Learning Self-prior for Mesh Denoising using Dual Graph Convolutional Networks

Shota Hattori, Tatsuya Yatagawa, Yutaka Ohtake, and Hiromasa Suzuki

School of Engineering, The University of Tokyo
{hattori,tatsy,ohtake,suzuki}@den.t.u-tokyo.ac.jp

Abstract. This study proposes a deep-learning framework for mesh denoising from a single noisy input, where two graph convolutional networks are trained jointly to filter vertex positions and facet normals apart. The prior obtained only from a single input is particularly referred to as a self-prior. The proposed method leverages the framework of the deep image prior (DIP), which obtains the self-prior for image restoration using a convolutional neural network (CNN). Thus, we obtain a denoised mesh without any ground-truth noise-free meshes. Compared to the original DIP that transforms a fixed random code into a noise-free image by the neural network, we reproduce vertex displacement from a fixed random code and reproduce facet normals from feature vectors that summarize local triangle arrangements. After tuning several hyper-parameters with a few validation samples, our method achieved significantly higher performance than traditional approaches working with a single noisy input mesh. Moreover, its performance is better than the other methods using deep neural networks trained with a large-scale shape dataset. The independence of our method of either large-scale datasets or ground-truth noise-free mesh will allow us to easily denoise meshes whose shapes are rarely included in the shape datasets. Our code is available at: https://github.com/astaka-pe/Dual-DMP.git.

Keywords: mesh denoising, self-prior, graph convolutional network

1 Introduction

The prevalence of low-cost three-dimensional (3D) scanning devices has made digitizing the shapes of real-world objects easier. However, the surface geometries captured by these 3D scanners are inevitably noisy, even when using the latest scanning devices, due to weak light reflection caused by the specularity and dark color of the object surface. Therefore, for accurate geometry processing and aesthetic reasons, denoising the surface data is essential in subsequent vision and graphics applications. A central problem of mesh denoising is the isolation of geometric features from noise. To this end, a large number of studies have been conducted for mesh denoising, including filter-based [4,9,14,22,26,39,40,42,45], feature-extraction-based [1,19,34], and optimization-based [8,32,44] approaches.

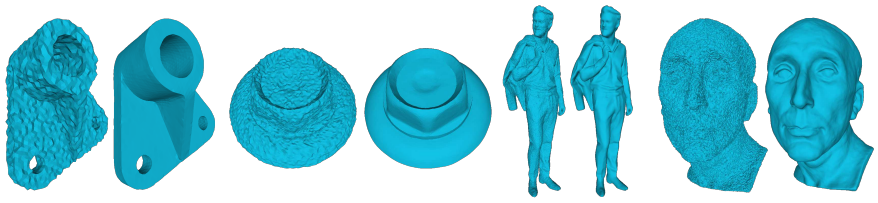

Fig. 1. Our method performs learning-based denoising only using a noisy input mesh, optimizing both the vertex positions and facet normals without using any training datasets. As shown, it works well for both the mechanical parts with sharp geometric features and human bodies, which are typical applications of 3D digital scanners.

However, despite the long continuous efforts, the isolation of geometric features is still challenging, especially in the presence of high-level noise.

Recently, in other research areas, deep-learning techniques have raised a new trend in data-driven approaches even for mesh denoising. To our knowledge, most existing methods in this kind regress the noise-free normals from different inputs, such as handmade local geometric features [30, 31, 43] and learned features encoded by a neural network [15, 17, 24, 43]. After the noise-free normals are obtained, the output vertex positions are reproduced by the standard iterative vertex updating [25], as performed by the traditional normal filtering methods [40, 45]. However, this vertex updating process is not included in their network architectures because it causes a huge computation graph and can significantly slow down the backpropagation. Moreover, the vertex updating as simple postprocessing can flip faces over and obtain wrong vertex positions.

In contrast to these data-driven approaches to filtering facet normals, this study proposes the dual deep mesh prior (DDMP), which learns a self-prior for mesh denoising using graph convolutional networks (GCNs). Compared to previous approaches, our method has two advantages. First, as the name of DDMP indicates, our method trains the pair of graph convolutional networks on dual graphs that filter vertex positions and facet normals, respectively. Our method synchronously trains two GCNs for positions and normals with the position-normal consistency loss. In this way, our method does not need the iterative vertex updating and enables the network to learn the mesh denoising in an end-to-end manner. Second, our method does not use a large shape dataset and acquires a prior for mesh denoising from a single input mesh. We refer to such a prior acquired with a single input as "self-prior." To this end, our method extends the deep image prior (DIP) [29], which also acquires a self-prior for image restoration, and denoises both vertex positions and facet normals. Therefore, our method does not need time-consuming pretraining of the network using a large dataset and can obtain a specialized result for each input shape. Furthermore, our method without a training dataset has been demonstrated to achieve denoising better than other supervised learning methods.

2 Related Work

Traditional mesh denoising. Mesh denoising is the task of filtering 3D signals such as vertex positions and normals. Hence, the previous approaches have been inspired by the techniques for image denoising. Early approaches for mesh denoising have processed vertex positions by spatial filtering [5,9,26], energy minimization [4], and iterative error diffusion [3] but have been followed by a large number of approaches based on normal filtering. These approaches by normal filtering smooth either vertex or facet normals and reproduce noise-free meshes by iteratively updating vertex positions with the filtered normals [21, 25, 27]. In the past, various filters have been applied to smooth noisy normals such as Gaussian filter [22], median filter [39], bilateral filter [12, 45], and other anisotropic filters [35, 40, 42]. When compared to filtering vertex positions, methods that used normal filtering are known to reproduce the sharp local geometries more accurately. However, these methods often cannot reproduce shape features well when they are scarcely recognized due to much noise. In contrast, optimization-based approaches solve an energy minimization problem to satisfy a set of priors imposed on the ground-truth geometry or noise patterns. They formulated the optimization problem based on techniques such as L_0-regularized minimization [8, 44], compressed sensing [32], and low-rank recovery [16, 33]. However, these heuristic priors cannot always be applied to shapes that do not conform to the ground-truth geometry or noise patterns to be assumed.

Learning-based mesh denoising. With the increasing use of learning-based denoising for images, mesh denoising has gained increasing attention. The pioneering study by Wang et al. [31] used a simple multilayer perceptron (MLP) to regress noise-free normals from the descriptor extracted by processing input facet normals with noise. This method was followed by a two-step filtering approach [30] where the first step was equivalent to the prior one [31] and the second step enhanced geometric details of the shape obtained by the first step. Zhao et al. [43] introduced a new geometric descriptor based on the local volume around a vertex of interest and applied 3D convolution to regress noise-free normals. Li et al. [17] instead employed a non-local patch-group normal matrix [16, 33] to regress the clean normals. DNF-Net [17] directly encoded the vertex positions and adjacency matrix by a neural network and regressed the clean facet normals from the encoded features. In contrast to these methods that extract geometric features using standard neural networks, GCN-Denoiser [24] scissored an input mesh into local patches and regressed the clean normals from the patch with a GCN. GeoBi-GNN [41] is akin to GCN-Denoiser, but GeoBi-GNN used two GCNs. The former filters noisy vertex positions in advance to normal regression, and the latter further filters the normals computed with filtered positions. Then, the final vertex positions are reproduced by an ordinary iterative process using both the filtered positions and normals.

Learning deep prior without ground-truth data. Current deep neural networks are usually composed of convolutional layers with learnable weights associated with a convolution kernel. Each set of weights is trained to extract

a structural pattern commonly appearing in the input data since the weight parameters are shared over an entire data domain. Such a prior acquired by a deep neural network is referred to as a "deep prior," and it is acquired even from a single input data. DIP [29] is a seminal approach that acquires the deep prior from a single input and performs various image restoration tasks, such as denoising and inpainting, only from a corrupted input image. The DIP paper revealed that the convolutional layer can learn structural patterns more quickly than random patterns such as white noise. This nature of convolutional layers is useful to obtain the self-prior and restore images without using clean data.

For shape data, Williams et al. [37] proposed Deep Geometric Prior (DGP) for reconstructing the surface geometry from a point cloud. DGP trains several MLPs individually to reconstruct different local surface patches. In contrast, Point2Mesh [7] leverages a self-prior to reconstruct the entire surface from an input point cloud without any pretraining using a training dataset. In their method, individual MeshCNNs [6] are trained to acquire the self-priors of the shape in different triangle resolutions. The higher-dimensional information with surface geometry with colors can also be reproduced by self-priors [36]. Although these approaches require only a single point cloud to reconstruct its surface geometry, they can fail to capture global features over the entire surface or resolution because the base networks are individually trained for different patches and resolutions.

3 Dual Deep Mesh Prior

Figure 2 shows the architecture of DDMP. The input is a noisy mesh $\mathcal{M}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges of the mesh. Based on the adjacencies of vertices and faces, we define dual graphs $\mathcal{G}_v$ and $\mathcal{G}_f$ whose nodes correspond to vertices and faces. On each graph, we define a GCN that denoises either vertex positions or facet normals. We refer to these GCNs for positions and normals as PosNet and NormNet, respectively. The denoised positions and normals are evaluated in terms of their reproducibility by E_{pos} and E_{nrm}, smoothness by E_{Lap} and E_{bnf}, and consistency by E_{con}. Specifically, our method evaluates the reproducibility with E_{pos} and E_{nrm} against the noisy input and works without the ground-truth data.

A previous method that is most similar to ours is GeoBi-GNN [41], which defines GCNs on dual graphs and filters both the vertex positions and facet normals. However, GeoBi-GNN needs the iterative vertex updating, which is not included in the network architecture. In contrast, our method is truly an end-to-end approach without requiring the iterative process [25] for recovering the final vertex positions. Furthermore, our method works well without either a large-scale shape dataset or time-consuming pretraining of the neural network.

3.1 Good self-prior for shape attributes

Before explaining the proposed method, we quickly review DIP [29], by which our paper is inspired. DIP successfully removes noise from an input corrupted image

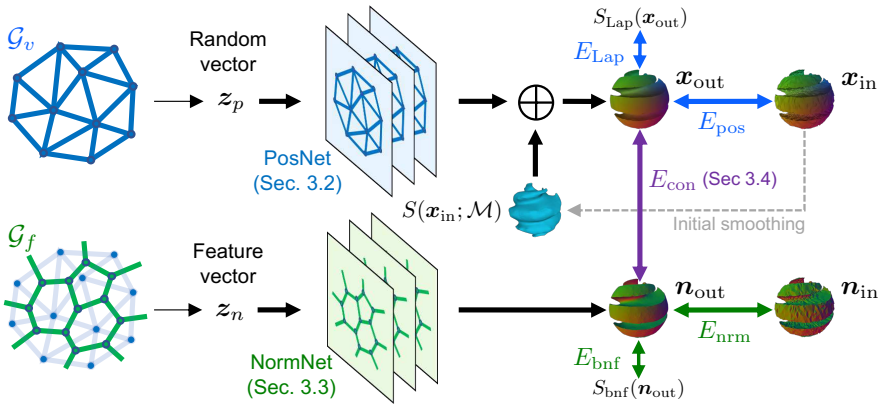

Fig. 2. DDMP builds dual graphs of the input mesh and associates two GCNs with them. The two GCNs, namely, PosNet and NormNet, are respectively in charge of filtering vertex positions and facet normals. Output positions and normals are evaluated into three groups of loss functions: (1) the first group evaluates vertex positions, (2) the second group evaluates facet normals, and (3) the last group evaluates the consistency of vertex positions and facet normals.

without using either a large-scale image dataset or a ground-truth clean image. The network of DIP transforms a static random code $\boldsymbol{z}$ to the restored image $\boldsymbol{x}_{\mathrm{out}}$ and is trained to minimize the difference between $\boldsymbol{x}_{\mathrm{out}}$ and the noisy input image $\boldsymbol{x}_{\mathrm{in}}$. Let F be the map defined by the network with a set of parameters Θ. Then, DIP trains the network to obtain the best parameter set Θ^*:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left\| \boldsymbol{x}^i_{\mathrm{out}} - \boldsymbol{x}^i_{\mathrm{in}} \right\|_2^2, \qquad \text{where} \quad \boldsymbol{x}_{\mathrm{out}} = F(\boldsymbol{z}; \Theta), \tag{1}$$

where $\mathcal{I}$ is an index set of image pixels, $|\mathcal{I}|$ is the size of $\mathcal{I}$ (i.e., the number of image pixels), $\boldsymbol{x}^i$ is the color of the ith pixel, and $\|\cdot\|_p$ is the ℓ_p-norm of a vector.

The DIP paper [29] shows that its capability of image restoration is reasoned by the nature of neural networks, where the network learns structural patterns more quickly than random patterns. Therefore, the presence of structural patterns must be considered to apply the framework of DIP to shape attributes, such as vertex positions and facet normals. Figure 3 shows dinosaur models colorized based on vertex positions and facet normals. In Fig. 3(a), the model is fit to a unit cube $[0, 1]^3$ by keeping its aspect ratio, and each vertex is colorized based on its position (i.e., $(r, g, b) = (x, y, z)$). However, the vertex colors in Fig. 3(a) are not structural, changing gradually over the surface and not corresponding to the bumps on its back. In contrast, the vertices in Fig. 3(b) are colored by the displacements from the smoothed mesh at the top left. A color pattern is repeated to follow the repetition of the bumps on the back. In light of these observations, a network will learn the displacements more easily than the positions themselves. When colorizing the faces with their normals (i.e.,

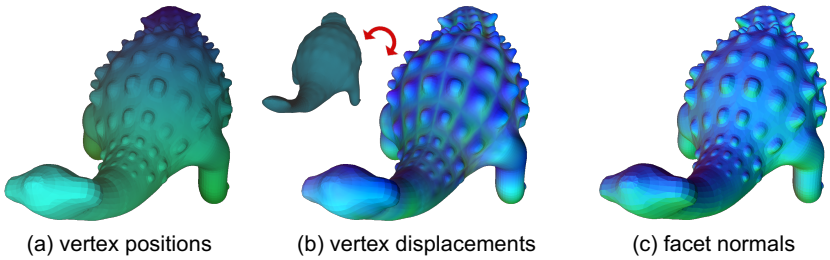

Fig. 3. Colorized meshes by (a) vertex positions, (b) vertex displacements from smoothed mesh, and (c) facet normals. For vertex positions, the displacement from a smoothed mesh is more appropriate to elicit the structural pattern, whereas the facet normals form a structural pattern without any special treatment.

$(r, g, b) = (\frac{n_x+1}{2}, \frac{n_y+1}{2}, \frac{n_z+1}{2})$), we can see the repetition of a color pattern on the bumps in Fig. 3(c). Hence, facet normals will be reconstructed due to the nature of convolutional layers without any special treatment.

The above observations suggest that PosNet and NormNet should be trained differently. We train PosNet to reproduce the displacements of vertices between a smoothed mesh and the noisy input mesh. On the other hand, we train NormNet to reproduce the facet normals as the original DIP does.

3.2 PosNet: Position filtering network

As previously discussed, we train PosNet to estimate the vertex displacements from the smoothed shape. PosNet consists of a stack of 12 graph convolutional layers [10] followed by 2 fully connected layers. As the original DIP does, PosNet F_p with parameters Θ_p transforms a tensor of static random codes $\boldsymbol{z}_p \in \mathbb{R}^{N_p \times D_p}$ to another tensor of vertex displacement vector $F_p(\boldsymbol{z}_p; \mathcal{M}, \Theta_\mathrm{p}) \in \mathbb{R}^{N_p \times 3}$, where N_p is the number of vertices, D_p is the number of dimensions of the random codes. To avoid that a mesh scale affects the result, we normalize the size of an input mesh by the average length of edges l_e. Then, the displacement is added to a smoothed mesh $S(\boldsymbol{x}_\mathrm{in}; \mathcal{M}) \in \mathbb{R}^{N_p \times 3}$ to obtain a clean output, where $\boldsymbol{x}_\mathrm{in} \in \mathbb{R}^{N_p \times 3}$ is a tensor of input vertex positions. Specifically, the output tensor of vertex positions $\boldsymbol{x}_\mathrm{out} \in \mathbb{R}^{N_p \times 3}$ is expressed as

$$\boldsymbol{x}_\mathrm{out} = S(\boldsymbol{x}_\mathrm{in}; \mathcal{M}) + F_p(\boldsymbol{z}_p; \mathcal{M}, \Theta_p). \tag{2}$$

The output positions were evaluated by two error functions. One is a reconstruction error defined with the root-mean-squared error (RMSE):

$$E_\mathrm{pos}(\boldsymbol{x}_\mathrm{out}, \boldsymbol{x}_\mathrm{in}) = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} \|\boldsymbol{x}^i_\mathrm{out} - \boldsymbol{x}^i_\mathrm{in}\|_2^2}, \tag{3}$$

where $\boldsymbol{x}^i$ is an ith row of the tensor. In addition, we employed a Laplacian error for vertex positions to encourage the smoothness of the output mesh.

$$E_{\text{Lap}}(\boldsymbol{x}_{\text{out}}) = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}\left\|\boldsymbol{x}^i_{\text{out}} - S_{\text{Lap}}(\boldsymbol{x}^i_{\text{out}})\right\|_2^2}, \quad S_{\text{Lap}}(\boldsymbol{x}^i_{\text{out}}) = \frac{1}{|\mathcal{V}_v(i)|}\sum_{j\in\mathcal{V}_v(i)}\boldsymbol{x}^j_{\text{out}}, \tag{4}$$

where $\mathcal{V}_v(i)$ is an index set for neighboring vertices around the ith vertex. As shown later, this error also serves as a regularization term for PosNet that prevents overfitting to the noisy input.

3.3 NormNet: Normal filtering network

Discussions in Sec. 3.1 showed facet normals can represent structural patterns of local surface geometry without any special treatment. Accordingly, we train NormNet F_n to reproduce clean facet normals as in the original DIP [29]. The architecture of NormNet is the same as that of PosNet (i.e., it has 12 graph convolutional layers followed by 2 fully connected layers), but its graph convolutional layers are defined on a dual graph where each node corresponds to a face. Besides, each output is normalized to be a unit vector.

Different from PosNet, we supply a set of handmade feature vectors to NormNet rather than the random codes to PosNet. The feature vector summarizes the geometric features of a face. For a face indexed by k, the feature vector includes barycenter $\boldsymbol{c}^k_{\text{in}}$, normal $\boldsymbol{n}^k_{\text{in}}$, and area A^k_{in}, which are computed for the noisy input mesh. Refer to the supplementary document for the experimental validation of the choice of feature elements.

Let $\boldsymbol{z}_n \in \mathbb{R}^{N_f \times D_n}$ be a tensor of the face features where N_f is the number of faces, and D_n is the number of dimensions of the facet features. Then, the clean output normals $\boldsymbol{n}_{\text{out}} \in \mathbb{R}^{N_f \times 3}$ are represented as

$$\boldsymbol{n}_{\text{out}} = F_n(\boldsymbol{z}_n; \mathcal{M}, \Theta_n), \tag{5}$$

where Θ_n is a set of parameters of NormNet. Although the RMSE is used to train PosNet, we employ a reconstruction error using the mean absolute error (MAE) to let NormNet more sensitive to sharp geometric features.

$$E_{\text{nrm}}(\boldsymbol{n}_{\text{out}}, \boldsymbol{n}_{\text{in}}) = \frac{1}{N_f}\sum_{i=1}^{N_f}\|\boldsymbol{n}^i_{\text{out}} - \boldsymbol{n}^i_{\text{in}}\|_1. \tag{6}$$

Additionally, we define another error function to evaluate the smoothness of output facet normals. The smoothness error is defined based on bilateral filter [28] that has been widely used for mesh smoothing [12, 45] and encourages the smoothness of the normals while preserving the local surface geometry such as sharp edges. We calculate the MAE between the output facet normals and those smoothed by the bilateral filter.

$$E_{\text{bnf}}(\boldsymbol{n}_{\text{out}}) = \frac{1}{N_f}\sum_{i=1}^{N_f}\left\|\boldsymbol{n}^i_{\text{out}} - S^{(t)}_{\text{bnf}}(\boldsymbol{n}^i_{\text{out}})\right\|_1, \tag{7}$$

where $S_{\mathrm{bnf}}^{(t)}$ means the bilateral filter S_{bnf} is applied t times. In our experiment, we empirically set $t = 5$ for CAD models that often contain sharp features and $t = 1$ for non-CAD models. As with E_{Lap} for PosNet, E_{bnf} serves as a regularization term that prevents overfitting to the noisy input.

3.4 Position and normal consistency

Although PosNet and NormNet filter noise on vertex positions and facet normals of the input mesh independently, these results can be inconsistent. Therefore, we may need an iterative vertex updating process [21,25,27] to make them consistent, as performed in many previous methods. However, this iterative process results in a large computation graph needed for the backpropagation and can slow down the network training significantly, which hinders the whole training process from being performed in an end-to-end manner. Furthermore, individual filters for position and normals by our PosNet and NormNet can be invalid due to the absence of the ground-truth noise-free mesh during training, resulting in flipped and self-intersected triangles.

To resolve these problems, we employ a position-normal consistency loss based on the formulation of the iterative vertex updating [25]. This approach updates the vertex positions based on the following updating rule:

$$\boldsymbol{x}_{\mathrm{new}}^{i} = \boldsymbol{x}^{i} + \frac{1}{3|\mathcal{F}_v(i)|} \sum_{j \in \mathcal{F}_v(i)} \boldsymbol{n}^{j} (\boldsymbol{n}^{j} \cdot (\boldsymbol{c}^{j} - \boldsymbol{x}^{i})), \tag{8}$$

where $\mathcal{F}_v(i)$ is a set of face indices around the ith vertex, and $\boldsymbol{c}^j$ is the centroid of the jth face. The second term of Eq. (8) represents an amount of vertex displacement during the update and is converged to zero when the update is terminated. This means that the second term can measure the consistency between given positions and normals of adjacent faces. Accordingly, we define a consistency error between the output vertex positions of PosNet and the output facet normals of NormNet by summing up the displacements.

$$E_{\mathrm{con}}(\boldsymbol{x}_{\mathrm{out}}, \boldsymbol{n}_{\mathrm{out}}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{j \in \mathcal{F}_v(i)} \boldsymbol{n}_{\mathrm{out}}^{j} \cdot (\boldsymbol{c}_{\mathrm{out}}^{j} - \boldsymbol{x}_{\mathrm{out}}^{i}). \tag{9}$$

Different from the original rule in Eq. (8), the normalizing constant $3|\mathcal{F}_v(i)|$ is omitted in the above equation for simplicity. This means that a vertex surrounded by more faces will be penalized more heavily for its displacement, but this change did not matter to the results significantly.

3.5 Loss function

Overall, we train PosNet and NormNet synchronously using the error:

$$E = k_1 E_{\mathrm{pos}} + k_2 E_{\mathrm{Lap}} + k_3 E_{\mathrm{nrm}} + k_4 E_{\mathrm{bnf}} + k_5 E_{\mathrm{con}}, \tag{10}$$

Table 1. Quantitative comparison of AAD and AHD values. In each cell, AAD and AHD are written to the left and right. The AHDs are in the unit of 10^{-3} and are calculated after the size is normalized by the diagonal length of a bounding box.

Model	Type	Input	BNF [45]	GNF [40]	DNF [15]	GCN-D [24]	DDMP
block	CAD	33.72°/4.32	7.06°/1.26	4.11°/0.99	4.13°/0.99	4.19°/0.98	**2.90°/0.74**
fandisk	CAD	28.42°/3.23	3.49°/0.76	2.93°/0.68	3.33°/0.69	3.28°/0.70	**1.93°/0.49**
nut	CAD	22.23°/1.61	5.70°/0.78	4.05°/0.55	4.21°/0.52	3.64°/0.50	**3.09°/0.44**
part-lp	CAD	18.87°/2.43	2.07°/0.66	2.59°/0.69	2.84°/0.67	2.40°/0.62	**1.78°/0.46**
trim-star	CAD	29.36°/3.42	6.57°/1.16	6.95°/1.11	5.55°/0.93	4.86°/0.89	**4.71°/0.84**
ankylo.	non-CAD	14.40°/0.58	6.37°/0.44	8.05°/0.46	5.69°/0.33	**4.59°**/0.31	4.68°/**0.29**
fertility	non-CAD	19.39°/1.35	5.54°/0.72	5.70°/0.58	3.83°/0.44	**2.68°/0.34**	3.40°/0.45
nicolo	non-CAD	30.92°/0.92	5.48°/0.29	6.79°/0.33	5.17°/0.26	**4.73°/0.25**	5.11°/0.26
suit-man	non-CAD	13.44°/0.33	8.45°/0.42	6.57°/0.22	5.22°/0.20	**4.47°/0.17**	5.26°/0.21

where $\mathcal{K} = (k_1, k_2, k_3, k_4, k_5)$ is a set of hyperparameters (i.e., the weights for the error terms). As with the data-driven approaches using deep learning, these hyperparameters affect the denoising performance of our method. Based on the result of hyperparameter tuning by Ray Tune [18], we set $\mathcal{K} = (3, 0, 3, 4, 2)$ for CAD models, whereas $\mathcal{K} = (3, 4, 4, 4, 1)$ are set for non-CAD models. Only four meshes (i.e., sharp-sphere, twelve, carter, and grayloc), which are not used in performance comparisons, were used in the tuning process[1]. In addition, we found that the output mesh can be extremely noisy at a relatively early stage in training and that E_{bnf} can retain high-frequency noise unintentionally. Therefore, we set the weight k_4 for E_{bnf} to zero during the first 100 steps to prevent the facet normals from converging to a local minimum different from the desired ones. We show the details of this experiment in the supplementary document.

4 Experiments

Our method is implemented using PyTorch and PyTorch Geometric and is tested on a computer equipped with Intel Core i7 5930K CPU (3.5 GHz, 6 cores), NVIDIA GeForce TITAN X (12 GB graphics memory), and 64 GB of RAM. PosNet and NormNet are trained over 1000 steps by Adam optimizer with learning rate of $\gamma = 0.01$ and decay parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. Following GCN-Denoiser [24], we used two metrics for evaluation, the average normal angular difference (AAD) and the average one-sided Hausdorff distance (AHD) between the ground-truth and the output. Note that AHDs are normalized by the diagonal length of a bounding box of the ground-truth mesh. A smaller value for both of them means better denoising performance. In the following figures, we visualize the denoised meshes by colorizing each face based on the normal angular difference. We also show the specific value θ of the AAD below each mesh.

Results on artificial noise. We demonstrate the denoising performance of our method for additive white Gaussian noise (AWGN), whose magnitude is set as a multiple of the average edge length l_e of each mesh. The results are compared with previous studies, including bilateral normal filtering (BNF) [45],

[1] All the meshes used in this study are shown in the supplementary document.

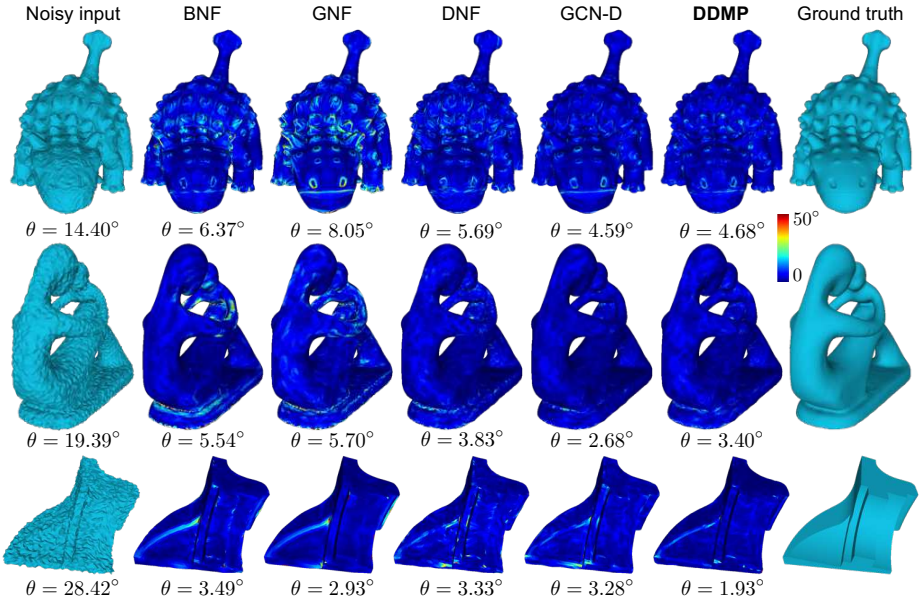

Fig. 4. Visual comparison of output meshes between our method and prior methods. AWGNs with different noise levels (i.e., $0.1l_e, 0.2l_e, 0.3l_e$) are added to three meshes from top to bottom.

guided normal filtering (GNF) [40], deep normal filtering network (DNF) [15], and mesh denoising with graph convolutional networks (GCN-D) [24]. For BNF and GNF (i.e., the traditional approaches that do not use deep learning), we tuned the hyperparameters manually. On the other hand, for DNF and GCN-D (i.e., the state-of-the-art approaches using deep learning), we used pretrained weights provided by the authors. These methods were tested with nine meshes provided by the authors of cascaded normal regression (CNR) [31] and GCN-D [24], which are classified into CAD models with many sharp features and non-CAD models with a few ones.

Table 1 shows the quantitative analysis for the meshes, and Fig. 4 shows the visual comparisons for some of them. They show that traditional approaches often oversharpen edges or oversmooth fine details, while our method removes noise while preserving both the sharp edges and fine details. Furthermore, although our method uses only a noisy input mesh, its performance is comparable with or better than the various methods using large training datasets. This tendency is more apparent in CAD models, which shows that our approach accommodates each input and keeps sharp edges and fine details.

Figure 5 shows the robustness of our method for spatially varying density of triangles, where densely triangulated regions are marked by red circles. For the "block" mesh in this figure, we added noise uniformly over the mesh surface, which means that the relative noise level is higher in the densely triangulated

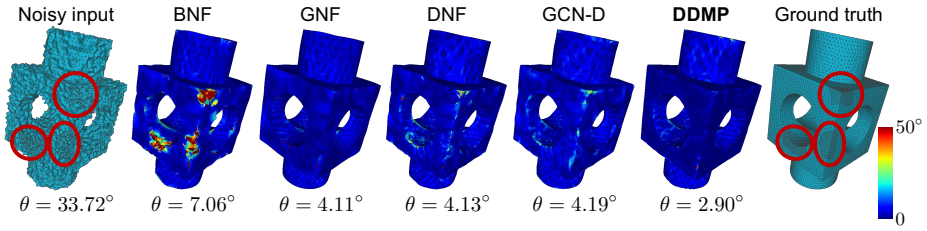

Fig. 5. Robustness to non-uniform density of triangles. The noisy input mesh and noise-free counterpart are shown to the both sides and are marked by red circles to show the densely triangulated regions. The noise level is $0.4l_e$ for this "block" example.

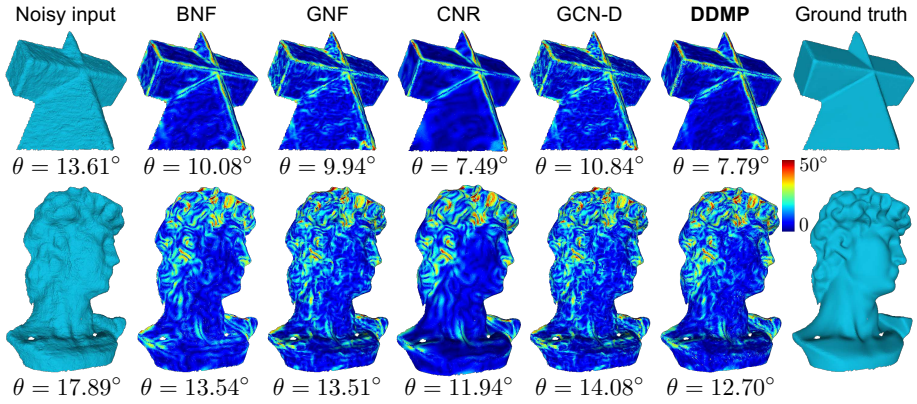

Fig. 6. Visual comparison of denoising performances for meshes obtained by real scans. The meshes are provided by the authors of the CNR paper [31].

regions. Such regions with a high noise level are not properly denoised by most previous approaches, while GNF is robust to the triangle density but oversharpens edges. Compared to these approaches, our method obtains the lowest AAD, working more robustly to the triangle density and keeping sharp edges.

Results on real scans. Next, we evaluate our method using meshes obtained by real scans. The meshes used here are provided by the authors of CNR [31] that are acquired by KinectFusion [20] with Microsoft Kinect v1. For GCN-D, we used a pretrained model for artificial noise, the only one available publicly. In this experiment, our method uses the hyperparameters $\mathcal{K} = (3, 4, 4, 4, 1)$ for non-CAD models. Training only for 50 steps was sufficient for these meshes because their noise levels are comparatively less than AWGN used previously. In addition, the noise that we can see in Fig. 6 behaves differently from AWGN and consists of both the low- and high-frequency components. Hence, the denoising performances of previous approaches in Tab. 2 are decreased compared to those for AWGN, except for CNR learning such particular noise. Nevertheless, for such

Table 2. Quantitative comparison for meshes obtained by real scans. Despite the absence of training with the ground-truth meshes, our method achieved as low AADs as CNR [31] that learns the ground-truth data.

Model	Input	BNF	GNF	CNR	GCN-D	DDMP
pyramid	13.61°/2.21	10.08°/2.18	9.94°/2.20	**7.49°/2.13**	10.84°/2.21	7.79°/2.16
david	17.89°/1.86	13.54°/**1.85**	13.51°/**1.85**	**11.94°**/1.86	14.08°/**1.85**	12.70°/1.88

Table 3. Quantitative comparison of DDMP with its variants where some of the errors have been ablated. For NormNet, NormNet w/o E_{bnf}, and DDMP w/o E_{con}, AAD and AHD are computed for output meshes recovered by posterior iterative vertex updating.

	PosNet	PosNet w/o E_{Lap}	NormNet	NormNet w/o E_{bnf}	DDMP w/o E_{con}	DDMP
E_{pos}	✓	✓			✓	✓
E_{Lap}	✓				✓	✓
E_{nrm}			✓	✓	✓	✓
E_{bnf}			✓		✓	✓
E_{con}						✓
sharp	—	15.60°/1.77	5.00°/1.43	9.55°/1.53	5.44°/1.44	**4.90°/1.40**
twelve	—	15.64°/2.14	**1.02°**/0.40	7.47°/1.36	1.04°/0.60	1.20°/**0.36**
carter	4.96°/0.31	9.27°/0.43	4.89°/0.32	**4.88°**/0.32	5.09°/0.33	4.95°/**0.31**
grayloc	9.19°/1.60	26.22°/2.03	10.50°/1.63	14.05°/1.74	8.89°/1.59	**7.25°/1.56**

a challenging scenario, our method achieved the second best AAD next to CNR and outperformed the other approaches, including GCN-D using deep learning. These results show the dependency of previous learning-based approaches on the distribution of training datasets, as well as the advantage of our method independent of datasets. In contrast, AHDs of the input noisy meshes were not decreased significantly by all the compared approaches and ours. This is due to the non-zero average of positional displacements by real scanning, even though most compared approaches including ours implicitly assume zero-mean noise.

Effect of error terms. We conduct an ablation study to verify the effect of each error term in Eq. (10). Table 3 shows a list of test cases in which some of the error terms are ablated. In each test case, AAD and AHD are evaluated for sample meshes. For NormNet, NormNet w/o E_{bnf}, and DDMP w/o E_{con}, we perform the iterative vertex updating [25] to obtain output meshes. Since "sharp-sphere" and "twelve" models are categorized as non-CAD models, and k_2 for non-CAD models are set to zero, the results of PosNet and PosNet w/o E_{Lap} will be exactly the same. Hence, we drew only horizontal lines to the cells for these two models processed by PosNet. As shown by Tab. 3, the performance of the full DDMP is better than those of both PosNet alone and NormNet alone. This result suggests that the synchronous training of our method allows the networks to interact with each other and obtains better results than those given by only one of two networks. Moreover, the full DDMP outperforms that without E_{con} using a non-learnable process of the iterative vertex updating. This means E_{con} works not only to enable the end-to-end learning of mesh denoising but also to enhance the interaction of PosNet and NormNet.

Moreover, we ablate error terms E_{Lap} and E_{bnf} from PosNet and NormNet to verify their effects. As supposed by the lower performance of PosNet w/o E_{Lap}

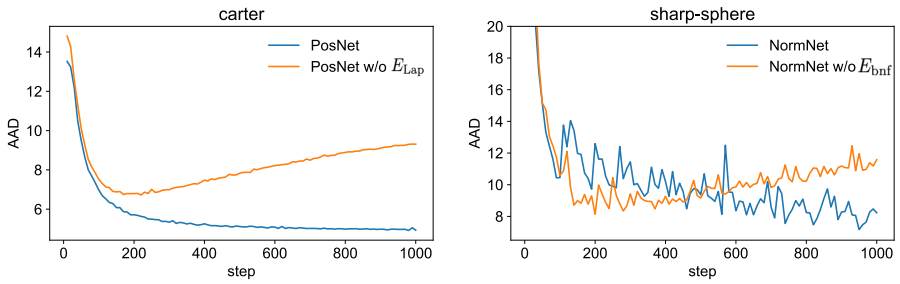

Fig. 7. Changes in AAD values during training PosNet and NormNet with and without smoothness error terms. These figures show the AAD values, which are unknown during training. Note that the AAD values of "sharp-sphere" shown in the right are computed for the direct output of NormNet before vertex updating.

and NormNet w/o E_{bnf} than those with the smoothness errors, these terms regularize the shape optimization and prevent it from being stuck to an inappropriate solution. Furthermore, the lack of E_{Lap} and E_{bnf} from each network causes overfitting to noise. As shown in Fig. 7, the AADs of PosNet w/o E_{Lap} and NormNet w/o E_{bnf} get higher after hundreds of steps, which shows the networks are overfitted to noise. Therefore, these regularization terms are helpful to obtain better results almost independently of when the training is terminated.

Effect of initial smoothing. Our method relies on a smoothing operation on the input mesh to extract the displacements at the vertex positions. To confirm the effect of smoothing, we compare the 30-step Laplacian smoothing "without" cotangent weights (LS-no-CW) [23], which we used in the above experiments, with three other options, i.e., no smooth, bilateral normal filtering (BNF), and 3-step Laplacian smoothing "with" cotangent weights (LS-CW). The denoised results for the different smoothing operations are shown in Fig. 8, in which the meshes after initial smoothing are also shown in the top-left insets. The quantitative comparison is also shown in Table 4. The comparisons show that BNF and LS-CW obtain better results for CAD models and non-CAD models, respectively. This means an initial smoothing that gives a closer shape to the noise-free mesh may obtain a better output mesh. In contrast, LS-no-CW performs well almost equally with BNF for CAD models and better than both the BNF and LS-CW for non-CAD models. During this experiment, we found that an input smoothed shape, which is very close to the desired output, can complicate the network to learn structural patterns from the small difference, and the network can be just an identity mapping. In light of this, LS-no-CW that we used in the above experiments is a good choice to achieve good denoising for a wide range of meshes, although BNF may be able to achieve better results for CAD models.

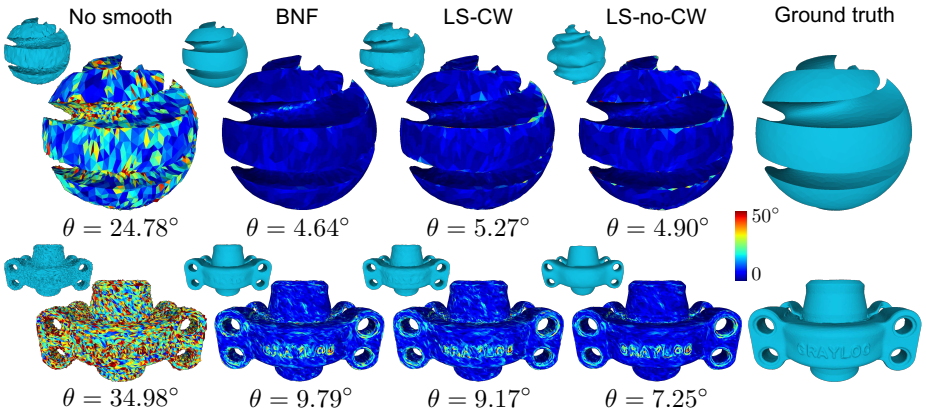

Fig. 8. Comparison of smoothed meshes obtained by different initial smoothing.

Table 4. Quantitative comparison for different initial smoothing operations.

Model	Type	Input	No smooth	BNF	LS-CW (3 itr.)	LS-no-CW (30 itr.)
sharp	CAD	24.81°/2.24	24.78°/2.24	**4.64°/1.34**	5.27°/1.35	4.90°/1.40
twelve	CAD	22.59°/2.74	22.42°/2.72	**0.99°/0.32**	1.47°/0.36	1.20°/0.36
carter	non-CAD	12.87°/0.48	12.87°/0.48	5.25°/**0.31**	4.98°/**0.31**	**4.95°/0.31**
grayloc	non-CAD	34.99°/2.32	34.98°/2.32	9.79°/1.58	9.17°/1.57	**7.25°/1.56**

5 Conclusion

This paper presented DDMP, a new mesh denoising framework using the self-prior acquired with a single input. DDMP optimizes both the vertex positions and facet normals synchronously with two GCNs (i.e., PosNet and NormNet) defined on dual graphs. These two networks can interact with each other to reduce noise while preserving sharp edges and fine details by introducing an error term to enhance position-normal consistency. Since the dual networks are trained with only a noisy input mesh, time-consuming pre-processing, such as patch generation and pretraining with large training datasets, is not required. Experimental results showed that our method achieved significantly higher performances than those of traditional approaches working only with a noisy input. Moreover, our method outperformed the state-of-the-art methods of learning many shapes.

In future work, we are interested in applying DDMP to other mesh restoration tasks, such as mesh completion and mesh simplification. To this end, we need further investigation on how an input mesh to DDMP should be obtained to share the same triangulation with an unknown desired output. We are also interested in combining DDMP with other self-supervised methods, such as Noise2Noise [13] and its variants [2,11,38], which propose their own data augmentation for noisy images and train the network without using ground-truth clean images.

Acknowledgment. This study is financially supported by a JSPS Grant-in-Aid for Early-career Scientists (22K17907).

References

1. Arvanitis, G., Lalos, A.S., Moustakas, K., Fakotakis, N.: Feature preserving mesh denoising based on graph spectral processing. IEEE Transactions on Visualization and Computer Graphics **25**(3), 1513–1527 (2019). https://doi.org/10.1109/tvcg.2018.2802926 1
2. Calvarons, A.F.: Improved Noise2Noise denoising with limited data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. pp. 796–805 (2021). https://doi.org/10.1109/cvprw53098.2021.00089 14
3. Clarenz, U., Diewald, U., Rumpf, M.: Anisotropic geometric diffusion in surface processing (2000). https://doi.org/10.1109/VISUAL.2000.885721 3
4. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). p. 317–324. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co. (1999). https://doi.org/10.1145/311535.311576 1, 3
5. Fleishman, S., Drori, I., Cohen-Or, D.: Bilateral mesh denoising. ACM Transactions on Graphics **22**(3), 950–953 (2003). https://doi.org/10.1145/882262.882368 3
6. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: MeshCNN: A network with an edge. ACM Transactions on Graphics **38**(4) (2019). https://doi.org/10.1145/3306346.3322959 4
7. Hanocka, R., Metzer, G., Giryes, R., Cohen-Or, D.: Point2Mesh: A self-prior for deformable meshes. ACM Transactions on Graphics **39**(4) (2020). https://doi.org/10.1145/3386569.3392415 4
8. He, L., Schaefer, S.: Mesh denoising via L0 minimization. ACM Transactions on Graphics **32**(4), 1–8 (2013). https://doi.org/10.1145/2461912.2461965 1, 3
9. Jones, T.R., Durand, F., Desbrun, M.: Non-iterative, feature-preserving mesh smoothing. ACM Transactions on Graphics **22**(3), 943–949 (2003). https://doi.org/10.1145/882262.882367 1, 3
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference on Learning Representations (ICLR) (2017) 6
11. Krull, A., Buchholz, T.O., Jug, F.: Noise2Void - Learning denoising from single noisy images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019). https://doi.org/10.1109/cvpr.2019.00223 14
12. Lee, K.W., Wang, W.P.: Feature-preserving mesh denoising via bilateral normal filtering. 2005 . https://doi.org/10.1109/CAD-CG.2005.40 3, 7
13. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., Aila, T.: Noise2Noise: Learning image restoration without clean data. In: Proceedings of International Conference on Machine Learning (ICML). pp. 2965–2974. PMLR (2018) 14
14. Li, T., Wang, J., Liu, H., gang Liu, L.: Efficient mesh denoising via robust normal filtering and alternate vertex updating. Frontiers of Information Technology & Electronic Engineering **18**(11), 1828–1842 (2017). https://doi.org/10.1631/FITEE.1601229 1
15. Li, X., Li, R., Zhu, L., Fu, C.W., Heng, P.A.: DNF-Net: a deep normal filtering network for mesh denoising. IEEE Transactions on Visualization and Computer Graphics (2020). https://doi.org/10.1109/TVCG.2020.3001681 2, 9, 10

16. Li, X., Zhu, L., Fu, C.W., Heng, P.A.: Non-local low-rank normal filtering for mesh denoising. Computer Graphics Forum **37**(7), 155–166 (2018). https://doi.org/10.1111/cgf.13556 3
17. Li, Z., Zhang, Y., Feng, Y., Xie, X., Wang, Q., Wei, M., Heng, P.A.: NormalF-Net: Normal filtering neural network for feature-preserving mesh denoising. Computer-Aided Design **127**, 102861 (2020). https://doi.org/10.1016/j.cad.2020.102861 2, 3
18. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018) 9
19. Lu, X., Deng, Z., Chen, W.: A robust scheme for feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics **22**(3), 1181–1194 (2016). https://doi.org/10.1109/TVCG.2015.2500222 1
20. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 127–136. IEEE (2011). https://doi.org/10.1109/ismar.2011.6092378 11
21. Ohtake, Y., Belyaev, A., Bogaevski, I.: Mesh regularization and adaptive smoothing. Computer-Aided Design **33**(11), 789–800 (sep 2001). https://doi.org/10.1016/s0010-4485(01)00095-1 3, 8
22. Ohtake, Y., Belyaev, A.G., Seidel, H.P.: Mesh smoothing by adaptive and anisotropic Gaussian filter applied to mesh normals. In: Proceeding of Vision, Modeling, and Visuallization (VMV). vol. 2, pp. 203–210. Citeseer (2002) 1, 3
23. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics **2**(1), 15 – 36 (1993). https://doi.org/10.1080/10586458.1993.10504266 13
24. Shen, Y., Fu, H., Du, Z., Chen, X., Burnaev, E., Zorin, D., Zhou, K., Zheng, Y.: GCN-Denoiser: Mesh denoising with graph convolutional networks. ACM Transactions on Graphics **41**(1), 1–14 (2022). https://doi.org/10.1145/3480168 2, 3, 9, 10
25. Sun, X., Rosin, P.L., Martin, R., Langbein, F.: Fast and effective feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics **13**(5), 925–938 (2007). https://doi.org/10.1109/TVCG.2007.1065 2, 3, 4, 8, 12
26. Taubin, G.: Curve and surface smoothing without shrinkage. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) . pp. 852–857. IEEE (1995) 1, 3
27. Taubin, G.: Linear anisotropic mesh filtering. Research Report RC2213 (2001) 3, 8
28. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) . pp. 839–846. IEEE (1998) 7
29. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) . pp. 9446–9454 (2018). https://doi.org/10.1109/cvpr.2018.00984 2, 4, 5, 7
30. Wang, J., Huang, J., Wang, F.L., Wei, M., Xie, H., Qin, J.: Data-driven geometry-recovering mesh denoising. Computer-Aided Design **114**, 133–142 (2019). https://doi.org/10.1016/j.cad.2019.05.027 2, 3
31. Wang, P.S., Liu, Y., Tong, X.: Mesh denoising via cascaded normal regression. ACM Transactions on Graphics **35**(6), 1–12 (2016). https://doi.org/10.1145/2980179.2980232 2, 3, 10, 11, 12

32. Wang, R., Yang, Z., Liu, L., Deng, J., Chen, F.: Decoupling noise and features via weighted $\ell 1$-analysis compressed sensing. ACM Transactions on Graphics **33**(2), 1–12 (2014). https://doi.org/10.1145/2557449 1, 3
33. Wei, M., Huang, J., Xie, X., Liu, L., Wang, J., Qin, J.: Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery. IEEE Transactions on Visualization and Computer Graphics **25**(10), 2910–2926 (2019). https://doi.org/10.1109/TVCG.2018.2865363 3
34. Wei, M., Liang, L., Pang, W.M., Wang, J., Li, W., Wu, H.: Tensor voting guided mesh denoising. IEEE Transactions on Automation Science and Engineering **14**(2), 931–945 (2017). https://doi.org/10.1109/TASE.2016.2553449 1
35. Wei, M., Yu, J., Pang, W.M., Wang, J., Qin, J., Liu, L., Heng, P.A.: Bi-normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics **21**(1), 43–55 (2015). https://doi.org/10.1109/TVCG.2014.2326872 3
36. Wei, X., Chen, Z., Fu, Y., Cui, Z., Zhang, Y.: Deep hybrid self-prior for full 3D mesh generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021). https://doi.org/10.1109/iccv48922.2021.00575 4
37. Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., Panozzo, D.: Deep geometric prior for surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) . pp. 10122–10131 (2019). https://doi.org/10.1109/CVPR.2019.01037 4
38. Xu, J., Huang, Y., Cheng, M.M., Liu, L., Zhu, F., Xu, Z., Shao, L.: Noisy-as-Clean: Learning self-supervised denoising from corrupted image. IEEE Transactions on Image Processing **29**, 9316–9329 (2020). https://doi.org/10.1109/tip.2020.3026622 14
39. Yagou, H., Ohtake, Y., Belyaev, A.: Mesh smoothing via mean and median filtering applied to face normals. In: Proceedings of Geometric Modeling and Processing. Theory and Applications. GMP. (2002). https://doi.org/10.1109/GMAP.2002.1027503 1, 3
40. Zhang, W., Deng, B., Zhang, J., Bouaziz, S., Liu, L.: Guided mesh normal filtering. Computer Graphics Forum **34**, 23–34 (2015). https://doi.org/10.1111/cgf.12742 1, 2, 3, 9, 10
41. Zhang, Y., Shen, G., Wang, Q., Qian, Y., Wei, M., Qin, J.: GeoBi-GNN: Geometry-aware bi-domain mesh denoising via graph neural networks. Computer-Aided Design **144**, 103154 (2022). https://doi.org/https://doi.org/10.1016/j.cad.2021.103154 3, 4
42. Zhao, W., Liu, X., Wang, S., Fan, X., Zhao, D.: Graph-based feature-preserving mesh normal filtering. IEEE Transactions on Visualization and Computer Graphics **27**(3), 1937–1952 (2021). https://doi.org/10.1109/TVCG.2019.2944357 1, 3
43. Zhao, W., Liu, X., Zhao, Y., Fan, X., Zhao, D.: NormalNet: Learning-based mesh normal denoising via local partition normalization. IEEE Transactions on Circuits and Systems for Video Technology **31**(12), 4697–4710 (2021). https://doi.org/10.1109/TCSVT.2021.3099939 2, 3
44. Zhao, Y., Qin, H., Zeng, X., Xu, J., Dong, J.: Robust and effective mesh denoising using L0 sparse regularization. Computer-Aided Design **101**, 82–97 (2018). https://doi.org/10.1016/j.cad.2018.04.001 1, 3
45. Zheng, Y., Fu, H., Au, O.K.C., Tai, C.L.: Bilateral normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics **17**(10), 1521–1530 (2011). https://doi.org/10.1109/TVCG.2010.264 1, 2, 3, 7, 9