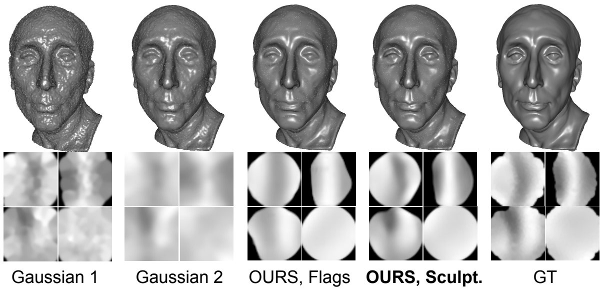

Figure 12: *Different denoising variations. The noisy heightmaps H_N of a noisy Nicolo model are denoised with: Gaussian smoothing with a small σ, Gaussian smoothing with a large σ, using our network trained on the flags dataset, using our network trained on sculptures. The resulting consolidated point clouds are shown (top) with four example denoised heightmaps (bottom). The ground truth (GT) data is shown for reference.*

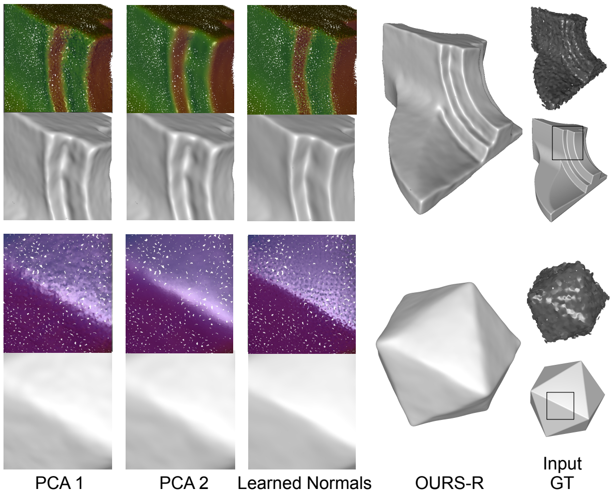

Figure 13: *Consolidated point clouds with color-coded normals and RIMLS reconstructions of a noisy input Fandisk (above) and Icosahedron (below) from the geometric shapes dataset. The normals are: estimated with PCA with a small radius (PCA 1), PCA with a large radius (PCA 2), or learned with our normal estimation network (Learned Normals). The latter better preserves sharp edges.*

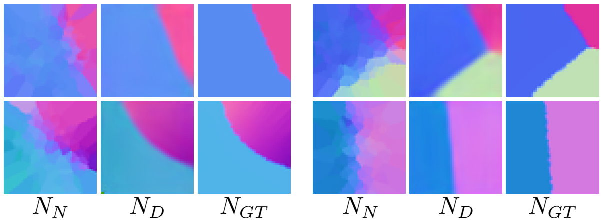

Figure 14: *Example normal map denoising results from the Fandisk model. N_N is the input noisy, and N_D is the denoised normal map. The ground truth normal map N_{GT} is shown for reference.*

to reproject the points. For the bimba model (of about 60k points), the total processing time was about 90 seconds on a GTX 970 and a i5-3570 CPU, 3.40GHz. As each patch is local and processed independently, our technique thus allows for real-time patch-wise consolidation, and is trivially parallelizable.

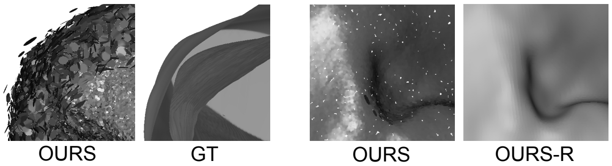

Figure 15: *Limitations of our method. In case of two surface sheets falling into the same patch (left, GT) , our method averages the positions of the points creating a noisy result (left, OURS). Occasional frames can be badly estimated in the presence of complex patches unseen during training, and thus some bad points can be generated (right, OURS). Due to the high density of our results, the final reconstructed surfaces are not affected (right, OURS-R).*

Limitations In this paper, we target the typical problems of noise and sparse sampling in input point clouds. However, when the input point cloud contains relatively large holes, our current scheme is not able to fill them with a sampling as dense as in the other parts of the point cloud. This is because there is less overlap of projected patches near the holes, since we generate patches of consolidated point clouds only around existing input points. As a consequence, the RIMLS reconstruction might lead to deformed surface parts in those regions. This can be seen for the base of the Boy model in Figure 11. This could be alleviated by having a denser sampling of patches around the holes, starting from the boundaries and progressively closing the holes, similar to texture synthesis. In case of missing parts considerably larger than the patch size, a global filling approach would be required. Our method is designed to capture local structures for manifold surfaces, as many previous techniques including MLS based approaches. This allows us to use local heightmaps as an intermediate representation. However, such a representation comes with well-known limitations for non-manifold structures and large surface parts that cannot be represented with such a parametrization. Possible solutions are utilizing multi-depth maps and more complex differentiable parametrizations that can be efficiently trained. A typical example is when two surface sheets fall into the same patch, where our current method would average their locations as shown in Figure 15 (left). If the input normals are provided, this problem can be solved by extracting patches considering location-wise and normal-wise close points, as mentioned in Section 4.4. We generate a patch around each point in an input point cloud at testing time. This means that for high levels of outliers, we might end up with extra output points that are far away from the surface. For these cases, a new training dataset and procedure need to be designed to set all depths values of H_D for an outlier to zero. Similarly, in cases of patches very different from the ones in the training set (e.g., partial patches at the borders), or badly performed training (e.g., too short training) occasional badly estimated outlier frames may occur. A badly estimated frame would, in most cases, produce a bad set of points, as can be seen in our output point cloud in Figure 15 (right). Since our produced point cloud is very dense and mainly made of consistent re-projections across patches, even in the presence of few bad points the final reconstruction is not negatively affected, as shown in the reconstructed surface in Figure 15 (right). Finally, our H_D images are sometimes slightly smoother than H_{GT}. This is a property of the used convolutions, and the behaviour could be improved by utilizing more advanced image network architectures such as Generative Adversarial Networks [GPAM*14].

6. Conclusions and Future Work

In this work, we presented PointProNets, a fully differentiable, CNN based deep learning architecture to process point clouds. The input unordered points are internally converted to regularly sampled height maps, which are suitable to be processed by modern and well-performing CNN architectures. We demonstrated the potential of this architecture by developing an end-to-end algorithm to consolidate raw point clouds, where local parametrizations and fitted surfaces are learned jointly, to achieve superior reconstructions where delicate features and details of surfaces are preserved.

Although we have focused on point cloud consolidation in the scope of this work, the proposed architecture has the potential to be used for many other points based geometry processing tasks. Moreover, as in our additional component for point normals denoising, the architecture could be easily extended to points with attributes, such as colors for joint depth-color data processing.

Acknowledgments Riccardo Roveri is supported by the Swiss National Science Foundation (grant No. 200021_146227/2).

References

[ABCO*03] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C. T.: Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (Jan 2003), 3–15. 2

[ASGCO10] Avron H., Sharf A., Greif C., Cohen-Or D.: &#8467;1sparsee reconstruction of sharp point set surfaces. *ACM Trans. Graph. 29*, 5 (Nov. 2010), 135:1–135:12. 2

[ASK*05] Anguelov D., Srinivasan P., Koller D., Thrun S., Rodgers J., Davis J.: Scape: Shape completion and animation of people. *ACM Trans. Graph. 24*, 3 (July 2005), 408–416. 3

[BTS*17] Berger M., Tagliasacchi A., Seversky L. M., Alliez P., Guennebaud G., Levine J. A., Sharf A., Silva C. T.: A survey of surface reconstruction from point clouds. *Computer Graphics Forum 36*, 1 (2017), 301–329. 2

[BZSL13] Bruna J., Zaremba W., Szlam A., LeCun Y.: Spectral Networks and Locally Connected Networks on Graphs. *ArXiv e-prints* (Dec. 2013). 3

[CBC*01] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C., Evans T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 67–76. 2

[DJÖ*17] Dibra E., Jain H., Öztireli A. C., Ziegler R., Gross M. H.: Human shape from silhouettes using generative hks descriptors and cross-modal neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 21-26, 2017* (2017). 3

[DQN17] Dai A., Qi C. R., Niessner M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017). 3

[ERW*17] Engelcke M., Rao D., Wang D. Z., Tong C. H., Posner I.: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017* (2017), pp. 1355–1361. 3

[FXD*15] Fang Y., Xie J., Dai G., Wang M., Zhu F., Xu T., Wong E.: 3d deep shape descriptor. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 2319–2328. 3

[GCB*17] Gharbi M., Chen J., Barron J. T., Hasinoff S. W., Durand F.: Deep bilateral learning for real-time image enhancement. *ACM Trans. Graph. 36*, 4 (July 2017), 118:1–118:12. 2

[GG07] Guennebaud G., Gross M.: Algebraic point set surfaces. *ACM Trans. Graph. 26*, 3 (July 2007). 2, 8

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, Ghahramani Z., Welling M., Cortes C., Lawrence N. D., Weinberger K. Q., (Eds.). Curran Associates, Inc., 2014, pp. 2672–2680. 12

[Gra14] Graham B.: Spatially-sparse convolutional neural networks. *ArXiv e-prints* (Sept. 2014). 3

[Gra15] Graham B.: Sparse 3d convolutional neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)* (September 2015), BMVA Press, pp. 150.1–150.9. 3

[GSH*07] Gal R., Shamir A., Hassner T., Pauly M., Cohen-Or D.: Surface reconstruction using local shape priors. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 253–262. 2

[GZC15] Guo K., Zou D., Chen X.: 3d mesh labeling via deep convolutional neural networks. *ACM Trans. Graph. 35*, 1 (Dec. 2015), 3:1–3:12. 3

[HLH*17] Han X., Li Z., Huang H., Kalogerakis E., Yu Y.: High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *IEEE International Conference on Computer Vision (ICCV)* (October 2017). 2, 3

[HLZ*09] Huang H., Li D., Zhang H., Ascher U., Cohen-Or D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 176:1–176:7. 2, 8

[HWG*13] Huang H., Wu S., Gong M., Cohen-Or D., Ascher U., Zhang H. R.: Edge-aware point set resampling. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 9:1–9:12. 2, 8

[IZZE16] Isola P., Zhu J.-Y., Zhou T., Efros A. A.: Image-to-image translation with conditional adversarial networks. *arxiv* (2016). 2

[JKG16] Jampani V., Kiefel M., Gehler P. V.: Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 3

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. 2, 8

[KH13] Kazhdan M., Hoppe H.: Screened poisson surface reconstruction. *ACM Trans. Graph. 32*, 3 (July 2013), 29:1–29:13. 2

[KLL15] Kim J., Lee J. K., Lee K. M.: Accurate image super-resolution using very deep convolutional networks. *CoRR abs/1511.04587* (2015). 5

[KMHG13] Kim Y. M., Mitra N. J., Huang Q.-X., Guibas L. J.: Guided real-time scanning of indoor objects. *Comput. Graph. Forum 32*, 7 (2013), 177–186. 2

[KMYG12] Kim Y. M., Mitra N. J., Yan D.-M., Guibas L.: Acquiring 3d indoor environments with variability and repetition. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 138:1–138:11. 2

[LCOLTE07] Lipman Y., Cohen-Or D., Levin D., Tal-Ezer H.: Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[LDGN15] Li Y., Dai A., Guibas L., Niessner M.: Database-assisted object retrieval for real-time 3d reconstruction. *Computer Graphics Forum 34*, 2 (2015). 2

[LPS*16] Li Y., Pirk S., Su H., Qi C. R., Guibas L. J.: FPNN: field probing neural networks for 3d data. *CoRR abs/1605.06240* (2016). 3

[MBBV15] Masci J., Boscaini D., Bronstein M. M., Vandergheynst P.: Geodesic convolutional neural networks on riemannian manifolds. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)* (Dec 2015), pp. 832–840. 3

[NXS12] Nan L., Xie K., Sharf A.: A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 137:1–137:10. 2

[OAG10] Öztireli A. C., Alexa M., Gross M.: Spectral sampling of manifolds. In *ACM SIGGRAPH Asia 2010 papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, ACM, pp. 168:1–168:8. 2

[OGG09] Öztireli A. C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum 28*, 2 (2009), 493–501. 2, 7, 8

[PGK02] Pauly M., Gross M., Kobbelt L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 163–170. 7

[PMA*14] Preiner R., Mattausch O., Arikan M., Pajarola R., Wimmer M.: Continuous projection for fast l1 reconstruction. *ACM Trans. Graph. 33*, 4 (July 2014), 47:1–47:13. 2

[PMG*05] Pauly M., Mitra N. J., Giesen J., Gross M., Guibas L. J.: Example-based 3d scan completion. In *Proceedings of the Third Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), SGP '05, Eurographics Association. 2

[QSMG17] Qi C. R., Su H., Mo K., Guibas L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017). 2, 3, 4, 7

[QSN*16] Qi C. R., Su H., Niessner M., Dai A., Yan M., Guibas L. J.: Volumetric and multi-view cnns for object classification on 3d data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 5648–5656. 3

[QYSG17] Qi C. R., Yi L., Su H., Guibas L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* (2017). 3

[RUG17] Riegler G., Ulusoy A. O., Geiger A.: Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017). 3

[SFCH12] Shen C.-H., Fu H., Chen K., Hu S.-M.: Structure recovery by part assembly. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 180:1–180:11. 2

[SGF16] Sharma A., Grau O., Fritz M.: Vconv-dae: Deep volumetric shape learning without object labels. *Computer Vision – ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III* (2016), 236–250. 3

[SKAG15] Sung M., Kim V. G., Angst R., Guibas L.: Data-driven structural priors for shape completion. *ACM Trans. Graph. 34*, 6 (Oct. 2015). 2

[SMKLM15] Su H., Maji S., Kalogerakis E., Learned-Miller E. G.: Multi-view convolutional neural networks for 3d shape recognition. *CoRR abs/1505.00880* (2015). 3

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 896–904. 2

[SSW15] Sun Y., Schaefer S., Wang W.: Denoising point sets via l 0 minimization. *Comput. Aided Geom. Des. 35*, C (May 2015), 2–15. 2

[SXZ*12] Shao T., Xu W., Zhou K., Wang J., Li D., Guo B.: An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 136:1–136:11. 2

[VDR*16] Varley J., DeChant C., Richardson A., Ruales J., Allen P.: Shape Completion Enabled Robotic Grasping. *ArXiv e-prints* (Sept. 2016). 3

[WBLP11] Weise T., Bouaziz S., Li H., Pauly M.: Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 4 (July 2011), 77:1–77:10. 3

[WHG*15] Wu S., Huang H., Gong M., Zwicker M., Cohen-Or D.: Deep points consolidation. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 176:1–176:13. 2

[WLT16] Wang P.-S., Liu Y., Tong X.: Mesh denoising via cascaded normal regression. *ACM Transactions on Graphics (SIGGRAPH Asia) 35*, 6 (2016). 7

[WP15] Wang D. Z., Posner I.: Voting for voting in online point cloud object detection. In *Robotics: Science and Systems* (2015), Kavraki L. E., Hsu D., Buchli J., (Eds.). 3

[WSK*15] Wu Z., Song S., Khosla A., Yu F., Zhang L., Tang X., Xiao J.: 3d shapenets: A deep representation for volumetric shapes. In *CVPR* (2015), IEEE Computer Society, pp. 1912–1920. 3

[XRY*15] Xu L., Ren J. S. J., Yan Q., Liao R., Jia J.: Deep edge-aware filters. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37* (2015), ICML'15, JMLR.org, pp. 1669–1678. 2

[XZZ*14] Xiong S., Zhang J., Zheng J., Cai J., Liu L.: Robust surface reconstruction via dictionary learning. *ACM Transactions on Graphics (Proc. SIGGRAPH Aisa) 33* (2014). 2

[YZW*16] Yan Z., Zhang H., Wang B., Paris S., Yu Y.: Automatic photo adjustment using deep neural networks. *ACM Trans. Graph. 35*, 2 (Feb. 2016), 11:1–11:15. 2

[ZZC*16] Zhang K., Zuo W., Chen Y., Meng D., Zhang L.: Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *CoRR abs/1608.03981* (2016). 5