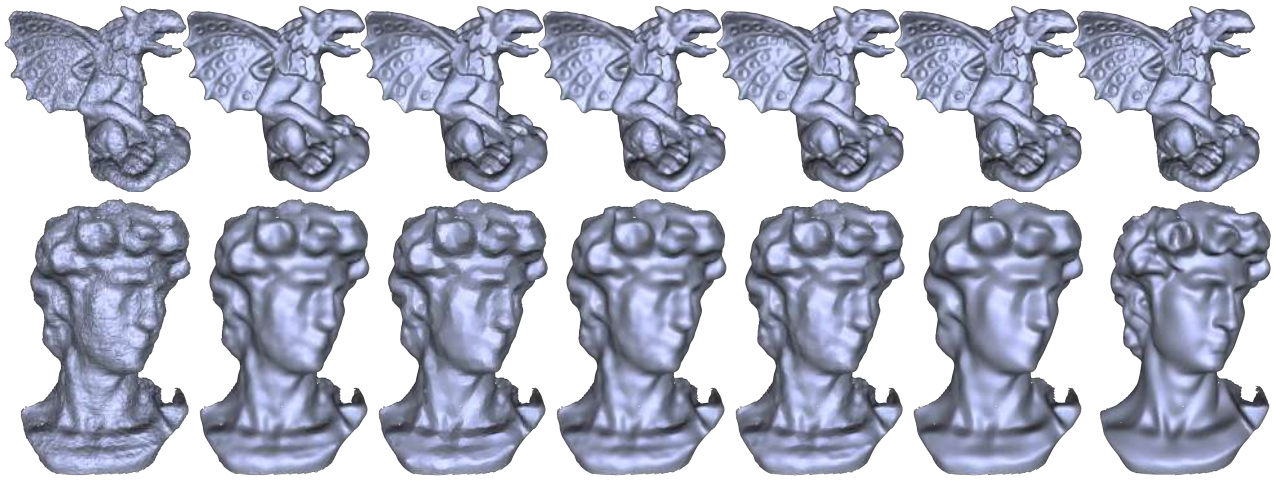
Mesh Denoising via Cascaded Normal Regression

Peng-Shuai Wang[*] [†]
[*] Tsinghua University

Yang Liu[†]

Xin Tong[†]
[†] Microsoft Research Asia



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth

Figure 1: *Mesh denoising. On a noisy input mesh (a) with its ground-truth mesh (g), we compare our cascaded normal regression method (f) with the state-of-the-art denoising methods: (b) bilateral normal filter; (c) L_0 smoothing; (d) guided normal filter; (e) the Bayesian method. The noise in the first row is synthetically generated with Gaussian noise. The noisy mesh in the second row is from Microsoft Kinect v1 with the Kinect-Fusion technique. Our denoising method recovers surface features much better than the existing methods with fine-tuned parameters, see the circular rings on the wings of the gargoyle model and David's face.*

Abstract

We present a data-driven approach for mesh denoising. Our key idea is to formulate the denoising process with cascaded non-linear regression functions and learn them from a set of noisy meshes and their ground-truth counterparts. Each regression function infers the normal of a denoised output mesh facet from geometry features extracted from its neighborhood facets on the input mesh and sends the result as the input of the next regression function. Specifically, we develop a *filtered facet normal descriptor (FND)* for modeling the geometry features around each facet on the noisy mesh and model a regression function with neural networks for mapping the FNDs to the facet normals of the denoised mesh. To handle meshes with different geometry features and reduce the training difficulty, we cluster the input mesh facets according to their FNDs and train neural networks for each cluster separately in an offline learning stage. At runtime, our method applies the learned cascaded regression functions to a noisy input mesh and reconstructs the denoised mesh from the output facet normals.

Our method learns the non-linear denoising process from the training data and makes no specific assumptions about the noise distribution and geometry features in the input. The runtime denoising process is fully automatic for different input meshes. Our method can be easily adapted to meshes with arbitrary noise patterns by training a dedicated regression scheme with mesh data and the particular noise pattern. We evaluate our method on meshes with both synthetic and real scanned noise, and compare it to other mesh denoising algorithms. Results demonstrate that our method outperforms the state-of-the-art mesh denoising methods and successfully removes different kinds of noise for meshes with various geometry features.

Keywords: Filtered facet normal descriptor, cascaded regression, mesh denoising

Concepts: •**Computing methodologies** → **Mesh models;**

1 Introduction

The prevalence of 3D scanners and depth cameras greatly simplifies the geometric modeling process and enables people to easily acquire 3D shapes from the real world. Different from the 3D meshes manually created by artists, a scanned 3D mesh always includes noise due to imperfections through the capturing and reconstruction processes. Removing the noise from the scanned meshes thus becomes an essential task in geometry processing.

The ultimate goal of mesh denoising is to recover the ground-truth surface $\mathbf{M}$ from its measurement $\mathbf{M}^{\star} = \mathbf{M} + \varepsilon$ with noise ε, which is an ill-posed problem since both the mesh and noise are unknown. Filter-based denoising methods assume the noise to be high frequency and remove the noise from the input by low-pass filters [Taubin 1995; Desbrun et al. 1999] or edge preserving filters [Jones et al. 2003; Fleishman et al. 2003; Zheng et al. 2011; Wei et al. 2015]. Although these approaches are computationally efficient and work well for various noisy inputs, users have to fine-tune the parameters and iteration numbers to obtain satisfactory results for meshes with different geometry features and noise levels. Optimization-based methods recover a mesh that best fits both the

SA '16 Technical Papers, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980232

input and a set of constraints defined by some priors of the underlying surface and noise, such as piecewise flat shape [He and Schaefer 2013], Gaussian noise or independent and identically distributed (i.i.d.) noise [Wang et al. 2014]. These approaches are automatic and could generate good results for meshes with specific geometry features or noise patterns, but they are difficult to generalize to meshes with different features and noise patterns because some assumptions may break in real scenario. For instance, the real scanner noise can be not independent Gaussian noise [Sun et al. 2009; Mallick et al. 2014]. Furthermore, the optimization process is usually slow.

In this paper, we present a data-driven method for mesh denoising. The key idea of our approach is to formulate the mesh denoising procedure as a regression function that maps the noisy input mesh to the underlying ground-truth surface. In particular, we assume the noise to be high frequency, and its magnitude to be smaller than the geometry features of the ground-truth mesh. Based on this assumption, our method models each facet normal on the ground-truth mesh as a non-linear function of geometry descriptors extracted from the local neighborhood of the corresponding facet on the input mesh. We learn the regression function from a set of noisy meshes and their ground-truth counterpart in an offline training stage. At runtime, given an input mesh, we first extract the geometry descriptors for each mesh facet and then apply the regression function to the extracted geometry descriptors to obtain the facet normals. Finally, we reconstruct the denoised mesh from the resulting facet normals computed by the regression function.

The major technical challenge in our method is how to design the geometry descriptor and the regression function. The geometry descriptor defined on each facet should not only characterize the underlying geometry features in the local region, but also be robust to the noise over the surface. Meanwhile, the regression function should not only faithfully recover the ground-truth facet normal from the noisy input, but also be efficient for both training and runtime evaluation. To this end, we define a *filtered facet normal descriptor* (FND) at each facet, which consists of responses to a series of bilateral normal filters. Because each bilateral normal filter partially removes noise and preserves geometric features at a specific scale, the FND models the local geometry features at different scales well and is robust to noise. To model the mappings between the FNDs to the ground-truth facet normal, we cluster the facets with their FND features and learn a neural network with a single hidden-layer for each cluster. Although this regression function is compact and easy to learn, it cannot totally remove the noise from the input. Therefore, we design a cascaded regression scheme to gradually remove the noise from the input. After the first regression computes the facet normals from the FNDs of the input noisy mesh, we reconstruct the mesh from the output facet normal and extract FNDs to feed into the next regression step. We repeat this procedure and reconstruct the resulting mesh from the facet normals obtained from the last regression step.

Our regression based method provides a general solution for mesh denoising. It makes no specific assumptions about the underlying geometry features and noise distributions and directly learns the denoising process from the training data. After training, the runtime denoising process is fully automatic. Our FND based cascaded normal regression scheme balances accuracy, training cost, and runtime complexity well. With the separate neural networks trained for facets with different FND features, our method not only removes the noise, but also preserves the geometric details of the underlying surface. By cascading a set of compact regression functions, our method efficiently reduces the training cost, and achieves the good accuracy with fast runtime evaluation. It can be easily adapted for denoising meshes from the specified device by training a dedicated regression scheme with meshes scanned by it. We evaluate our method on synthetic noisy meshes with different noise distributions and real datasets scanned by Kinect v1 and Kinect v2 and compare it with other state-of-the-art mesh denoising algorithms. Results show that our method outperforms the state-of-the-art mesh denoising methods in terms of result quality and efficiency.

2 Related Work

Filter-based mesh denoising methods usually assume that the noise over the surface is high frequency and design filters that operate on vertex position or facet normal for noise removal.

Early methods apply Laplacian smoothing or its improved versions [Taubin 1995; Desbrun et al. 1999] on vertex positions for mesh denoising. Although these approaches can reduce noise, they also smooth sharp features over the surface. Later methods adapt the edge-preserving filtering techniques used for image denoising to vertex positions for mesh denoising, like anisotropic diffusion [Clarenz et al. 2000; Tasdizen et al. 2002; Bajaj and Xu 2003; Yagou et al. 2003; Hildebrandt and Polthier 2004], bilateral filtering [Jones et al. 2003; Fleishman et al. 2003], and mean and median filtering [Yagou et al. 2002; Shen and Barner 2004]. With carefully selected parameters and iteration numbers for each input, these methods can remove the noise and preserve the geometry features over the surface. However, choosing appropriate parameters is a trial and error procedure even for professor users.

To better preserve sharp features over a surface, a set of recent methods apply the bilateral filter [Lee and Wang 2005; Zheng et al. 2011; Wei et al. 2015] or mean shift filter [Solomon et al. 2014] to the facet normals first and then reconstruct the mesh to fit the filtered normals [Sun et al. 2007]. The latest guided normal filtering technique [Zhang et al. 2015] utilizes reliable guided normals in the joint bilateral filter for robustly recovering geometry features from the input with relatively large noise. Although these normal based methods improve the robustness and accuracy of denoising, parameter tuning is still needed. Another set of methods first detect the geometry features from the noisy input via various techniques including quadric fitting [Fan et al. 2010], L_1 approximation [Lu et al. 2016] and normal variance clustering [Wei et al. 2015], and then they apply different filters for feature and non-feature parts separately. Although these methods can successfully recover strong features from the noisy input, the weak features that are difficult to detect are generally over-smoothed.

Our method also assumes that the noise is high frequency and performs denoising on facet normals. We apply the bilateral normal filter to extract surface descriptors for regression. By automatically clustering the facets around different geometry features with FNDs and training separate regression function for each cluster, our regression scheme automatically applies different denoising strategies for regions with different geometry features at runtime. The regression scheme learned from the training data can produce better denoising results than the general filtering techniques.

Optimization-based mesh denoising methods formulate the denoising as an optimization problem and seek for a denoised mesh that can best fit to the input mesh and a set of constraints defined by the priors of the ground-truth geometry and noise distribution. Diebel and Thrun [2006] reconstruct the mesh from noisy input with a Bayesian framework, where they assume that the noise follows the Gaussian distribution and the adjacent facet normal changes follow an application specific distribution. He and Schaefer [2013] develop a L_0 based scheme for recovering CAD-like models from noisy inputs by minimizing facet normal variations across edges. Wang et al. [2014] propose a compressive sensing solution for decoupling surface and noise and prove that their method can recover the ground-truth mesh from a piecewise C^2 smooth surface with i.i.d. noise. Although these methods can automatically recover the surface

from noisy inputs that satisfy their assumptions, they are difficult to generalize for handling mesh with different geometry features and noise patterns. In contrast, our method makes no assumptions about the underlying geometry features and noise patterns, and can be easily adapted to meshes with different patterns by training dedicated regression functions from training mesh data with new noise patterns.

Data-driven denoising methods have been used for removing noise in real images and rendering results.

For real images, Burger *et al*. [2012] demonstrate that a multilayer perceptron (MLP) trained on large image databases is competitive to other state-of-the-art filtering techniques for denoising images with fixed-level noise. Fanello *et al*. [2014] present the *filter forests* technique for learning a set of filters for image denoising and depth map refinement, where the filter kernel size and values are optimized for each data point. Xu *et al*. [2015] construct a general deep convolutional neural network for approximating different kinds of edge-aware filters for image processing. However, it is a non-trivial task to adapt these techniques working on 2D images with regular grids to 3D meshes with irregular topology connections. Also, the geometry features of a surface and its noise patterns are different from color images. Taking these differences in consideration, we propose a FND-based cascaded regression scheme that is designed for 3D mesh denoising.

For rendering results, Moon *et al*. [2014; 2015] reconstruct the rendering image from noised Monte-Carlo rendering results with linear regression functions learned from the scene information collected in image pixels. Kalantari *et al*. [2015] remove Monte Carlo noise in the rendering results with joint bilateral filters, the parameters of which are learned from training data via neural networks. All these techniques rely on extra scene information gathered on the image plane for denoising and thus cannot be used for mesh denoising, where the input is a noisy mesh itself.

For mesh denoising, Diebel and Thrun [2006] try to learn a pairwise normal potential function from application-specific geometry shapes to constrain the reconstructed geometry shapes in their optimization method. In contrast, we make no assumptions about the underlying geometry features and directly learn a cascaded normal regression scheme for denoising each facet normal.

Geometry descriptors have been well studied for 3D shape matching, recognition, and segmentation. Geometry descriptors play an important role in classifying the geometric similarity of shapes. A set of geometry descriptors have been proposed for geometry feature detection and description, such as MeshDoG, MeshHoG [Zaharescu et al. 2009], MeshSIFT [Maes et al. 2010], spin image [Johnson and Hebert 1999], etc. However, these descriptors are sparsely defined over the surface and thus cannot describe the local geometry around each facet. Kalogerakis *et al*. [2010] construct a feature descriptor that consists of a geometric feature vector and contextual label features for mesh segmentation. Although this descriptor is defined on each facet, its one value can correspond to different facet normals and thus cannot be used for our regression function. In our work we propose filtered facet normal descriptor for mesh denoising.

Cascaded regression is a popular regression scheme for approximating a highly nonlinear functions or optimization process. It has been widely used in various vision and graphics tasks, such as image deconvolution and restoration [Schelten et al. 2015; Schmidt et al. 2016], 2D pose estimation [Dollár et al. 2010], 2D and 3D face alignment and tracking [Cao et al. 2014; Lee et al. 2015; Chen et al. 2014] and hand pose estimation [Sun et al. 2015]. We apply the cascaded regression scheme for the mesh denoising task, which achieves a good balance between accuracy and computational cost.

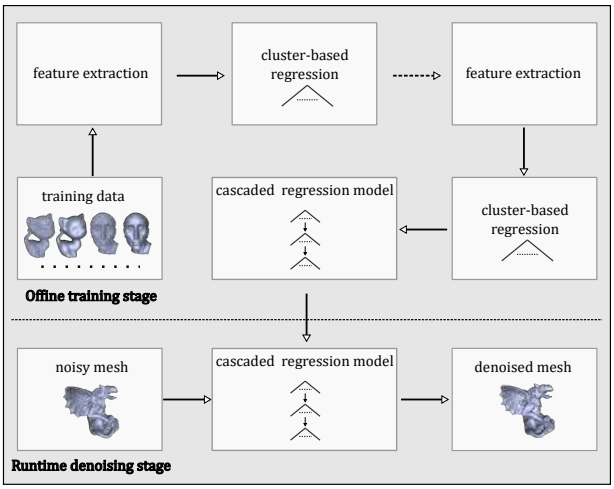


Figure 2: *The cascaded FND regression scheme.*

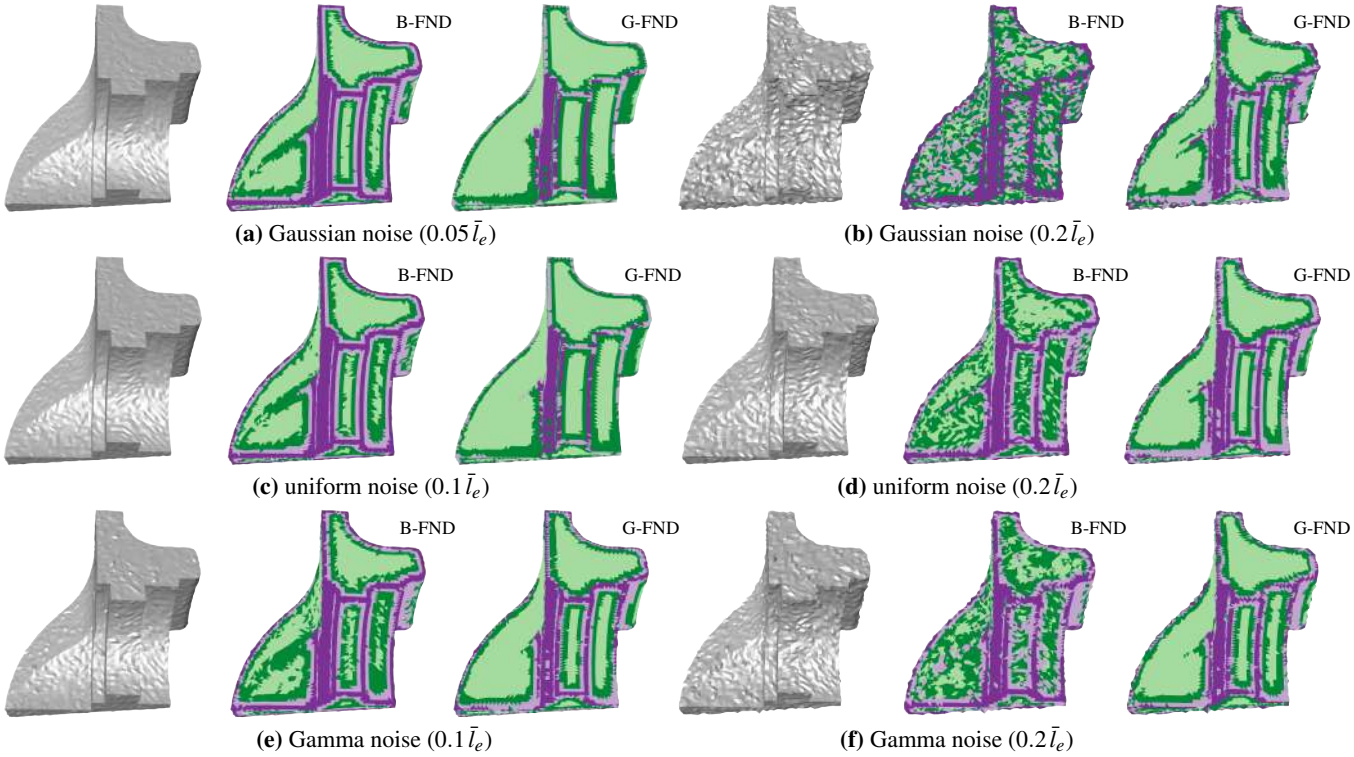
3 Algorithm overview

In this paper, we aim to learn the relationship between noisy geometry and the ground-truth geometry from a training dataset, and use the learning result for mesh denoising. By assuming that the noise to be high frequency and its magnitude to be smaller than geometry features, we model a ground-truth facet normal $\mathbf{n}_f$ as a function $\mathscr{F}$ of its local noisy region Ω_f: $\mathbf{n}_f = \mathscr{F}(\Omega_f)$. Instead of using an existing facet-normal filter and fine-tuning its parameters to approximate $\mathscr{F}$, we propose a regression-based approach to learn $\mathscr{F}$ from noisy meshes and their ground-truth counterparts. The motivation that using the facet normal but not the vertex position is inspired by the previous work on mesh denoising, which reveals that a bilateral filter on mesh facet normals [Zheng et al. 2011] could better preserve geometry features than a bilateral filter on mesh vertices [Jones et al. 2003; Fleishman et al. 2003].

However, the domain Ω_f cannot be easily parameterized, and the different local regions need to be aligned for the training. The difficulty is due to the irregular connection of 3D meshes, which is different from the image that is defined on a regular grid. To overcome this difficulty, we propose the *filtered facet normal descriptor* (FND) as the geometry descriptor based on bilateral normal filters extracted from Ω_f and FND possesses nice properties for learning: robust to noise and invariant to global rigid transformation (Section 4). We partition the training data into different clusters based on FNDs of the noisy meshes, and train regression functions separately for each cluster via a neural network with a single hidden layer (Section 5). Due to the complexity of $\mathscr{F}$, we propose a cascaded regression scheme to gradually reduce the approximation error: the output from the current regression function serves as the input of the next regression function. From the final output facet normals, the underline geometry of the noisy input mesh can be recovered via a normal-driven mesh deformation.

Before diving into the details of FNDs and our cascade regression scheme, we first sketch our algorithm at a high level. Our denoising algorithm includes two stages (see Figure 2): an offline training stage and a runtime denoising stage.

Offline training stage The input of the offline training stage is a set of noisy meshes and their noise-free counterparts (i.e. ground-truth). We extract the FND $\mathbf{S}_i$ for each facet from the noisy mesh and bind it with its ground-truth facet normal $\bar{\mathbf{n}}_i$ to get a training pair. We use a regression-based training technique to learn the function $\mathscr{F} : \mathbf{S}_i \mapsto \bar{\mathbf{n}}_i, \forall i$.



(a) Gaussian noise ($0.05\,\bar{l}_e$) **(b)** Gaussian noise ($0.2\,\bar{l}_e$)

(c) uniform noise ($0.1\,\bar{l}_e$) **(d)** uniform noise ($0.2\,\bar{l}_e$)

(e) Gamma noise ($0.1\,\bar{l}_e$) **(f)** Gamma noise ($0.2\,\bar{l}_e$)

Figure 3: *Robustness of FNDs. We add scaled Gaussian noise (zero mean, standard deviation 1) , uniform noise (in* $[-1,1]$*) and Gamma*$(2,2)$ *noise to the Fandisk model. The scale values are* $0.05\,\bar{l}_e$*,* $0.1\,\bar{l}_e$ *and* $0.2\,\bar{l}_e$*. Mesh vertices are shifted along the vertex normal by the Gaussian and Gamma noise. Uniform noise is added to the vertex coordinates. Surface regions are classified by B-FND and G-FND. Clusters are labeled with colors.*

Runtime denoising stage For a noisy input, we extract the FND for each facet and then apply the learned cascaded regression function $\mathscr{F}$ on it to obtain its new facet normal. Finally, we take the output facet normals as the normals of the denoised mesh and deform the input to match these facet normals to obtain the denoising result.

4 Design of geometry descriptor

As a key to the learning based approach, the geometry descriptor (called *feature vector* in machine learning) should be able to characterize the underlying geometry features robustly from the noisy input. We propose to use multiscale bilateral normal filters to design such a geometry descriptor.

4.1 Filtered facet normal descriptor

We first briefly introduce an edge-preserving denoiser called the *bilateral normal filter*. Denote a given 3D triangular mesh as $\mathbf{M} = (\mathbf{V},\mathbf{F})$, where $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N_v}$ represents the set of vertices, and $\mathbf{F} = \{f_i\}_{i=1}^{N_f}$ is the set of facets. Denote the centroid of facet f_i as $\mathbf{c}_i$, the normal of triangle f_i as $\mathbf{n}_i$, and the area of triangle f_i as A_i. The $(k+1)$-th iteration of the bilateral normal filter is defined as [Zheng et al. 2011]:

$$\mathbf{n}_i^{k+1} := \Lambda\Big(\sum_{f_j\in\mathbf{F}} A_j\, W_s(\|\mathbf{c}_i-\mathbf{c}_j\|)\, W_r(\|\mathbf{n}_i^k-\mathbf{n}_j^k\|)\,\mathbf{n}_j^k\Big). \qquad (1)$$

Here $\Lambda(\cdot)$ is the vector normalization operator, $\mathbf{n}_j^0 := \mathbf{n}_j$, and W_s and W_r are monotonically decreasing weighting functions characterizing the position similarity and the normal similarity between a pair of facets respectively. Gaussian functions $W_\sigma(x) = \exp(-x^2/(2\sigma^2))$ are commonly used, with standard deviations σ_s and σ_r for W_s and W_r respectively. By first filtering the facet normals then updating vertex positions to fit the facet normals [Sun et al. 2007], a noisy mesh can be denoised while the sharp features are preserved if σ_s, σ_r and the iteration number are properly set.

When the level of noise is high, the bilateral normal filter does not perform well. The *joint bilateral filter* [Kopf et al. 2007] (also known as *guided bilateral filter*) that introduces a reliable guidance to the bilateral filter is more robust to the high-level noise. Zhang *et al.* suggest to construct a proper guidance for mesh denoising [2015]. The general formulation of a guided normal filter is as follows:

$$\mathbf{n}_i^{k+1} := \Lambda\Big(\sum_{f_j\in\mathbf{F}} A_j\, W_s(\|\mathbf{c}_i-\mathbf{c}_j\|)\, W_r(\|g(\mathbf{n}_i^k)-g(\mathbf{n}_j^k)\|)\,\mathbf{n}_j^k\Big). \qquad (2)$$

Here $g(\cdot)$ is the user-defined guidance of facet normals. In our paper we use the Gaussian normal filter as the guidance, i.e., $g(\mathbf{n}_i^k) = \Lambda(\sum_{f_j\in\mathbf{F}} W_s(\|\mathbf{c}_i-\mathbf{c}_j\|)\,\mathbf{n}_j^k)$.

Bilateral filtered facet normal descriptor We rewrite the left side of Equation (1) as $\mathbf{n}_i^{k+1}(\sigma_s,\sigma_r)$ to bind the parameters σ_s and σ_r. Assume that there is a parameter set P containing a set of Gaussian parameter pairs: $P := \{(\sigma_{s_j},\sigma_{r_j})\}_{j=1}^{L}$ and the max iteration number is K. We construct a geometry descriptor $\mathbf{S}_i$ on facet f_i by assembling filtered facet normals with different parameters in P and iteration numbers:

$$\begin{aligned}\mathbf{S}_i :=& (\mathbf{n}_i^1(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_i^1(\sigma_{s_L},\sigma_{r_L}),\mathbf{n}_i^2(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_i^2(\sigma_{s_L},\sigma_{r_L}),\\ &\cdots,\mathbf{n}_i^K(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_i^K(\sigma_{s_L},\sigma_{r_L})).\end{aligned}$$

We call $\mathbf{S}_i$ a *bilateral filtered facet normal descriptor* (B-FND) of f_i. $\mathbf{S}_i$ contains the responses of $\mathbf{n}_i$ under different bilateral normal filters and iteration numbers.

Guided filtered facet normal descriptor Similarly we can construct a *guided filter facet normal descriptor* (G-FND):

$$\mathbf{S}_i^g := (\mathbf{n}_{g,i}^1(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_{g,i}^1(\sigma_{s_L},\sigma_{r_L}),\mathbf{n}_{g,i}^2(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_{g,i}^2(\sigma_{s_L},\sigma_{r_L}),\cdots,\mathbf{n}_{g,i}^K(\sigma_{s_1},\sigma_{r_1}),\cdots,\mathbf{n}_{g,i}^K(\sigma_{s_L},\sigma_{r_L})),$$

where $\mathbf{n}_{g,i}^{k+1}(\sigma_s,\sigma_r)$ denotes the left side of Equation (2).

4.2 Properties of FND

Next, we show that our FND possesses the following nice properties that are well suited for machine learning based denoising task.

Invariance with respect to rigid transformation FND is invariant to translation because FND is just the weighted average of facet normals. However, by applying a global rotation to the normals of an FND directly, one can get another version of FND. To make FND invariant to rotation, we use the normal tensor to align all the FNDs as follows. Assume that an FND $\mathbf{S}$ contains d filtered normals: $\{\mathbf{m}_1,\ldots,\mathbf{m}_d\}$. We construct the normal tensor as $\mathbf{T}=\sum_{j=1}^d \mathbf{m}_j\times\mathbf{m}_j^T$. The three eigenvectors (sorted by eigenvalues) of $\mathbf{T}$ defines a rotation matrix $\mathbf{R}$ that aligns the eigenvector frame to the coordinate axes. By multiplying each $\mathbf{m}_j$ with $\mathbf{R}^{-1}$, $\mathbf{S}$ is aligned, i.e. the rotation ambiguity is removed.

Robustness to noise An FND is composed of a series of bilateral or guided normal filters which are not sensitive to the noise in general if the magnitudes of the noise on face centers and facet normals are smaller than σ_s and σ_r, respectively. Here we show the robustness of FND on noisy synthetic data. We added two levels of Gaussian noise to the Fandisk model (Figure 3(a)&(b)). Gaussian noise perturbed the positions of mesh vertices along the normal direction. We constructed both B-FND and G-FND for all the facets of the noisy models. The parameter set P and the iteration number are set as the default values (see Section 6). For each noisy mesh, we apply a k-means algorithm to partition the mesh facets according to their aligned FNDs. We set $k=4$ and color-code the clusters. We can see that the feature and non-feature regions fall into different groups clearly. For meshes with large noise, G-FND is more robust than B-FND in distinguishing large features while B-FND is more sensitive to weak features (see the crease line on the left side of Fandisk). We also test the robustness of FND on noisy meshes with different types of noises, like uniform noise and Gamma noise, the behaviors of FNDs are very similar.

Efficiency FND is computed by bilateral-like filters that can be regarded as high-dimensional Gaussian transformation, can be sped up via the permutohedral lattice technique [Adams et al. 2010] with a time complexity of $\mathcal{O}(d^2n)$ for filtering n values in d dimensional space. Given the parameter set $\{(\sigma_{s_j},\sigma_{r_j})\}_{j=1}^L$ and the max iteration number K, the time complexity of computing all the FNDs for a noisy mesh with N_f facets is only $\mathcal{O}(L\times K\times d^2\times N_f)$, where $d=6$ is the dimension of Gaussian transformation that counts the dimensions of face centers and normals. Note that the computations of different filtered normals are totally independent, therefore they can be easily parallelized.

5 Cascaded FND regression

The properties of FND enable us to design a data-driven approach to learn the mapping from FNDs to ground-truth facet normals. We propose *cluster-based cascaded regression* to find the mapping. The cluster-based strategy splits the training dataset into small clusters according to the similarity of FNDs, and trains a regression function for each cluster instead of the whole training data. It greatly reduces the complexity of the training and speeds up the training process. The cascaded regression scheme can cascade simple regression functions to gradually reduce the approximation error and yield better denoising quality. The details of our regression scheme are given as follows.

5.1 Offine training stage

Our cascaded regression scheme consists of a set of cluster-based regression functions $\mathscr{F}_1,\mathscr{F}_2,\ldots,\mathscr{F}_t$. Each regression function represents a mapping from FND to the ground-truth normal. The output of $\mathscr{F}_i$ serves as the input of $\mathscr{F}_{i+1}$; thus, all the regression functions are connected in a cascaded manner, and t is the cascaded depth.

Training dataset The FND associated with a mesh facet and the ground-truth facet normal form a training pair. We collect the training pairs from a large set of mesh models (the ground-truth and noisy meshes are known). We denote the dataset as $\mathscr{D}=\{\mathbf{S}_i,\bar{\mathbf{n}}_i\}_{i=1}^{N_D}$ where $\mathbf{S}_i$ is the FND of face f_i and $\bar{\mathbf{n}}_i$ represents the ground-truth facet normal of f_i. The details of our synthetic and real dataset are presented in Section 6.

5.1.1 Cluster-based regression

We first introduce how to compute the cluster-based regression.

Feature vector extraction The FNDs are computed for all the noisy meshes in the dataset given the parameter set $\{(\sigma_{s_j},\sigma_{r_j})\}_{j=1}^L$ and the max iteration number K. The FND vector is composed of $L\times K$ filtered normals, and its vector dimension is $3\times L\times K$. We assume that each feature descriptor and its corresponding ground-truth facet normal have been reoriented via a rotation according to the procedure of Section 4.2.

Cluster-based regression For a dataset $\mathscr{D}_{\mathbf{X}}$ containing a set of training pairs $\{\mathbf{S}_i,\bar{\mathbf{n}}_i\}_{i=1}^{N_X}$, we build a cluster-based regression function to train the function that maps $\mathbf{S}_i$ to $\bar{\mathbf{n}}_i$. We first partition the training data into K_c clusters via a k-means algorithm, so that facets with similar geometric features can be grouped together. Then for each cluster C_l, we split the data into two sets: the training set $\mathscr{D}_{\mathbf{X}(C_l)}$ and the validation set $\mathscr{D}_{\mathbf{V}(C_l)}$. Here we pick 15% data randomly from C_l to form the validation set. we train a regression function Φ_l by minimizing the following cost function:

$$E := \sum_{f_i\in\mathscr{D}_{\mathbf{X}(C_l)}} \|\Lambda(\Phi_l(\mathbf{S}_i))-\bar{\mathbf{n}}_i\|^2+\lambda E_{reg}.$$

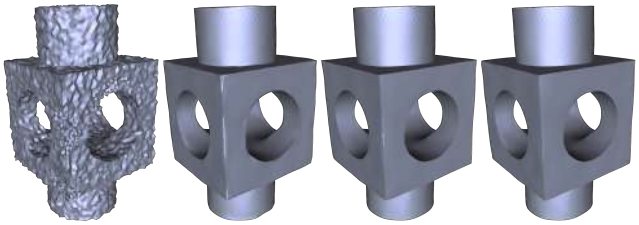
The E_{reg} is the commonly used L_2 regularization term of the unknown parameters of $\Phi_l(\cdot)$ to avoid overfitting. In our experiments, we found that $\lambda=0.02$ provides a good balance of the regression error and the regularization term. We design the function $\Phi_l(\cdot)$ as a single-hidden layer feed forward network (SLFN) with N_r hidden nodes for efficient training. In our implementation, we model it with the radial basis function (RBF):

$$\Phi_l(\mathbf{S})=\sum_{k=1}^{N_r}\exp(-\|\mathbf{W}_{l,k}^T\bar{\mathbf{S}}-b_{l,k}\|^2)\,\mathbf{a}_{l,k}.$$

Here $\bar{\mathbf{S}}$ is the feature standardization version of $\mathbf{S}$ in cluster C_l. The parameters $\mathbf{W}_{l,k}\in\mathbb{R}^{3LK}, b_{l,k}\in\mathbb{R}, \mathbf{a}_{l,k}\in\mathbb{R}^3$ are initialized according to the Nguyen-Widrow initialization algorithm [Nguyen and Widrow 1990], and then optimized via the conjugate gradient method. For avoiding overfitting and increasing generalizability, We stop the iteration of conjugate gradient method once the approximation error on the validation set $\sum_{f_i\in\mathscr{D}_{\mathbf{V}(C_l)}}\|\Lambda(\Phi_l(\mathbf{S}_i))-\bar{\mathbf{n}}_i\|^2$ increases. The collection of $\Phi_1,\ldots,\Phi_{K_c}$ forms the regression function

$$\mathscr{F}(\mathbf{S}) := \Phi_l(\mathbf{S}),\ \texttt{if}\ \|\mathbf{S}-\mathbf{c}_l\|\le\|\mathbf{S}-\mathbf{c}_k\|,\forall k.$$

Here $\mathbf{c}_k$ is the cluster center of C_k.



(a) Input **(b)** G-FND **(c)** B-FND **(d)** Both

Figure 4: *(a) The input noisy CAD model. (b) The denoised result by the cascaded regression using G-FND in each regression function. (c) The denoised result by the cascaded regression using B-FND in each regression function. (d) The denoised result by the cascaded regression with both G-FND and B-FND features. The average angular differences to the ground-truth are* 19.8°, 3.38°, 2.50°, 2.31°. *Here all the cascaded regressions are trained from the synthetic dataset.*

Mesh reconstruction The normalized regression output are the denoised facet normals: $\hat{\mathbf{n}}_i := \Lambda(\mathscr{F}(\mathbf{S}_i))$, with which the noisy mesh are updated. We follow the iterative approach of [Sun et al. 2007] to get the updated vertex positions:

$$\mathbf{v}_i^{new} := \mathbf{v}_i^{old} + \frac{1}{3|\Omega(\mathbf{v}_i)|} \sum_{f_k \in \Omega(\mathbf{v}_i)} \sum_{\mathbf{e}_{ij} \in \partial f_k} (\hat{\mathbf{n}}_k \cdot (\mathbf{v}_j^{old} - \mathbf{v}_i^{old}))\, \hat{\mathbf{n}}_k.$$

Here $\Omega(\mathbf{v}_i)$ is the one-ring neighborhood of $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ is the edge of triangle f_k with end vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. The iterative approach is essentially a gradient decent method with a fixed step-size for the following semi-definite quadratic energy function [Sun et al. 2007, Corollary 1] $\sum_{f_i} \sum_{\mathbf{e}_{jk} \in \partial f_i} \left[\hat{\mathbf{n}}_i \cdot (\mathbf{v}_j - \mathbf{v}_k)\right]^2$. In our implementation, the iteration number is set to 20.

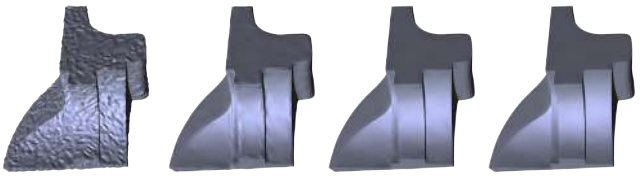
5.1.2 Cascaded scheme

We use the output of $\mathscr{F}_i$ to denoise the noisy meshes in the training dataset, then compute the FNDs from these updated meshes and use them as input to build the next regression function $\mathscr{F}_{i+1}$. We stop cascading the regression functions when the total cost on the total validation set $\sum_{f_i \in \bigcup_l \mathscr{D}_{\mathbf{v}(C_l)}} \|\hat{\mathbf{n}}_i - \bar{\mathbf{n}}_i\|^2$ is not decreasing.

Use of G-FND We found that using the G-FNDs to train the first regression function $\mathscr{F}_1$ can reduce more noise than using B-FND, especially when the noise has large variation. However, using G-FND in the rest regression functions tends to smooth sharp features because G-FND is not sensitive to them. So we only use G-FND in the first regression function and use B-FND in the remaining cascaded regression functions to preserve and recover sharp features. Figure 4 shows that combining both G-FND and B-FND in the cascaded regression is better than using one of them only.

Choices of regression schemes Although it is possible to learn a direct mapping from FND to the groundtruth facet normal, the resulting model will be much more complicated than our scheme. This not only results in difficulty in the training stage (i.e. much more training data to avoid overfitting and suboptimal result due to model non-linearity), but also leads to lengthy computation at run-time. Compared to single-level regression, the cascaded scheme used in our solution significantly improves the result accuracy, especially for sharp geometric features and real scanned meshes with large noise. Figure 5 shows the intermediate denoising outputs of a noisy mesh. The cascaded regression is trained from a synthetic dataset (Section 6.2). We can see that the cascaded regression gradually reduces noise and recovers geometric features.

In our implementation, we use the RBF function as the regression function. It is possible to use other forms of non-linear regression functions to approximate $\mathscr{F}$. We also tested the linear regression and found that its training error increases about 40% compared to our RBF-based regression on our synthetic dataset (Section 6.2). This indicates that the linear regression is not appropriate due to its weak approximation ability.



(a) Input **(b)** $\mathscr{F}_1$ **(c)** $\mathscr{F}_2$ **(d)** $\mathscr{F}_3$

Figure 5: *An input noisy mesh (a) is denoised by our cascaded regression. (b), (c) and (d) are the output of $\mathscr{F}_1$, $\mathscr{F}_2$ and $\mathscr{F}_3$, respectively. The average angular differences to the ground-truth are* 19.84°, 3.38°, 2.54°, 2.41°.

5.2 Runtime denoising stage

Once the cascaded regression scheme is obtained, we can use it to denoise meshes that share noise characteristics similar to the training data. Figure 2 shows the denoising procedure:

1. For each facet of the input mesh, compute its G-FND and feed it to the regression function $\mathscr{F}_1$.
2. The input mesh is updated by the new facet normals. For each facet of the updated mesh, we computed the B-FND and feed it to $\mathscr{F}_2$ to obtain a new facet normal. This procedure is executed sequentially until the last regression function outputs the new facet normal. The output mesh of the last regression function is the denoised mesh.

6 Experiments and comparisons

We trained and tested our regression scheme on both synthetic and real dataset. The experiment settings and comparisons are detailed as follows.

6.1 Implementation details

Choice of hyperparameters In the training stage, the parameter set $P := \{(\sigma_{s_j}, \sigma_{r_j})\}_{j=1}^{L}$ and the max iteration number K determine the construction of FNDs. Here is our default setting for these parameters:

1. The set of σ_s is $\{\bar{l}_e, 2\bar{l}_e\}$, where $\bar{l}_e$ is the average edge length.
2. The set of σ_r is $\{0.1, 0.2, 0.35, 0.5, \infty\}$.
3. $K = 1$.

Note that the bilateral normal filter degenerates to the traditional Gaussian filter when $\sigma_r = \infty$. The combinations of σ_s and σ_r from the above sets form P. In our experiments, we found our method with these parameters can achieve good performance and results, and enlarging the parameter set or K improves the results slightly but with a higher computational cost.

On our synthetic and real dataset, our cross validation shows that 3 cascaded regressions are enough to generate good results.

For the partition number K_c, we tried the values $2, 4, 8, 12$ and examine the total cost of the validation set. We found that parameter $K_c = 4$ is a good choice that has a low cost on both the synthetic and real datasets. A larger K_c improves the results slightly.

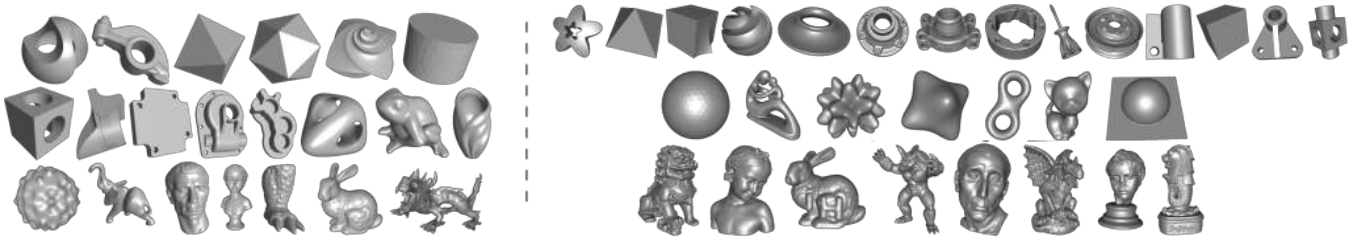


Figure 6: *Left: ground-truth models of training data. Right: ground-truth models of benchmark data (first row: CAD models; second row: smooth models; third row: models with rich features.)*

Model	Fig.12-1	Fig.11-1	Fig.10-1	Fig.10-2	Fig.1-1	Fig.10-3
N_f	10k	25k	54k	99k	171k	566k
Bilateral normal	1.2s	2.7s	6.3s	14.2s	23.7s	71.4s
L_0 smoothing	4.7s	37.1s	286.2s	622.4s	885.2s	3155.7s
Guided normal	2.6s	7.2s	19.2s	44.9s	99.7s	558.9s
Bayesian	6.1s	16.5s	39.1s	76.6s	126.2s	394.6s
Our method	**0.8**s	**1.8**s	**2.9**s	**5.7**s	**11.3**s	**28.3**s

Table 1: *Timing comparisons with other state-of-the-art methods. The first row shows the model position in the corresponding figure, e.g. Fig.12-1 is the model in the 1^{st} row of Fig.12. The second row is the facet number of the model. The running time is in seconds.*

For the number of hidden nodes N_r, we tested the following numbers: 10, 20, 30, 50, 100. We found that $N_r = 20$ achieves similar good results to other large N_rs. We choose it as the default setting because it uses less RBF basis.

Timing We conducted our experiments on a desktop PC with a 2.4 GHz Intel Xeon CPU and 36 GB RAM. We implemented our cascaded normal regression in MATLAB with parallelization. Our regression is fast at runtime and the performance comparison on typical models is summarized in Table 1. As shown in the table, the non-linear optimization based denoising algorithms such as L_0 smoothing [He and Schaefer 2013] and the Bayesian method [Diebel et al. 2006] is very slow, especially on large models. The filter based denoising method such as the bilateral normal filter [Zheng et al. 2011] and the guided normal filter [Zhang et al. 2015] usually need to run about 20 iterations to get relatively good results but they are also slower than our method.

6.2 Experiments on synthetic dataset

Training dataset We collected three types of geometry models as ground-truth data: CAD-like models, smooth models, and models with multi-scale geometry features. Figure 6-left shows our ground-truth meshes (1 millon facets in total). The noisy input models were generated by adding fixed-scale Gaussian noise to the vertices of the ground-truth meshes along the vertex normals. The standard deviations of the Gaussian noise are $0.1\bar{l}_e$, $0.2\bar{l}_e$ and $0.3\bar{l}_e$ ($\bar{l}_e$ is the average edge length of each mesh). In total, 62 noisy meshes were constructed.

The ground-truth facet normal and its corresponding FND on the noisy mesh form a training pair. To avoid the case that some dense meshes overwhelm the training procedure, if the facet number of a mesh is larger than 10k, we clustered the facets of this mesh into 4 clusters by using its FND features, and uniformly sample mesh facets from each cluster and choose totally 10k faces as ground-truth facets. Otherwise, we chose all of its facets. Finally, about 600k training pairs were collected, which contain enough variations in noise level and surface features. Our cascaded normal regression took about 6 minutes in training.

Benchmark dataset Our benchmark data also includes three types of models. Figure 6-right shows the noise-free models. We added similar Gaussian noise on these models. In total, there are 87 noisy meshes for the benchmark.

Competitors We chose five state-of-the-art methods for our benchmark: the bilateral filter [Fleishman et al. 2003], the bilateral normal filter [Zheng et al. 2011], the guided normal filter [Zhang et al. 2015], L_0 smoothing [He and Schaefer 2013], and the Bayesian method [Diebel et al. 2006]. Since each method has its own parameters, we select three parameter settings for each method, with one of them being the parameter settings suggested by the authors. We use the symbol A, B, C, D, E to denote the above methods, and "symbol+number" denote a method with one of the parameter settings, like A1, A2, and A3. Symbol F denotes our method. The parameters of the competitors are as follows:

- A1, A2, A3: bilateral filter [Fleishman et al. 2003]. The iteration number is 5, 15, 30 respectively. Other parameters are the same as the suggested values by the authors.
- B1, B2, B3: bilateral normal filter [Zheng et al. 2011]. $\sigma_s = \bar{l}_e$, the iteration number of normal update and vertex update are 20. σ_r is $0.2, 0.35$ and 0.45 respectively.
- C1, C2, C3: guided normal filter [Zhang et al. 2015]. $\sigma_s = \bar{l}_e$, the iteration number of normal updates and vertex updates are 20. $\sigma_r = 0.2, 0.35$ and 0.45 respectively.
- D1, D2, D3: L_0 smoothing [He and Schaefer 2013]. $\lambda = \lambda_0, 2\lambda_0, 3\lambda_0$ respectively. λ_0 is the default value recommended by the authors.
- E1, E2, E3: the Bayesian method [Diebel et al. 2006]. $\sigma = \bar{l}_e, 2\bar{l}_e, 3\bar{l}_e$ respectively. σ is the deviation of the Gaussian distribution used in the Bayesian method.

Quality metrics We evaluate the denoised mesh quality with the following metrics.

- The average angular difference (in degrees) between the facet normals of the denoised mesh and the ground-truth facet normals.
- The average one-sided Hausdorff distance from the denoised mesh to the known ground-truth mesh. It is normalized by the diagonal length of the bounding box of the mesh.

Benchmark results We count all the normal angular differences on all the meshes and denotes their average as D^n. The average one-side Hausdorff distances are linearly combined over the meshes weighted with facet numbers, denoted by D^v. We compare these metrics on different types of models and illustrate them via bar charts (see Figure 7, shorter (red bar) is better). Our method outperforms the other methods on these metrics.

We also compared the visual quality of the results. Figure 1-upper and Figure 10 shows the comparisons on four noisy meshes. The results from our method are more faithful to the ground-truth. For the gargoyle model, the screwdriver model and the Nicolo model, our method recovers more small details than the others and has no over-sharpening effect. For instance, our result has more details on the

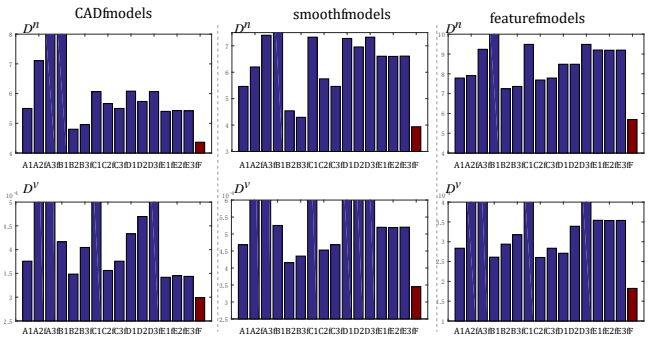


Figure 7: *Comparisons with state-of-the-art methods on the benchmark dataset. Bar charts in 1st row: the average normal angular difference D^n. Bar charts in 2nd row: the average distance D^v. Each column corresponds to one type of models. Higher bars are truncated for better illustration.*

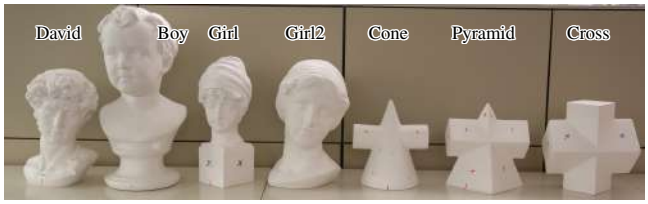


Figure 8: *Models for real scan.*

mouth and eyes of the Nicolo model while the guide-normal filter, L_0 smoothing and the Bayesian method over-sharpen these regions. For the Merlion model, our method removes spatially variant noise well, and recovers more details and features, for instance, the characters on the base are much clearer than the others. Note that, to make the comparison fair, we have fine-tuned the parameters of the other methods and selected their best results.

6.3 Experiments on Real scans

High-resolution ground-truth data We scanned seven models made by plaster and resin (see Figure 8) using Artec Spider$^{\text{TM}}$ scanner (accuracy 0.5 mm). The height of models is about 35 cm. For each model $\mathbf{M}$, the high-resolution scanned result is regarded as the ground-truth $\mathbf{M}^h$.

Training dataset We created three kinds of training datasets and trained them separately.

- Data-I: Meshes scanned by Microsoft Kinect v1. Each mesh is extracted from a single depth frame. There are 71 scans, and the total number of facets is about 2.6M. We scanned David, Girl2, Pyramid and Cone from different views.
- Data-II: Meshes scanned by Microsoft Kinect v2. Each mesh is extracted from a single depth frame. There are 72 scans, and the total number of facets is about 930k. We scanned David, Girl2, Pyramid and Cone from different views.
- Data-III: Meshes scanned by Microsoft Kinect v1 via the Kinect-Fusion technique [Izadi et al. 2011]. Each scan covers about 180 degrees of the object. The scanned models are Girl, Girl2 and Cone. The total number of facets is about 200k.

The distance from the model to the Kinect camera is about 90 cm. The noise characteristics of Kinect v1 and v2 are different [Sarbolandi et al. 2015] due to their different scanning principles: v1 is based on structured light and v2 is based on time-of-flight. The noise in Data-I and Data-II are mainly from these scanning principles.

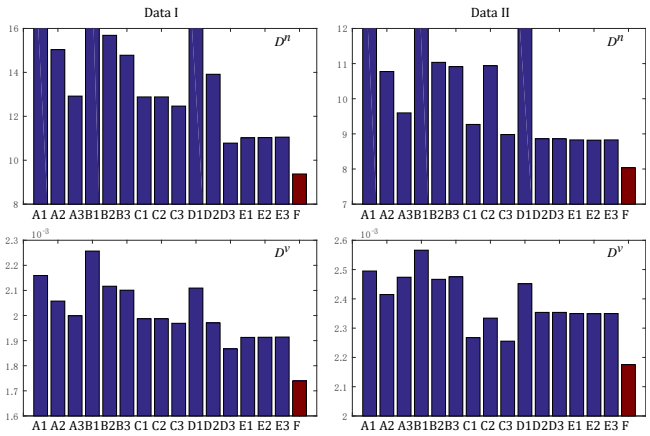


Figure 9: *Comparisons with state-of-the-art methods on the Kinect benchmark datasets. 1^{st} column: bar charts of D^n and D^v for Data I (Kinect v1). 2^{nd} column: bar charts of D^n and D^v for Data II (Kinect v2). Higher bars are truncated for better illustration.*

Data-I has larger noise than Data-II. And to deal with the large noise, the set of σ_s is enlarges as $\{\bar{l}_e, 2\bar{l}_e, 4\bar{l}_e\}$ for Data-I. For Data-III, mesh reconstruction from Kinect-Fusion is also a source of noise.

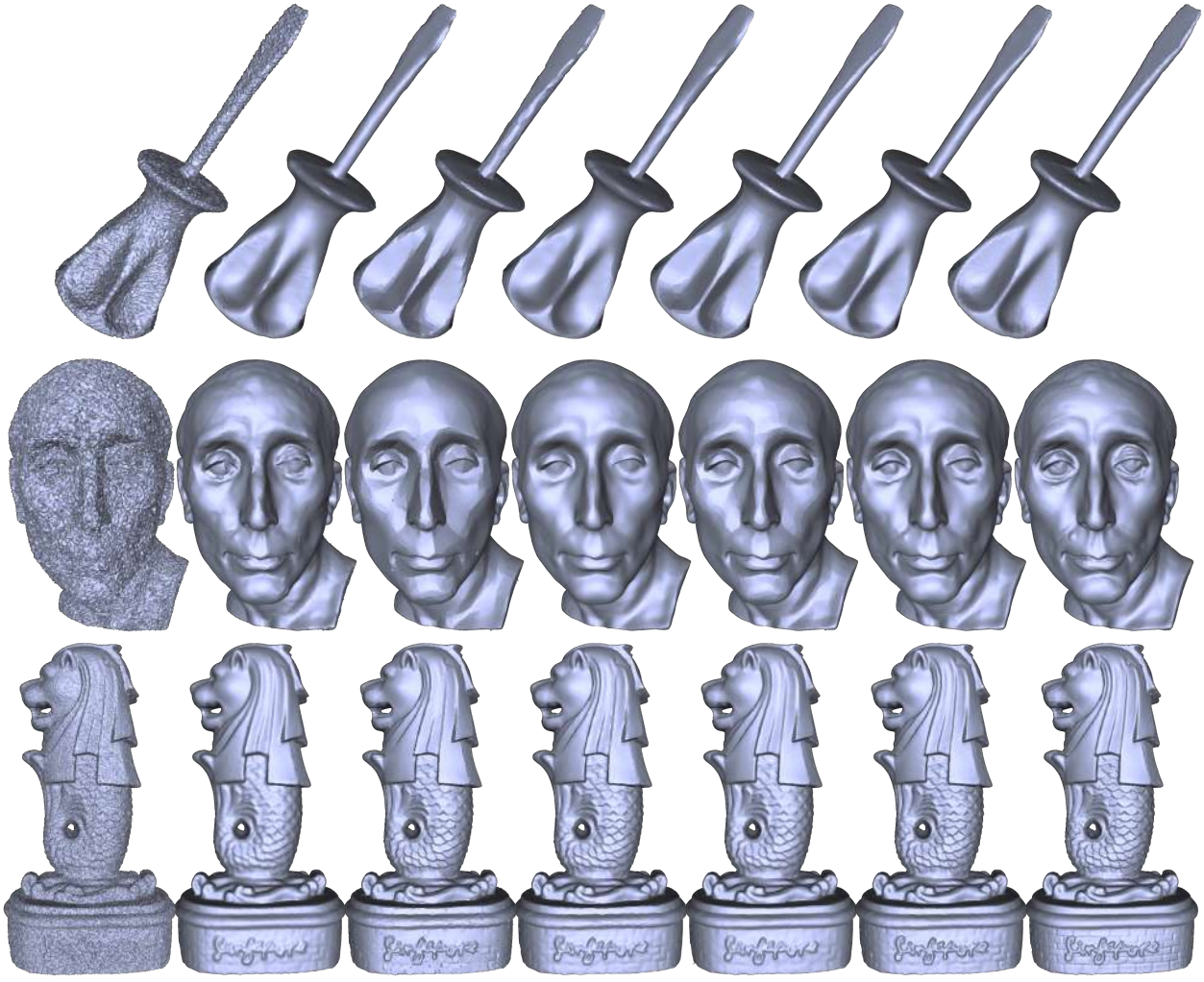
For these noisy scanned data, we created their ground-truth counterparts that have the same mesh connectivity as follows. We first registered the noisy mesh $\mathbf{M}^s$ and its high-resolution ground-truth $\mathbf{M}^h$ by a rigid ICP. For Data-III, we projected each vertex of $\mathbf{M}^s$ onto its nearest point on $\mathbf{M}^h$; for Data-I and Data II, since we know the projection matrix of the Kinect camera, we projected each vertex onto $\mathbf{M}^h$ along its projection direction. We denote the projected mesh as $\mathbf{M}^g$ and perturbed its vertices to make it best fit to $\mathbf{M}^h$ by minimizing the following function: $\sum_{f_k} \int_{\mathbf{p}\in f_k} \text{dis}(\mathbf{p}, \mathbf{M}^h)^2 \, \text{d}\mathbf{x}$. For Data-I and Data-II, we only allow mesh vertices to move along its projection direction. $\{\mathbf{M}^s, \mathbf{M}^g\}$ serves as the noisy input and its ground-truth counterpart.

For each dataset, we use the same sampling strategy used in the synthetic dataset to determine the training dataset and the validation dataset. The training time for Data-I and Data-II is about 5 minutes, the time for Data-III is about 1 minutes.

Benchmark For data-I and data-II, we use the scans of Boy, Girl and Cross as the benchmark dataset (73 scans, 930k facets). For data-III, we use the scans of David, Boy and Pyramid as the benchmark dataset (300k facets). We scanned them in the manner used for creating training data. We compared our method with other state-of-the-art methods that are the same methods in Section 6.2. the results (Fig. 9) show that our method outperforms the others and is significantly better in reducing noisy bumps on smooth regions and preserving features as shown in Figure 11, 12, 13 and 1 (2^{nd} row). It is clear that ours are more faithful to the ground-truth while the others oversharpen the features and/or have more bumps on smooth regions.

7 Conclusion and discussion

We present a novel data-driven approach for mesh denoising. We formulate the denoising process as a regression function that maps the filtered facet normal descriptor to the ground-truth facet normal, and use the updated facet normals to denoise the noisy input. The cascaded normal regression scheme we trained has no parameters to tune at runtime, and it outperforms the existing state-of-the-art methods. We believe that our approach will find more applications especially in improving the accuracy of low-resolution 3D scanners.



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth

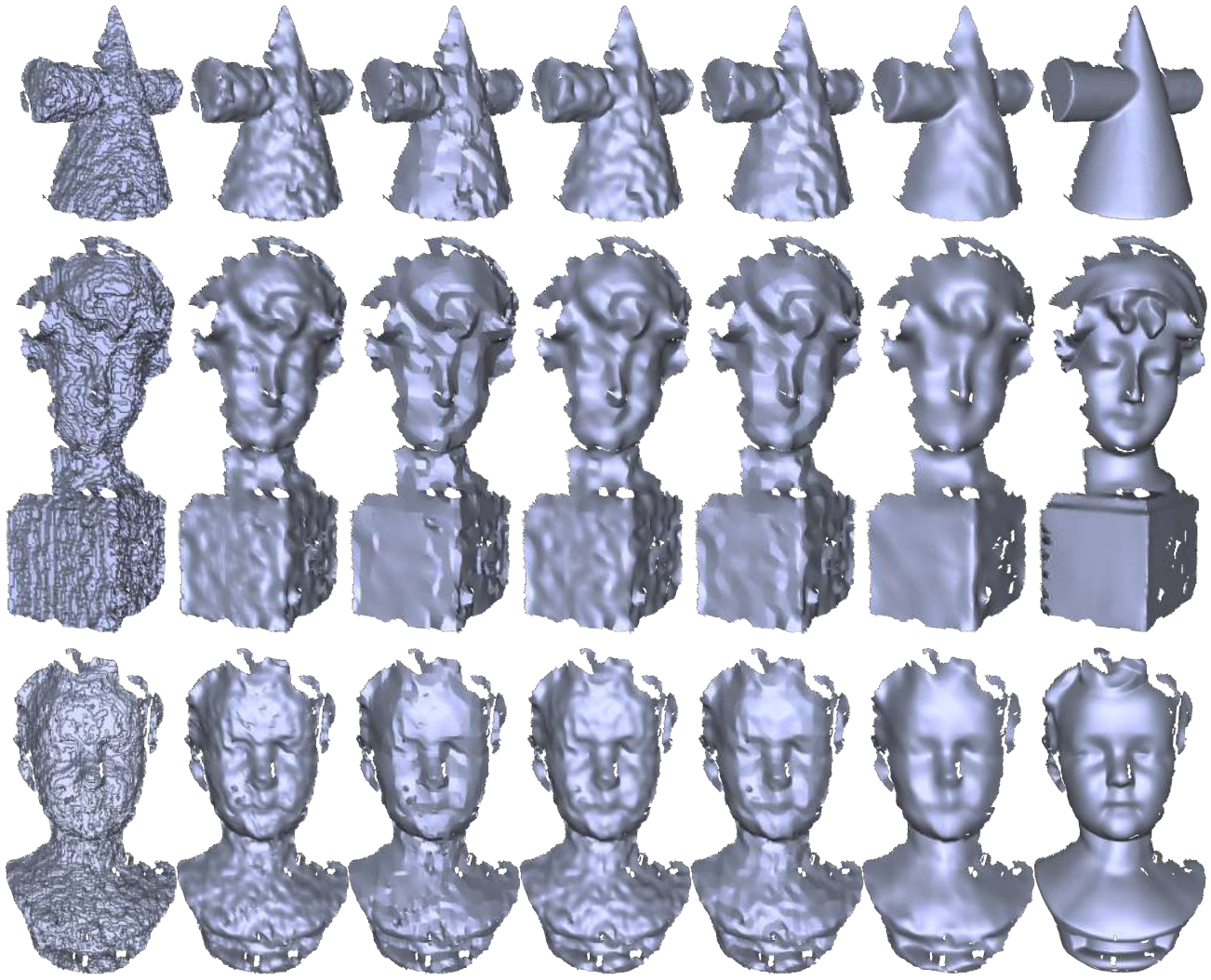
Figure 10: *Visual comparisons of denoised results. Ground-truth models: a screwdriver model (1^{st} row), the Nicolo model with many small features (2^{nd} row) and the Merlion model with rich details (3^{rd} row). Gaussian noise with the same level are added to the first two models. A spatially variant Gaussian noise is added to the Merlion model. The average normal angular differences are: (1^{st} row)* 21.34°, 4.44°, 5.60°, 4.69°, 4.35°, ***3.02***°*; (2^{nd} row)* 30.92°, 5.95°, 6.87°, 5.86°, 5.86°, ***5.12***°*; (3^{rd} row)* 24.56°, 6.07°, 7.34°, 5.68°, 5.54°, ***4.19***°.

Discussions Concurrent to our work, Boulch and Marlet [2016] present a deep learning based solution for point cloud normal estimation. They actually use the Hough accumulator as the feature vector for normal estimation. Compared to our cascaded regression scheme, their training and evaluation process are time-consuming, and the normals around sharp features cannot be well recovered (see Figure 11 of their work). In future we would like to develop an end-to-end deep learning system to combine the feature learning process and regression process together, and replace our handcrafted FND and the cascaded regression scheme with it.

Our regression framework is general for users to quickly learn a dedicated regression scheme for meshes generated by a specific scanning and reconstruction process from a small set of training data. The cascaded regression scheme obtained from the training data can be applied to noisy meshes that share the similar noise pattern (e.g meshes generated by the similar scanning/reconstruction process). We also tested our regression schemes on meshes with different noise distributions and found that the results look acceptable if the amplitudes of these noise distributions are not very different. For instance, our cascaded regression scheme learned from meshes with Gaussian noise (zero mean, standard deviation 1) works well for meshes with uniform noise distribution (sampled in $[-1,1]$).

Limitations The quantity and quality of the training dataset affect the capability of training-based methods. In our method, we assume that the noise and the geometric features of the input data can be distinguished by the FND. If the geometry variation of the noise of some models is similar to the small features of other models, the mapping found by our method may smooth such features or fail to remove such noise. This case is also hard to handle for other methods such as filter-based and optimization-based methods due to the indistinguishable variations.

Our method also cannot handle large positional bias between the noisy mesh and ground-truth mesh. For instance, objects scanned by time-of-flight scanners may have a global positional bias on regions where the light travels a long distance due to multiple reflections. Such a bias breaks our assumption that the majority of the noise is only related to the local geometry of the object. The result of our method can have a good approximation to facet normals, but the absolute position still differs from the ground-truth. A possible solution is to combine both normal and positional information to build a geometry descriptor and finding the mapping from such a geometry descriptor to the ground-truth normal and position.



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth

Figure 11: *The denoised Kinect v1 single-frame meshes. The average normal angular differences are: (1st row)* 34.9°, 16.4°, 15.0°, 13.4°, 12.4°, ***8.8***°*; (2nd row)* 33.8°, 18.1°, 16.7°, 16.0°, 15.1°, ***13.5***°*; (3rd row)* 32.2°, 15.8°, 14.2°, 13.2°, 12.6°, ***10.6***°.

Acknowledgements

Meshes used in synthetic dataset are courtesy of the Aim@Shape repository and 3D mesh database of the Inria GAMMA group. We thank the anonymous reviewers for their valuable feedback.

References

ADAMS, A., BAEK, J., AND DAVIS, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. *Comput. Graph. Forum (EG) 29*, 2, 753–762.

BAJAJ, C. L., AND XU, G. 2003. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graph. 22*, 1, 4–32.

BOULCH, A., AND MARLET, R. 2016. Deep learning for robust normal estimation in unstructured point clouds. *Comput. Graph. Forum (SGP) 35*, 5.

BURGER, H., SCHULER, C., AND HARMELING, S. 2012. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, 2392–2399.

CAO, C., HOU, Q., AND ZHOU, K. 2014. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph. (SIGGRAPH) 33*, 4, 43:1–43:10.

CHEN, D., REN, S., WEI, Y., CAO, X., AND SUN, J. 2014. Joint cascade face detection and alignment. In *ECCV*, 109–122.

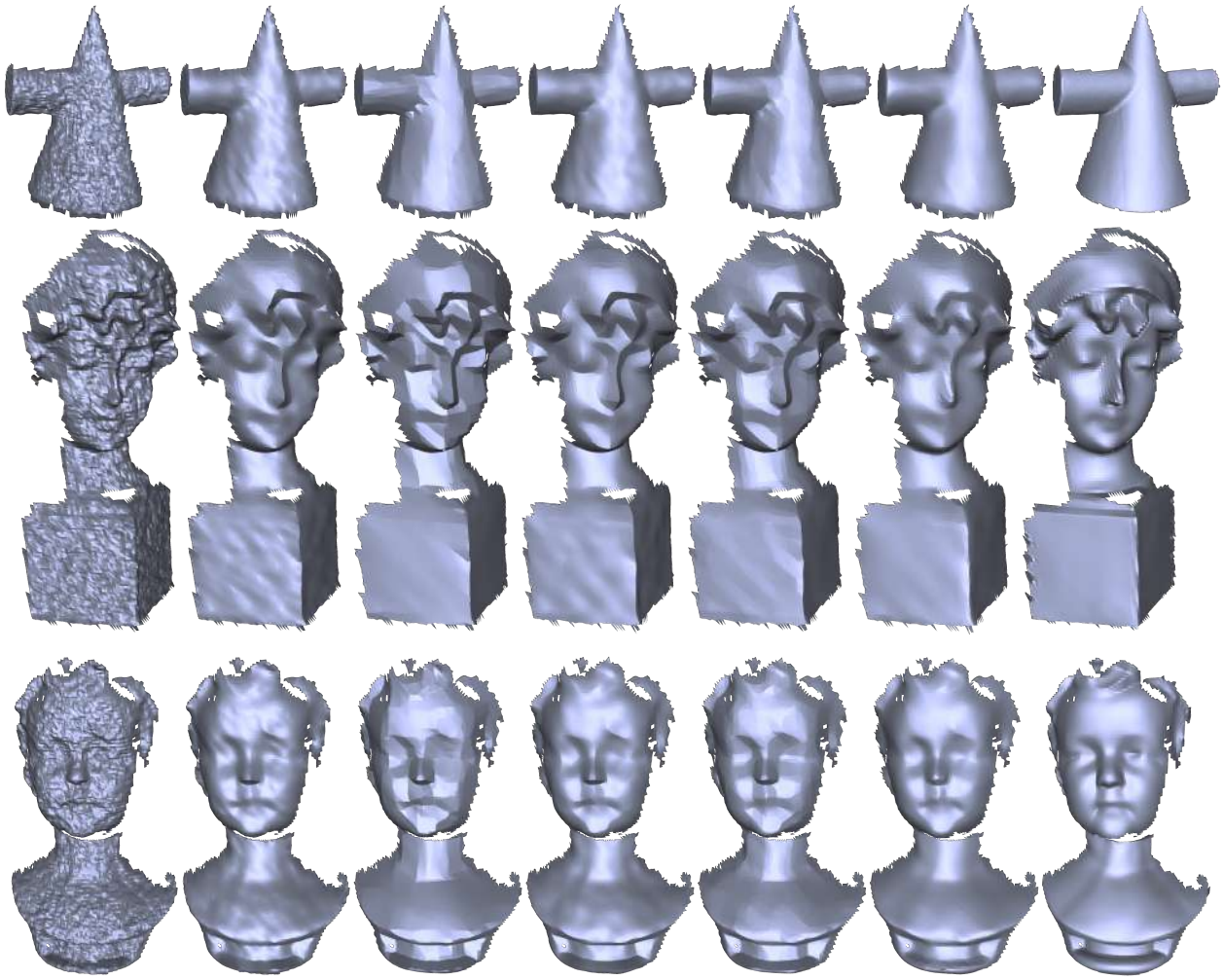
CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *Proc. of the conference on Visualization*, 397–405.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*, 317–324.

DIEBEL, J. R., THRUN, S., AND BRÜNIG, M. 2006. A Bayesian method for probable surface reconstruction and decimation. *ACM Trans. Graph. 25*, 1, 39–59.

DOLLÁR, P., WELINDER, P., AND PERONA, P. 2010. Cascaded pose regression. In *CVPR*, 1078–1085.

FAN, H., YU, Y., AND PENG, Q. 2010. Robust feature-preserving mesh denoising based on consistent subneighborhoods. *IEEE. T. Vis. Comput. Gr. 16*, 2, 312–324.



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth

Figure 12: *The denoised Kinect v2 single-frame meshes. The average normal angular differences are: (1st row)* 24.4°, 10.4°, 9.0°, 8.8°, 9.9°, ***7.9***°*; (2nd row)* 25.6°, 13.4°, 12.5°, 12.1°, 11.7°, ***11.3***°*; (3rd row)* 24.3°, 12.2°, 12.2° , 11.4°, 11.2°, ***9.9***°.

FANELLO, S., KESKIN, C., KOHLI, P., IZADI, S., SHOTTON, J., CRIMINISI, A., PATTACINI, U., AND PAEK, T. 2014. Filter forests for learning data-dependent convolutional kernels. In *CVPR*, 1709–1716.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Trans. Graph. (SIGGRAPH) 22*, 3, 950–953.

HE, L., AND SCHAEFER, S. 2013. Mesh denoising via L_0 minimization. *ACM Trans. Graph. (SIGGRAPH) 32*, 4, 64:1–64:8.

HILDEBRANDT, K., AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. *Comput. Graph. Forum (EG) 23*, 3, 391–400.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 559–568.

JOHNSON, A., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 5, 433–449.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. (SIGGRAPH) 22*, 3, 943–949.

KALANTARI, N. K., BAKO, S., AND SEN, P. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph. (SIGGRAPH) 34*, 4, 122:1–122:12.

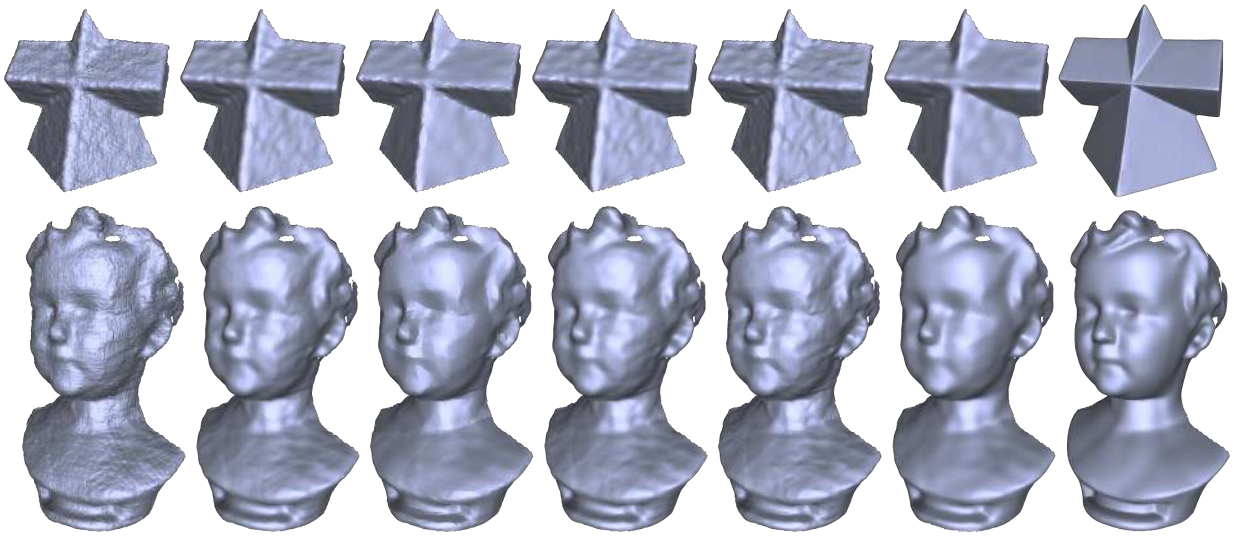
KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D mesh segmentation and labeling. *ACM Trans. Graph. (SIGGRAPH) 29*, 4, 102:1–102:12.

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graph. (SIGGRAPH) 26*, 3, 96:1–96:5.

LEE, K.-W., AND WANG, W.-P. 2005. Feature-preserving mesh denoising via bilateral normal filtering. In *Ninth International Conference on Computer Aided Design and Computer Graphics*.

LEE, D., PARK, H., AND YOO, C. 2015. Face alignment using cascade Gaussian process regression trees. In *CVPR*, 4204–4212.

LU, X., DENG, Z., , AND CHEN, W. 2016. A robust scheme for feature-preserving mesh denoising. *IEEE. T. Vis. Comput. Gr. 22*, 3, 1181–1194.



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth

Figure 13: *The denoised Kinect-fusion meshes. It is clear that our results have fewer bumps on smooth and flat regions than the others. The average normal angular differences are: (1st row)* 13.6°, 10.1°, 7.6°, 8.5°, 9.2°, ***7.5***°*; (2nd row)* 13.7°, 9.4°, 8.1°, 8.1°, 8.9°, ***7.7***°.

MAES, C., FABRY, T., KEUSTERMANS, J., SMEETS, D., SUETENS, P., AND VANDERMEULEN, D. 2010. Feature detection on 3D face surfaces for pose normalisation and recognition. In *Biometrics: Theory Applications and Systems (BTAS)*, 1–6.

MALLICK, T., DAS, P., AND MAJUMDAR, A. 2014. Characterizations of noise in Kinect depth images: A review. *IEEE Sensors J. 14*, 6, 1731–1740.

MOON, B., CARR, N., AND YOON, S.-E. 2014. Adaptive rendering based on weighted local regression. *ACM Trans. Graph. (SIGGRAPH) 33*, 5, 170:1–170:14.

MOON, B., IGLESIAS-GUITIAN, J. A., YOON, S.-E., AND MITCHELL, K. 2015. Adaptive rendering with linear predictions. *ACM Trans. Graph. 34*, 4, 121:1–121:11.

NGUYEN, D., AND WIDROW, B. 1990. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, 21–26.

SARBOLANDI, H., LEFLOCH, D., AND KOLB, A. 2015. Kinect range sensing: structured-light versus time-of-flight Kinect. *Journal of Computer Vision and Image Understanding 13*, 1–20.

SCHELTEN, K., NOWOZIN, S., JANCSARY, J., ROTHER, C., AND ROTH, S. 2015. Interleaved regression tree field cascades for blind image deconvolution. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, 494–501.

SCHMIDT, U., JANCSARY, J., NOWOZIN, S., ROTH, S., AND ROTHER, C. 2016. Cascades of regression tree fields for image restoration. *IEEE Trans. Pattern Anal. Mach. Intell. 38*, 4, 677–689.

SHEN, Y., AND BARNER, K. 2004. Fuzzy vector median-based surface smoothing. *IEEE. T. Vis. Comput. Gr. 10*, 3, 252–265.

SOLOMON, J., CRANE, K., BUTSCHER, A., AND WOJTAN, C. 2014. A general framework for bilateral and mean shift filtering. `arXiv:1405.4734 [cs.GR]`.

SUN, X., ROSIN, P. L., MARTIN, R. R., AND LANGBEIN, F. C. 2007. Fast and effective feature-preserving mesh denoising. *IEEE. T. Vis. Comput. Gr. 13*, 5, 925–938.

SUN, X., ROSIN, P. L., MARTIN, R. R., AND LANGBEIN, F. C. 2009. Noise analysis and synthesis for 3D laser depth scanners. *Graphical Models 71*, 2, 34–48.

SUN, X., WEI, Y., LIANG, S., TANG, X., AND SUN, J. 2015. Cascaded hand pose regression. In *CVPR*, 824–832.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. In *Proc. of the conference on Visualization*, 125–132.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH*, 351–358.

WANG, R., YANG, Z., LIU, L., DENG, J., AND CHEN, F. 2014. Decoupling noise and features via weighted l_1-analysis compressed sensing. *ACM Trans. Graph. 33*, 2, 18:1–18:12.

WEI, M., YU, J., PANG, W.-M., WANG, J., QIN, J., LIU, L., AND HENG, P.-A. 2015. Bi-normal filtering for mesh denoising. *IEEE. T. Vis. Comput. Gr. 21*, 1, 43–55.

XU, L., REN, J. S., YAN, Q., LIAO, R., AND JIA, J. 2015. Deep edge-aware filters. In *ICML*.

YAGOU, H., OHTAKE, Y., AND BELYAEV, A. 2002. Mesh smoothing via mean and median filtering applied to face normals. In *Geom. Model. and Proc.*, 124–131.

YAGOU, H., OHTAKE, Y., AND BELYAEV, A. 2003. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. In *Computer Graphics International*, 28–33.

ZAHARESCU, A., BOYER, E., VARANASI, K., AND HORAUD, R. 2009. Surface feature detection and description with applications to mesh matching. In *CVPR*, 373–380.

ZHANG, W., DENG, B., ZHANG, J., BOUAZIZ, S., AND LIU, L. 2015. Guided mesh normal filtering. *Comput. Graph. Forum (PG) 34*, 23–34.

ZHENG, Y., FU, H., AU, O. K.-C., AND TAI, C.-L. 2011. Bilateral normal filtering for mesh denoising. *IEEE. T. Vis. Comput. Gr. 17*, 10, 1521–1530.