symmetry

MDPI

Article

Mesh Denoising Based on Recurrent Neural Networks

Yan Xing [1], Jieqing Tan [1,*], Peilin Hong [2,*], Yeyuan He [1] and Min Hu [3]

[1] School of Mathematics, Hefei University of Technology, Hefei 230009, China; xingyan@hfut.edu.cn (Y.X.); 2020111426@mail.hfut.edu.cn (Y.H.)
[2] School of Medical Information Engineering, Anhui University of Chinese Medicine, Hefei 230012, China
[3] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230602, China; jsjxhumin@hfut.edu.cn
* Correspondence: jieqingtan@hfut.edu.cn (J.T.); hongpeilin@ahtcm.edu.cn (P.H.)

Abstract: Mesh denoising is a classical task in mesh processing. Many state-of-the-art methods are still unable to quickly and robustly denoise multifarious noisy 3D meshes, especially in the case of high noise. Recently, neural network-based models have played a leading role in natural language, audio, image, video, and 3D model processing. Inspired by these works, we propose a data-driven mesh denoising method based on recurrent neural networks, which learns the relationship between the feature descriptors and the ground-truth normals. The recurrent neural network has a feedback loop before entering the output layer. By means of the self-feedback of neurons, the output of a recurrent neural network is related not only to the current input but also to the output of the previous moments. To deal with meshes with various geometric features, we use k-means to cluster the faces of the mesh according to geometric similarity and train neural networks for each category individually in the offline learning stage. Each network model, acting similar to a normal regression function, will map the geometric feature descriptor of each facet extracted from the mesh to the denoised facet normal. Then, the denoised normals are used to calculate the new feature descriptors, which become the input of the next similar regression model. In this system, three normal regression modules are cascaded to generate the last facet normals. Lastly, the model's vertex positions are updated according to the denoised normals. A large number of visual and numerical results have demonstrated that the proposed model outperforms the state-of-the-art methods in most cases.

Keywords: mesh denoising; recurrent neural networks; supervised learning; face clustering

Citation: Xing, Y.; Tan, J.; Hong, P.; He, Y.; Hu, M. Mesh Denoising Based on Recurrent Neural Networks. *Symmetry* **2022**, *14*, 1233. https://doi.org/10.3390/sym14061233

Academic Editors: Zhixun Su and Rushi Lan

Received: 2 April 2022
Accepted: 10 June 2022
Published: 14 June 2022

1. Introduction

Three-dimensional sensing and scanning technology has been widely used to capture the digital surface of physical objects. However, noise inevitably sneaks into the process of 3D capture and reconstruction, which significantly hinders the subsequent geometry processing tasks. Therefore, restoring high-quality 3D models from noise-damaged meshes has attracted widespread attention in computer graphics. Generally speaking, noise and sharp features are both high frequencies. Therefore, how to maintain sharp features when effectively removing noise is still a challenging problem in mesh denoising. In recent years, researchers have made significant contributions to mesh denoising.

Traditional denoising methods blur some features, more or less. For example, the early work from Taubin [1] and Desbrun et al. [2] focused on mesh fairing and smoothing, which led to feature blurring. With the rapid development of machine learning in all kinds of application areas, we aim at applying machine learning methods to mesh denoising to achieve better results. Based on Wang et al.'s work [3] and some other methods [4–6], we propose a data-driven mesh denoising method based on a recurrent neural network. The Recurrent Neural Network (RNN) is a neural network model that achieves a prominent performance on important tasks that include language modeling, speech recognition, and machine translation. In the traditional feedforward neural network, it is assumed that

neurons in the same layer are independent of each other. From the input layer to the hidden layer and then to the output layer, the layers are all connected, but the nodes in the same layer are not connected. This kind of common neural network cannot effectively solve the problems with time sequence or nonlinear data. RNN adds a delayer in the hidden layer as a one-step delay operator to achieve the purpose of memory. Therefore, the system has the ability to adapt to the time-varying characteristics and enhances the global stability of the network. It has stronger computational capability than the general feedforward neural network and can also be used to solve the problem of fast optimization. What makes RNN so special is the recurrent nets allow us to operate over sequences of vectors: sequences in the input, in the output, or both, generally. Two very nice papers from DeepMind [7,8] showed that the way of sequence processing still works well in the absence of sequence information. The takeaway is that even if your data is not in the form of sequences, you can still formulate and train a powerful RNN to process them sequentially. We train the RNN to map the feature descriptor to the denoised normal. Although the inputs seem unordered, the experiments show that the trained RNN can effectively denoise the mesh in real time.

2. Related Works

In the past two decades, various smoothing algorithms have been introduced to remove undesirable noise while maintaining sharp features in geometry. Classical Laplacian smoothing [9,10] is a very fast and simple surface smoothing method. However, when Laplacian smoothing is applied to a noisy 3D surface, it will lead to obvious shape distortions and surface shrinkage, in addition to noise removal. To overcome the shrinkage problem, Taubin [1] developed a surface smoothing algorithm, where the discrete Laplacian function of the surface was defined by the weighted averaging neighborhood normals. Desbrun et al. introduced an implicit smoothing algorithm [2], which produced stable results on irregular meshes and avoided volume shrinkage. Later, other isotropic methods were also proposed, including a low-pass/high-pass filter framework, volume-preserving method and differential characteristic-based methods [11–15]. The concept of differential coordinates was introduced by Alexa [11] as a geometric local shape descriptor.

However, early work focused on mesh smoothing and fairing algorithms, which usually resulted in feature blurring. In order to preserve the features, many researchers began to pay attention to anisotropic methods. The early anisotropic methods [16–20] were based on anisotropic geometric diffusion, inspired by image processing [21].

Bilateral filtering denoising techniques originate from image denoising [22]. Fleishman et al. successfully proposed a single-step iterative bilateral mesh denoising method directly applied to vertices [23], which is closely related to the anisotropic filter. In [23], a vertex and its normal define a parameterization plane, over which a bilateral filter is applied to the neighborhood of that vertex, and the vertex will be updated along its normal vector according to the displacement obtained from the filter. However, experiments show that this method does not always accurately maintain the fine features of the mesh. Since mesh normals can represent the surface geometry more directly than vertex positions, lately, many researchers, such as Sun et al. [24], Zheng et al. [4], Wei et al. [25], Zhang et al. [5], and Yadav et al. [15], have used a two-stage framework, first filtering the mesh normals and then updating the vertex positions according to the estimated normals. More recently, in a two-stage mesh denoising fashion, Yadav et al. [15] used Tukey's bi-weight function as a similarity function in the bilateral normal filtering, which stops the diffusion at sharp edges to retain the features, and introduced an edge-weighted Laplace operator to compute a differential coordinate to produce a high-quality mesh. Bilateral filtering is also the core of many multi-step normal filtering methods [4,5]. Compared with the single-stage method, the two-stage method can usually recover the features more effectively when dealing with high-level noises.

Based on the observation that sharp features may be sparse on the 3D model, several mesh denoising methods based on sparse optimization were proposed. For example, He et al. [6] use the L0 norm, which directly measures sparsity, to preserve sharp features

when smoothing the surface. Wang et al. [26] and Wu et al. [27] performed L1 optimization to restore sharp features. Although these global methods are more robust numerically, often, the meshes with local fine details are excessively smoothed. Zhang et al. [28] combined the total variation and piecewise constant function to perform variational mesh denoising. Correct estimations of differential geometric properties are usually essential for these methods.

Lately, the methods based on deep neural networks have made important progress in various fields and achieved impressive results. Li et al. [29] presented a deep normal filtering network, called DNF-Net, for mesh denoising. To capture the local geometry, DNF-Net takes patches of facet normals as inputs and directly outputs the corresponding denoised facet normals of the patches. The network includes a multi-scale feature embedding unit, a residual learning unit, and a joint supervision loss function. However, DNF-Net may produce unsatisfying denoising results when encountering a test mesh whose noise pattern is very different from that in the training set. Shen et al. [30] presented a GCN-Denoiser to perform graph convolutions in the dual spaces of triangular meshes, which utilizes both static and dynamic edge convolutions to learn both the explicit mesh structure and implicit potential relations among non-adjacent neighbors. To obtain a better denoising effect, they also employed multiple GCNs to progressively regress the facet normals in a cascade manner.

As the pioneers of neural network methods for mesh denoising, Wang et al. [3] proposed a cascaded radial basis function (RBF) neural network to denoise 3D meshes. It is a data-driven method, in which a large number of noisy meshes and ground truth meshes are used for learning regression function, and a regression model is established in order to remove mesh noise. However, this regression model [3] adopted a single hidden layer radial basis neural network to carry out the global fitting of the data. The radial basis neural network is a typical feedforward neural network. A feedforward neural network is the simplest neural network, in which neurons are all arranged hierarchically and are only adjacent to the neurons in the previous layer. The neuron in the current layer receives the output of the previous layer as input and transmits its output as the input of the next layer. There is no feedback between the layers.

Due to the lack of feedback between layers, the denoising effects of the RBF neural network are not particularly ideal. In this paper, the recurrent neural network is introduced to denoise 3D meshes. The input/output of our model seems to have no sequential information, but it is important to realize that the positional relationships in the spatial domain can still be processed in a sequential manner using RNN with powerful temporal capabilities. The steps of our method are as follows. Firstly, descriptors representing geometric features are extracted from the mesh. Then, we use the recurrent neural network to train the regression function, which maps the descriptor to the expected face normal. The output of the hidden layer not only relates to the current input but, also, to its output at the previous moments by using neurons with self-feedback. Then, the denoised normals will be used to calculate the new feature descriptors as the next input of the regression model to perform cascaded regression. Finally, the normals obtained from the trained regression model are used to update the vertex coordinates to reconstruct a new mesh.

3. Methods

3.1. RNN

RNN, born in the 1980s, is a typical dynamic recurrent neural network, which is based on the basic structure of the BP network and adds a delayer to the hidden layer as a one-step delay operator. The original intention of RNN is to process sequential data. RNN provides a very elegant way of dealing with (time) sequential data, which makes use of correlations between data points that are close in the sequence. The current output is not only related to the current input but also related to the previous output. Therefore, the system can adapt to time-varying characteristics and enhances the global stability of the network. It has stronger computing power than feedforward neural networks and can also be used to

solve the problem of fast optimization. Figure 1 shows a basic RNN architecture with a delay line.

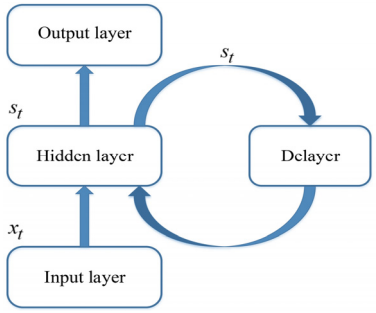

Figure 1. The structure of RNN.

The structure of RNN can be generally divided into four layers: input layer, hidden layer, delayer, and output layer. The connection of the input layer, hidden layer, and output layer is similar to a feedforward network. Usually, the activation function of the hidden layer takes a nonlinear hyperbolic tangent function (tanh). The output of the hidden layer unit is sent to the delayer for storage and then fed into the hidden layer at the next moment, forming the next input of the hidden layer together with the input signal. In this way, the delayer acts as a delay operator with one step delay function. By the storage and delay of the delayer, the network is time-sensitive. This internal feedback mechanism enhances the ability of the network to process dynamic information.

Figure 2 shows an unfolded diagram of the hidden layer in two time steps. $t-1$ and t represent the time steps, x is the input samples, s_t is the memory of the samples at time step t, U and W represent the input weight and the memory weight, respectively, and V is the output weight. At time step $t = 0$, s will be initialized as $s_0 = 0$, and U, W, and V will be initialized randomly. State s_t, the storage state of the time step t, participates in the next prediction. This rule and the final output are formulated as follows:

$$s_t = f(Ux_t + Ws_{t-1}), \tag{1}$$

$$o_T = g(Vs_T), \tag{2}$$

where f and g are the activation functions of the hidden layer and output layer, respectively. f is usually a sigmoid function. g is usually a linear function in regression case and a SoftMax function in a classification case.

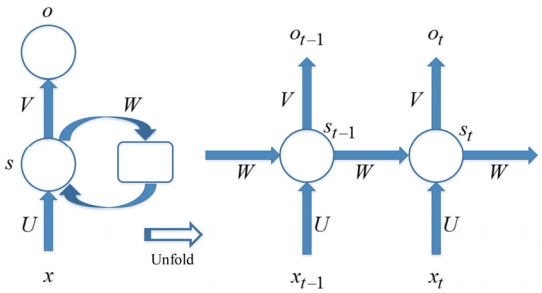

Figure 2. An unfolded diagram of hidden layer in two time steps.

3.2. Geometric Feature Descriptors

3.2.1. Bilateral Filtering and Guided Bilateral Filtering

We use $M = (V, F)$ to represent a mesh, where $V = \{v_i : i = 1, \ldots, n\}$ represents the set of vertices, and $F = \{f_i : i = 1, \ldots, m\}$ represents the set of faces. Denote the centroid, normal, area, and neighborhood of f_i by c_i, n_i, A_i, and N_{f_i}. The well-known bilateral normal filter of 3D mesh is defined by

$$n_i{}' = norm(\sum_{f_j \in N_{f_i}} A_j \exp(\frac{-\|c_i - c_j\|^2}{2\sigma_s^2}) \exp(\frac{-\|n_i - n_j\|^2}{2\sigma_r^2}) n_j), \tag{3}$$

where σ_s and σ_r are the standard deviation of spatial Gaussian filter and signal Gaussian filter, respectively. To deal with a large noise, the guided normal filter is introduced via a more accurately guided signal. Similarly, the guided normal joint filter is defined by

$$n_{g,i}{}' = norm(\sum_{f_j \in N_{f_i}} A_j \exp(\frac{-\|c_i - c_j\|^2}{2\sigma_s^2}) \exp(\frac{-\|g(n_i) - g(n_j)\|^2}{2\sigma_r^2}) n_j). \tag{4}$$

In the formula, $g(n_i)$ and $g(n_j)$ are the guided normal vectors of the face f_i and f_j, respectively, and the guided normal vector is calculated by

$$g(n_i) = norm(\sum_{f_j \in N_{f_i}} A_j \exp(\frac{-\|c_i - c_j\|^2}{2\sigma_s^2}) n_j). \tag{5}$$

3.2.2. Feature Descriptors

The bilateral normal feature descriptor D_i and guided bilateral normal feature descriptor $D_{g,i}$ for each face are defined by binding different parameters to the bilateral normal filter and guided bilateral normal filter.

$$D_i = (n_i{}'(\sigma_{s_1}, \sigma_{r_1}), n_i{}'(\sigma_{s_2}, \sigma_{r_2}), \ldots, n_i{}'(\sigma_{s_L}, \sigma_{r_L})), \tag{6}$$

$$D_{g,i} = (n_{g,i}{}'(\sigma_{s_1}, \sigma_{r_1}), n_{g,i}{}'(\sigma_{s_2}, \sigma_{r_2}), \ldots, n_{g,i}{}'(\sigma_{s_L}, \sigma_{r_L})). \tag{7}$$

$D_{g,i}$ is used in the first cascaded regression and D_i in the remaining cascaded regression.

3.3. Regression Model

The dataset P is partitioned into the training set P_X, validation set P_V, and test set P_T. P_X consists of a series of training data pairs $\{D_i, \overline{n_i}\}_{i=1}^{N_{P_X}}$. $\overline{n_i}$ is the ground truth normal of face f_i. First, divide the training data into four clusters based on the feature descriptors via the k-means method. The faces in each cluster have strong geometric similarities. For cluster l, we divide the data into the training set $P_X(l)$, validation set $P_V(l)$, and test set $P_T(l)$. A regression function is trained by minimizing the following energy function:

$$E = \sum_{\{D_i, \overline{n_i} \in P_X(l)\}} \|norm(R_l(D_i)) - \overline{n_i}\|^2 + \mu Ereg, \tag{8}$$

where E_{reg} is an L2 regularization term to prevent overfitting, μ is set as 0.05 in our experiments, and the regression function $R_l(D)$ for cluster l is defined by

$$R_l(D) = g(s_{T,l} V) = \sum_{k=1}^{N_r} s_{T,l}(k) V_{l,k} \tag{9}$$

where N_r is the number of the hidden nodes, and

$$s_{t,l}(k) = \tanh(U_{l,k}^T D_t + W_{l,k}^T s_{t-1,l}(k) + W_{l,k}^T s_{t-2,l}(k) + b_{l,k}) = \frac{2}{1 + \exp\left\{-2 * [U_{l,k}^T D_t + W_{l,k}^T s_{t-1,l}(k) + W_{l,k}^T s_{t-2,l}(k) + b_{l,k}]\right\}} - 1. \tag{10}$$

where $U_{l,k} \in R^{3L}, W_{l,k} \in R^{N_r}, b_{l,k} \in R, V_{l,k} \in R^3$. In our experiment, N_r is set to 10. The regression model is iteratively trained until the approximation error on the validation set increases. Since the k-means clustering is carried out according to the similarity of

descriptors, and the models are trained for each cluster separately, the final regression function is

$$R(D) = R_l(D),\ if\ \forall k(\|D - c_l\| \leq \|D - c_k\|), \tag{11}$$

where c_l refers to the center of cluster l. The normal of the final output should be

$$n_i^* = norm(R(D)). \tag{12}$$

4. Vertex Update

After the estimation of the face normals, the vertex positions need to be updated to match the new normals $\{n_i^*\}$. We adopt the iterative scheme from [23] for the vertex update. Specifically, the iterative approach is

$$v_i^{t+1} = v_i^t + \frac{1}{3|F_{v_i}|} \sum_{f_k \in F_{v_i}} \sum_{(i,j) \in \partial f_k} n_k^* (n_k^* \cdot (v_j^t - v_i^t)), \tag{13}$$

where F_{v_i} is the set of faces that share a common vertex v_i, and ∂f_k is the set of edges of face f_k.

5. Experiments and Analysis

5.1. Visual Comparisons

In this paper, we have proposed a data-driven feature-preserving denoising method for mesh surfaces. The core of our approach lies in using the special structure of recurrent neural network, which has more feedback layers than the feedforward neural network. Our method was tested on many meshes corrupted with either synthetic or raw scanning noise to validate the efficiency and robustness of the proposed method. We compare the results with seven state-of-the-art denoising methods, namely, He et al.'s L0 minimization [6], Zheng et al.'s local bilateral normal filter [4], Zhang et al.'s guided normal filter [5], Wang et al.'s cascaded normal regression method [3], Yadav et al.'s robust and high-fidelity mesh denoising [15], Li et al.'s DNF-Net [29], and Shen et al.'s GCN-Denoiser [30]. Different mesh denoising methods have different sets of parameters. To allow for fair comparisons, we chose the parameters suggested by the authors of [3,29,30] or fine-tuned the parameters to produce visually better results.

First, these approaches are tested on the Nicolo model with low-level noise. Obviously, all the methods can remove this low-level noise (see Figure 3). However, [6] oversmooths and slightly shrinks this mesh, and the results of [4,5] are not as good as ours in maintaining some fine details. Our method is very close to [3,15,29,30] in visual effect but has a smaller mean angular error (See Table 1). Figure 4 shows a Bunny-Hi model with rich textures corrupted by medium low-level noise. One can see that the results by [4–6] lost the textures but [3,15,29] and our method maintains them. The GCN [30] seems to retain the most textures. Later, we will give numerical comparisons to distinguish the superior method, especially when visual results are similar. For the Trim-star model with medium high-level noise in Figure 5, Refs. [3–5,29] removed most of the noise, but some artifacts remained. Ref. [6] oversmooths the model and sharpens the inner five corners. Ref. [30] and our method have the best denoising results.

In addition to synthetic cases, the regression model based on RNN was verified on the other two datasets of meshes with raw noise. The results generated by the eight methods shared visual similarities when handling the low-level noise. However, in the case of high-level noise, the existing methods could not remove the noise completely. The Boy model with high-level noise in Figure 6 and the Pyramid model with moderate-level noise in Figure 7 were acquired by a Microsoft Kinect V1 camera and Microsoft Kinect V2 camera, respectively. These scanning data from Kinect V1, V2, [3,15,30], and our method can remove noise to the maximum extent, meanwhile protecting the features well.

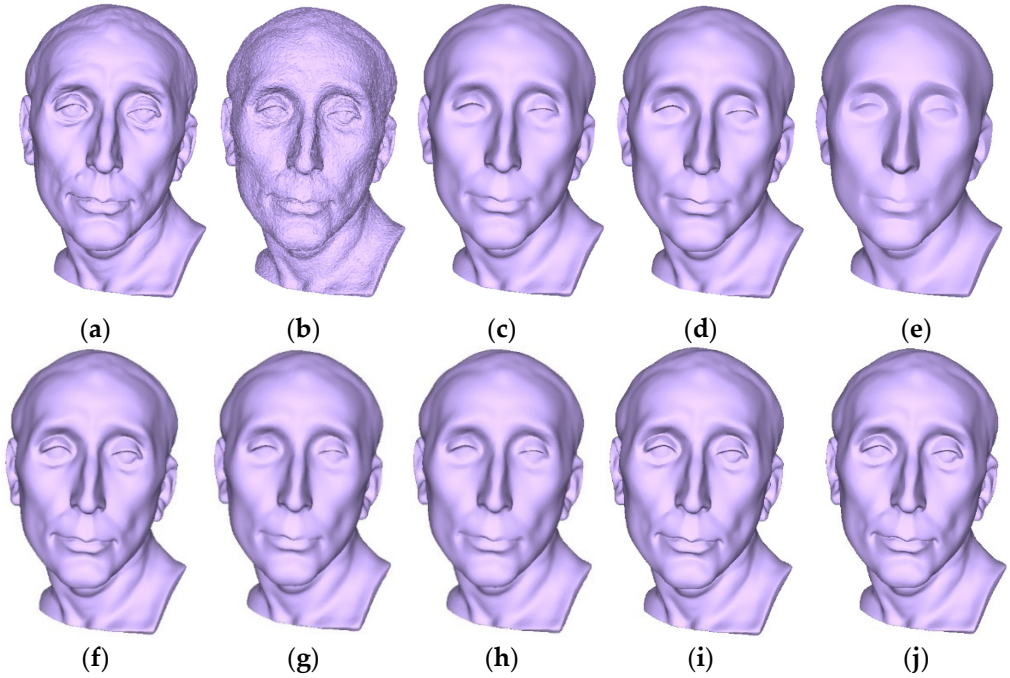

Figure 3. Denoising of the Nicolo model corrupted by Gaussian noise (0.1) in a normal direction. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

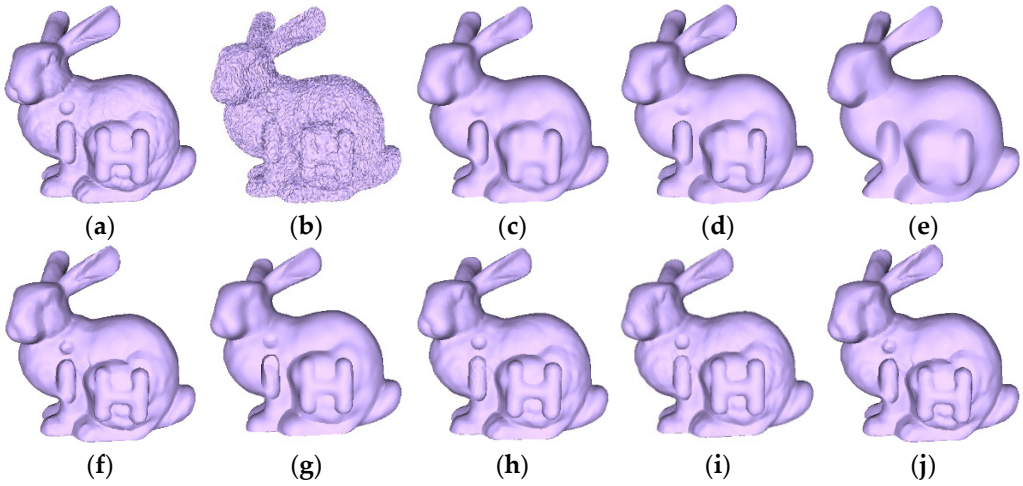

Figure 4. Denoising of the Bunny-hi model corrupted by Gaussian noise (0.3) in a normal direction. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

For meshes with an extremely high level of noise, denoising is an arduous process for traditional methods. However, our method shows its strong power to remove large noise while maintaining appearances. Figure 8 shows the Fandisk model with sharp features, which is contaminated by Gaussian noise of 0.9 times the average edge length. Refs. [4,6] failed in the denoising task. Refs. [3,5,29] could not obtain satisfactory denoising results, but [15,30] and our method obtained acceptable denoising results.

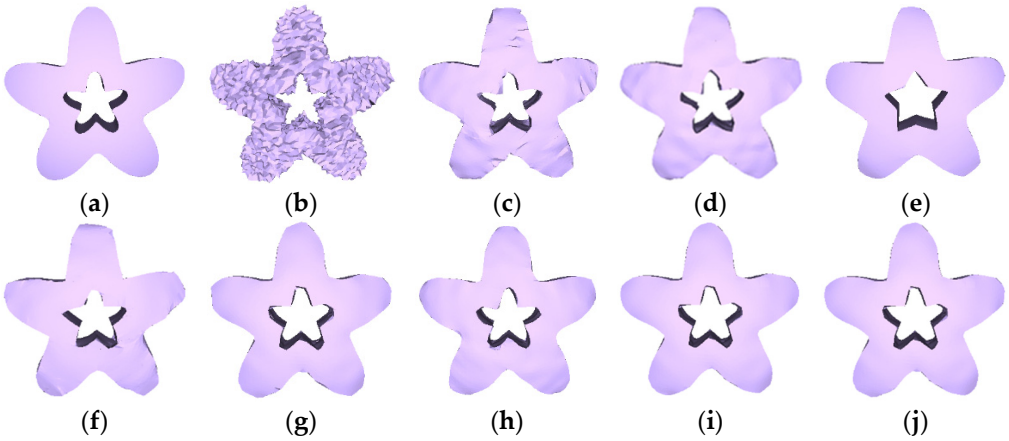

Figure 5. Denoising of the Trim-star model corrupted by Gaussian noise (0.5) in a normal direction. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

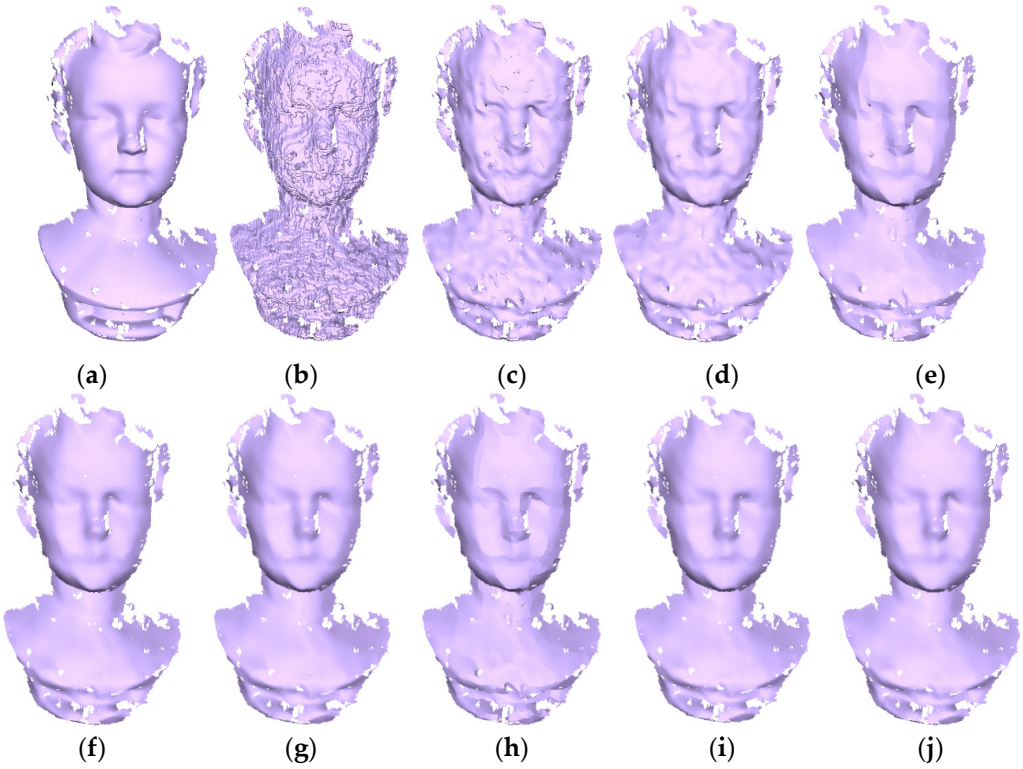

Figure 6. Denoising of the Boy model corrupted by a Kinect V1 camera. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

Table 1. The comparison of the MAE.

Models	BNF	GNF	L0	CNR	RFD	DNF	GCN	OURS
Nicolo (Gaussian 0.1)	5.378	5.755	7.990	4.099	7.704	5.453	4.377	**4.077**
Fandisk (Gaussian 0.2)	2.032	2.195	5.143	1.780	2.029	4.257	**1.639**	1.740

Table 1. *Cont.*

Models	BNF	GNF	L0	CNR	RFD	DNF	GCN	OURS
Gargoyle (Gaussian 0.2)	11.210	12.480	16.230	7.540	9.325	7.979	**7.254**	7.500
Trim-star (Gaussian 0.2)	6.515	6.924	8.123	4.142	9.657	5.477	4.428	**4.057**
Rockerarm (Gaussian 0.3)	7.639	7.735	12.130	5.382	9.172	6.244	**5.073**	5.318
Bunny-Hi (Gaussian 0.3)	7.341	7.162	11.030	5.775	6.677	6.522	**5.469**	5.699
Boy (Kinect V1)	14.660	11.801	10.510	9.798	9.873	10.079	**9.576**	9.754
Pyramid (Kinect V2)	9.042	7.963	7.622	6.897	6.897	7.419	6.797	**6.789**

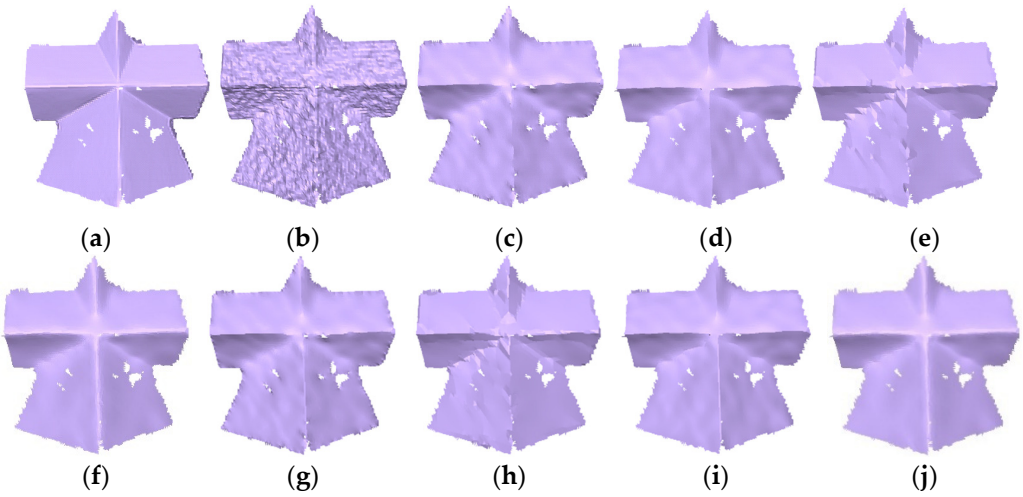

Figure 7. Denoising of the Pyramid model corrupted by a Kinect V2 camera. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

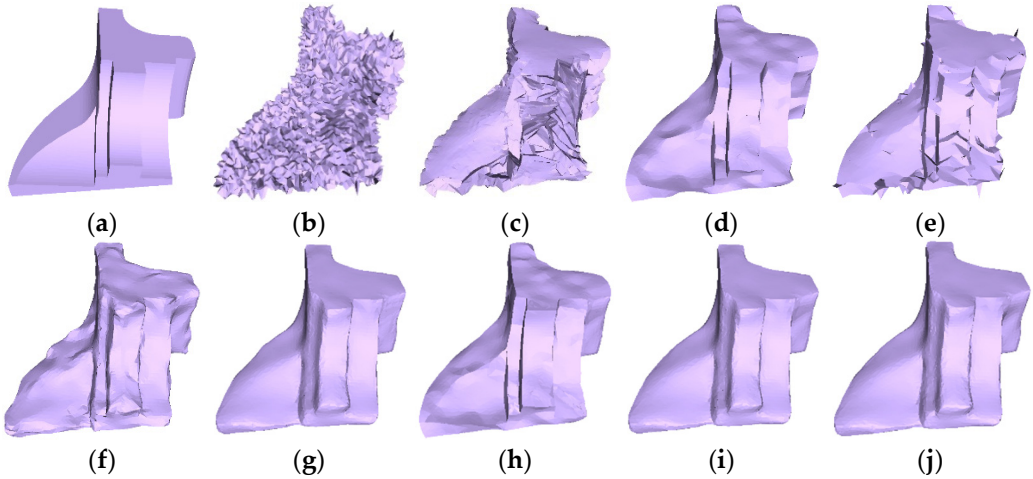

Figure 8. Denoising of the Fandisk model corrupted by Gaussian noise (0.9) in a normal direction. (**a**) Original model. (**b**) Noisy model. (**c**) BNF [4]. (**d**) GNF [5]. (**e**) L0 [6]. (**f**) CNR [3]. (**g**) RFD [15]. (**h**) DNF [29]. (**i**) GCN [30]. (**j**) Ours.

5.2. Quantitative Comparisons

We also make quantitative comparisons to supplement qualitative comparisons. Sometimes, it is easy to judge the performance of different methods through visual comparisons. However, when there exist only small visual differences, one needs numerical metrics to distinguish them. We choose the mean angular error and Hausdorff distance as the quantitative metrics suggested by the previous works [3,24]. The mean angular error (MAE) is to estimate the mean angular difference between the face normals of the denoising mesh and those of the ground truth mesh and is defined as

$$MAE = E[\angle(n_d, n)], \tag{14}$$

where $\angle(n_d, n)$ is the angle between the denoising normal and the ground truth, and E is the expectation operator, which is realized by averaging the angles between the predicted normals and the ground truth normals of all facets in the mesh. The small mean angular error indicates that the shape of the denoising surface is close to that of the ground truth surface. We calculated the MAEs on different types of meshes, which have corresponding ground truth normals. The MAE values of experimental results are listed in Table 1, and the best results are shown in bold. The results of GCN [30] and our method are closest to the ground truth models, since the smallest MAE values are either obtained by [30] or by our method. The statistics of the MAE, together with the above visual comparisons, show that our approach is convincing both numerically and visually.

Hausdorff distance is used to measure the positional error between the denoised mesh and the ground truth mesh and is defined as

$$D_H = \max[D_h(M_d, M_g), D_h(M_g, M_d)], \tag{15}$$

where

$$D_h(M_1, M_2) = \max_{v_1 \in M_1} \min_{v_2 \in M_2} \|v_1 - v_2\|. \tag{16}$$

where M_d and M_g are the denoised mesh and the ground truth mesh, respectively, and v_1 and v_2 are the vertices in the mesh. The Hausdorff distances for the experimental results are listed in Table 2, and the smallest results are shown in bold. Our method generates the denoising result with the smallest Hausdorff distance on meshes Gargoyle, Trim-star, and boy (Kinect V1). GCN [30] has the smallest Hausdorff distance on models Nicolo, Rockerarm, and Bunny-Hi. The other two minimum Hausdorff distances appear in the CNR [3] and BNF [4] methods, respectively. Except for Pyramid captured by the Kinect V2 camera, the Hausdorff distances of the results generated by our method were the minimum or close to the minimum. This indicates that our method can preserve the volume of the 3D model and avoid volume shrinkage. The Hausdorff distance is usually, but not always, consistent with the visual comparisons. This may be because the repositioning of vertices after denoising will result in larger or smaller Hausdorff distances.

Table 2. The comparisons of Hausdorff distances.

Models	BNF	GNF	L0	CNR	RFD	DNF	GCN	OURS
Nicolo (Gaussian 0.1)	0.0208	0.0282	0.0435	0.0151	0.0260	0.0210	**0.0138**	0.0151
Fandisk (Gaussian 0.2)	0.8707	0.9979	0.9997	**0.7512**	0.7616	0.7574	0.7557	0.7601
Gargoyle (Gaussian 0.2)	2.6390	0.7996	1.4213	0.5928	0.6045	0.6115	0.5989	**0.5840**
Trim-star (Gaussian 0.2)	0.5004	0.4498	0.5076	0.2257	0.2411	0.2741	0.2299	**0.2079**
Rockerarm (Gaussian 0.3)	0.0104	0.0190	0.0171	0.0075	0.0218	0.0106	**0.0060**	0.0074

Table 2. *Cont.*

Models	BNF	GNF	L0	CNR	RFD	DNF	GCN	OURS
Bunny-Hi (Gaussian 0.3)	0.0095	0.0102	0.0187	0.0074	0.0088	0.0089	**0.0059**	0.0069
Boy (Kinect V1)	6.1152	6.1374	5.8713	5.9764	5.9965	6.0018	6.0104	**5.8206**
Pyramid (Kinect V2)	**6.5917**	6.8797	7.6550	7.0556	7.0723	6.7052	7.0018	7.0093

To compare the effects of these methods more intuitively, Figures 9 and 10 plot the bar charts of the MAE and Hausdorff distances, respectively. For the convenience of observation, for the three models Nicolo, Rockerarm, and Bunny-Hi in Table 2, the values of Hausdorff distances are multiplied by 100.

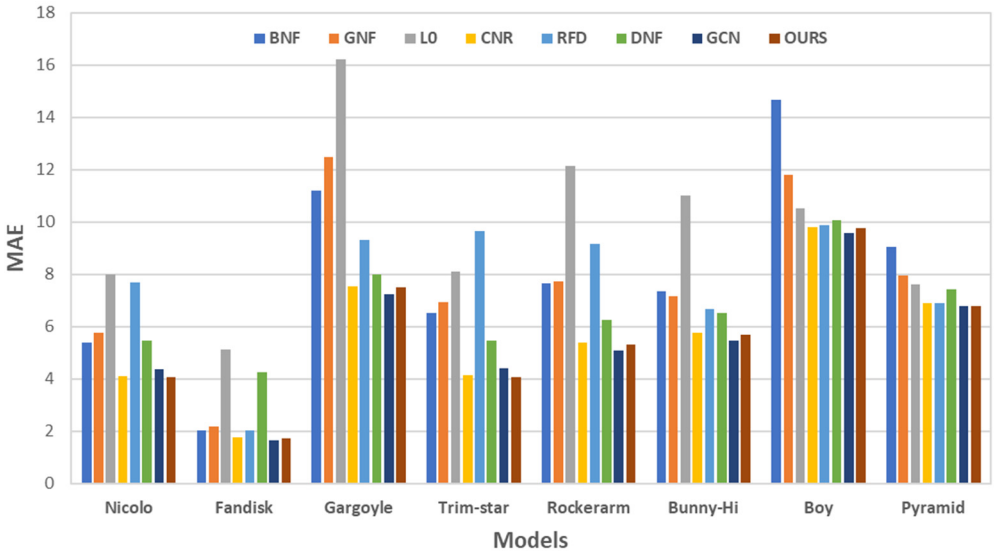

Figure 9. Comparisons of the MAE.

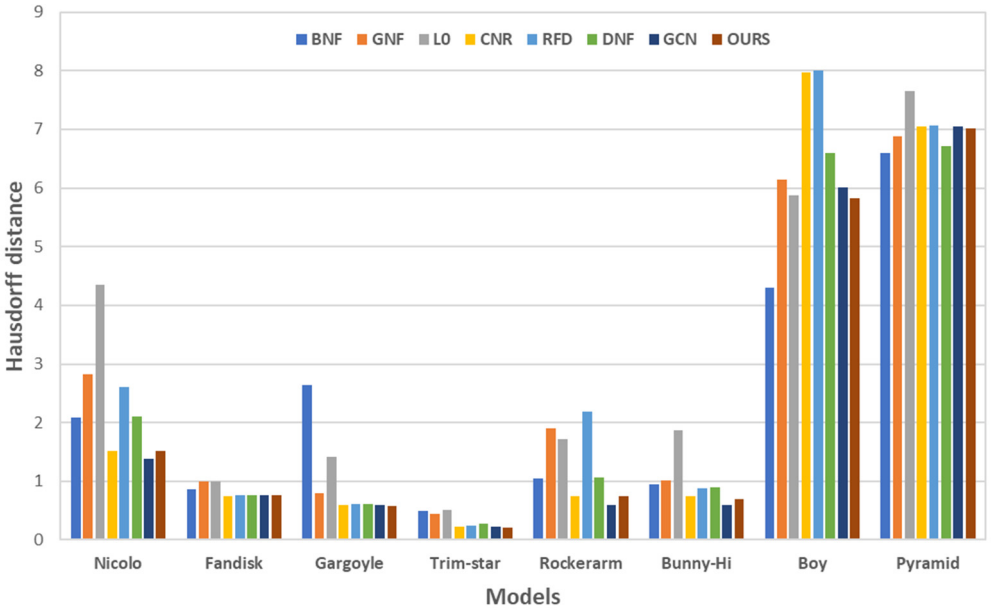

Figure 10. Comparisons of the Hausdorff distances.

5.3. Running Time

We recorded the time cost of each method in the experiments. The denoising time is listed in Table 3. The minimum running time is highlighted in bold. Our method is the fastest of all. Whether the number of facets in a mesh is large or small, the time performance of our method is very good. It can be seen from Table 3 that the networks trained by CNR [3] and our method can be directly used for real-time denoising. Although the method of GCN [30] has achieved better results than our method in many cases, the running time of our method is much faster than that of GCN. As for why their speed is slower than CNR, the authors of [30], we believe that it is because they used complex neural networks. In conclusion, our approach has a good balance between efficacy (effectiveness) and time efficiency.

Table 3. Computational times (s).

Models	#Vertices	#Faces	#Edges	BNF	GNF	L0	CNR	RFD	DNF	GCN	Ours
Nicolo	50,419	99,994	150,412	15.56	275.24	1873.95	3.47	18.43	1280.87	40.33	**3.36**
Fandisk	6475	12,946	19,419	1.88	33.87	131.94	0.62	4.07	132.23	5.12	**0.60**
Gargoyle	85,558	171,112	256,668	28.41	584.90	1948.63	7.74	19.05	2605.41	112.45	**7.17**
Trim-star	5192	10,384	15,576	1.43	53.96	92.38	0.56	3.12	102.53	4.56	**0.54**
Rockerarm	9413	18,826	28,239	2.90	69.39	232.91	0.86	4.90	222.42	7.15	**0.84**
Bunny-Hi	34,877	69,537	104,417	11.04	178.40	1241.35	2.72	13.22	989.36	27.13	**2.65**
Boy	28,187	53,229	81,541	22.34	176.21	458.98	1.78	6.77	621.05	16.30	**1.75**
Pyramid	6915	13,296	20,229	2.37	31.85	97.25	0.61	3.58	132.18	5.04	**0.57**

6. Limitations

Although our method does not need to tune the parameters in denoising, the sizes of training datasets and the intensities and types of noise involved in the datasets will affect the generalization of the final regression model. When encountering large noise, the geometric feature descriptors may be unable to tell noise from features. Searching for better alternate feature descriptors will be one of our future efforts.

7. Conclusions and Future Works

In this paper, a novel RNN-based mesh denoising model is proposed, inspired by cascade normal regression [3]. The feedback property of RNN is utilized, which the RBF neural network does not have. The previous output of the hidden layer can be used as the memory and fed back to the next input, which connects the previous learning results with the current learning and reinforces the training effect. Although our task seems to have nothing to do with sequence processing, it is still possible to use this powerful formalism of RNN to process our data in a sequential manner. Experimental results show that the proposed method is superior to or comparable to the state-of-the-art methods in removing noise and maintaining the features and details of meshes for both synthetic noise and scanning noise. In the future, we can try to design better feature descriptors to characterize the mesh and distinguish noise and features accurately. In addition, we can also try to construct an end-to-end deep learning network to automatically learn mesh features, so that we can obtain better denoising results without calculating feature descriptors in advance.

Author Contributions: Conceptualization, J.T.; methodology, Y.X.; software, Y.H. and Y.X.; validation, P.H.; formal analysis, M.H.; investigation, P.H.; data curation, Y.H.; writing—original draft preparation, Y.X. and Y.H.; writing—review and editing, Y.X. and J.T.; supervision, M.H.; project administration, J.T.; and funding acquisition, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 62172135 and No. 62176084 and in part by the Fundamental Research Funds for the Central Universities of China under Grant No. PA2021GDSK0094.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The meshes used in the synthetic dataset are courtesy of the Aim@Shape repository and the 3D mesh database of the Inria GAMMA group. We would like to thank Peng-Shuai Wang for providing the data and codes for the mesh denoising. We would like to thank Sunil Kumar Yadav, Xianzhi Li, and Yuefan Shen for sharing the implementation of their methods in [15,29,30], respectively. We also thank the editors and the anonymous reviewers for their hard work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taubin, G. A signal processing approach to fair surface design. In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 6–11 August 1995; pp. 351–358.
2. Desbrun, M.; Meyer, M.; Schroder, P.; Barr, A. Implicit fairing of irregular meshes using diffusion and curvature flow. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 26–30 July 1999; Volume 33, pp. 317–324.
3. Wang, P.S.; Liu, Y.; Tong, X. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.* **2016**, *35*, 232. [CrossRef]
4. Zheng, Y.; Fu, H.; Au, O.K.-C.; Tai, C.-L. Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 1521–1530. [CrossRef] [PubMed]
5. Zhang, W.; Deng, B.; Zhang, J.; Bouaziz, S.; Liu, L. Guided mesh normal filtering. *Comput. Graph. Forum.* **2015**, *34*, 23–34. [CrossRef]
6. He, L.; Schaefer, S. Mesh denoising via l0 minimization. *ACM Trans. Graph. (TOG)* **2013**, *32*, 1–8.
7. Ba, J.; Mnih, V.; Kavukcuoglu, K. Multiple object recognition with visual attention. *arXiv* **2014**, arXiv:1412.7755.
8. Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.J.; Wierstra, D. Draw: A recurrent neural network for image generation. *Comput. Sci.* **2015**, *37*, 1462–1471.
9. Field, D.A. Laplacian smoothing and delaunay triangulations. *Int. J. Numer. Methods Biomed. Eng.* **1988**, *4*, 709–712. [CrossRef]
10. Vollmer, J.; Mencl, R.; Müller, H. Improved Laplacian smoothing of noisy surface meshes. *Comput. Graph. Forum.* **1999**, *18*, 131–138. [CrossRef]
11. Alexa, M. Differential coordinates for local mesh morphing and deformation. *Vis. Comput.* **2003**, *19*, 105–114. [CrossRef]
12. Nehab, D.; Rusinkiewicz, S.; Davis, J.; Ramamoorthi, R. Efficiently combining positions and normal for precise 3D geometry. *ACM Trans. Graph. (TOG)* **2005**, *24*, 536–543. [CrossRef]
13. Nealen, A.; Igarashi, T.; Sorkine, O.; Alexa, M. Laplacian mesh optimization. In Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, Kuala Lumpur, Malaysia, 29 November–2 December 2006; pp. 381–389.
14. Su, Z.X.; Wang, H.; Cao, J.J. Mesh denoising based on differential coordinates. In Proceedings of the 2009 IEEE International Conference on Shape Modeling and Applications, Beijing, China, 26–28 June 2009; pp. 1–6.
15. Yadav, S.K.; Reitebuch, U.; Polthier, K. Robust and high fidelity mesh denoising. *IEEE Trans. Vis. Comput. Graph.* **2019**, *25*, 2304–2310. [CrossRef] [PubMed]
16. Bajaj, C.L.; Xu, G. Anisotropic diffusion on surfaces and functions on surfaces. *ACM Trans. Graph. (TOG)* **2003**, *22*, 4–32. [CrossRef]
17. Clarenz, U.; Diewald, U.; Rumpf, M. *Anisotropic Geometric Diffusion in Surface Processing*; IEEE: Piscataway, NY, USA, 2000; pp. 397–405.
18. Ouafdi, A.F.E.; Ziou, D. A global physical method for manifold smoothing. In Proceedings of the 2008 IEEE International Conference on Shape Modeling and Applications, Stony Brook, NY, USA, 4–6 June 2008; pp. 11–17.
19. Desbrun, M.; Meyer, M.; Schroder, P.; Barr, A.H. Anisotropic feature-preserving denoising of height fields and bivariate data. In Proceedings of the Graphics Interface 2000, Montréal, QC, Canada, 15–17 May 2000; pp. 145–152.
20. Tsuchie, S.; Higashi, M. Surface mesh denoising with normal tensor framework. *Graph. Models* **2012**, *74*, 130–139. [CrossRef]
21. Perona, P.; Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 629–639. [CrossRef]
22. Tomasi, C.; Manduchi, R. Bilateral filtering for gray and color images. In Proceedings of the 6th International Conference on Computer Vision, Bombay, India, 4–7 January 1998; pp. 839–846.
23. Fleishman, S.; Drori, I.; Cohen-Or, D. Bilateral mesh denoising. In *ACM SIGGRAPH 2003 Papers*; Association for Computing Machinery: New York, NY, USA, 2003; pp. 950–953.
24. Sun, X.; Rosin, P.; Martin, R.; Langbein, F. Fast and effective feature preserving mesh denoising. *IEEE Trans. Vis. Comput. Graph.* **2007**, *13*, 925–938. [CrossRef]
25. Wei, M.; Yu, J.; Pang, W.; Wang, J.; Qin, J.; Liu, L.; Heng, P. Bi-normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graph.* **2015**, *21*, 43–55. [CrossRef]
26. Wang, R.; Yang, Z.; Liu, L.; Deng, J.; Chen, F. Decoupling noise and features via weighted l1-analysis compressed sensing. *ACM Trans. Graph. (TOG)* **2014**, *33*, 1–12.

27. Wu, X.; Zheng, J.; Cai, Y.; Fu, C. Mesh denoising using extended rof model with l1 fidelity. *Comput. Graph. Forum.* **2015**, *34*, 35–45. [CrossRef]
28. Zhang, H.; Wu, C.; Zhang, J.; Deng, J. Variational mesh denoising using total variation and piecewise constant function space. *IEEE Trans. Vis. Comput. Graph.* **2015**, *21*, 873–886. [CrossRef]
29. Li, X.; Li, R.; Zhu, L.; Fu, C.-W.; Heng, P.-A. DNF-Net: A deep normal filtering network for mesh denoising. *IEEE Trans. Vis. Comput. Graph.* **2020**, *27*, 4060–4072. [CrossRef]
30. 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 Trans. Graph. (TOG)* **2022**, *41*, 1–14. [CrossRef]