Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (2015), Number 7

Guided Mesh Normal Filtering

Wangyu Zhang[1] Bailin Deng[2,3] Juyong Zhang[1†] Sofien Bouaziz[2] Ligang Liu[1]

[1]University of Science and Technology of China [2]École polytechnique fédérale de Lausanne [3]University of Hull

Abstract

The joint bilateral filter is a variant of the standard bilateral filter, where the range kernel is evaluated using a guidance signal instead of the original signal. It has been successfully applied to various image processing problems, where it provides more flexibility than the standard bilateral filter to achieve high quality results. On the other hand, its success is heavily dependent on the guidance signal, which should ideally provide a robust estimation about the features of the output signal. Such a guidance signal is not always easy to construct. In this paper, we propose a novel mesh normal filtering framework based on the joint bilateral filter, with applications in mesh denoising. Our framework is designed as a two-stage process: first, we apply joint bilateral filtering to the face normals, using a properly constructed normal field as the guidance; afterwards, the vertex positions are updated according to the filtered face normals. We compute the guidance normal on a face using a neighboring patch with the most consistent normal orientations, which provides a reliable estimation of the true normal even with a high-level of noise. The effectiveness of our approach is validated by extensive experimental results.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

The bilateral filter, introduced in [TM98], is a nonlinear filter for smoothing images while preserving sharp features such as edges. It replaces the intensity value at a pixel by a weighted average of the intensity values from neighboring pixels. The weights depend on the distance as well as the intensity difference between the pixels. In this way a pixel is only influenced by nearby pixels with similar intensity, which prevents averaging pixel values across sharp edges and preserves image features [PKTD07]. Due to its success in image processing and computational photography, many attempts have been made to adapt bilateral filtering to geometry processing such as mesh denoising and smoothing [FDCO03, JDD03, ZFAT11, SCBW14].

Extending the capability of bilateral filtering, the *joint bilateral filter* was proposed simultaneously in [PSA*04] and [ED04] for processing flash/no-flash image pairs. The basic idea is that the filtering weights can be determined using the intensity difference from another image, called the *guidance*, instead of the input image. When the guidance provides more reliable information about the image structure than the input image, joint bilateral filtering leads to better results than the classical bilateral filter. For example, this is the case for flash/no-flash image pairs, where the flash image can be used as the guidance for filtering the image without flash, since the flash image contains more information about the high-frequency features [PSA*04, ED04]. Thus by incorporating the guidance, the joint bilateral filter provides more flexibility in controlling the filtering process.

Although joint bilateral filters have been successfully applied to image processing, it is not easy to adapt them to geometry signals. The main difficulty lies in the construction of the guidance, which needs to be defined in the same domain as the input signal while providing enough information about the desired output. Unlike image signals which are defined over rectangular domains with regular sampling, geometry signals can be defined on surfaces with arbitrary topology and irregular sampling, which requires compatible surface topology and discretization between the input and the guidance signals [SCBW14]. Compared to the case of images, such guidance geometry is not easily available from measure devices, but often has to be constructed computationally.

In this paper, we tackle the problem of constructing the guidance for joint bilateral filtering of geometry signals. We

† Corresponding author: juyong@ustc.edu.cn (Juyong Zhang)

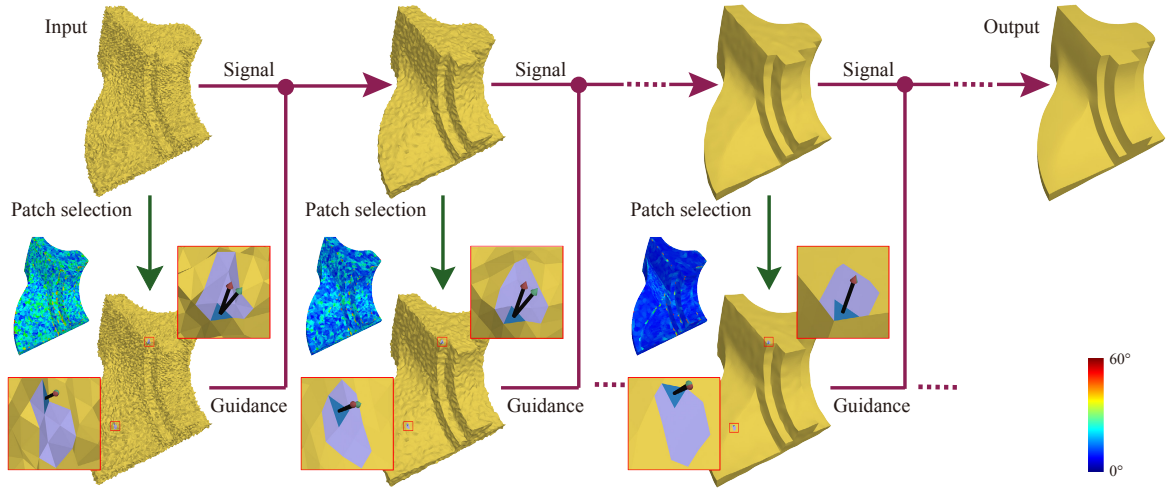

Figure 1: *The pipeline of our denoising method, using two faces close to sharp features on the ground truth mesh (shown in blue) to demonstrate our patch-based construction of the guidance normals. The ground truth normals and the guidance normals are shown in red and green, respectively. The color coding shows the angles between the ground truth and the guidance normals, which gradually decrease during the iterations.*

focus on feature-preserving denoising of triangle meshes, done via face normal filtering in a two-stage process: in the first stage, we apply joint bilateral filtering to the face normals, which are considered as a signal defined over a mesh surface; afterwards, the mesh vertices are updated to match the filtered face normals. The sensitivity of the normals with respect to the mesh shapes enables us to effectively control the features of the output mesh via normal filtering. The key to the success of this approach is a properly constructed guidance normal field for filtering the face normals, such that the joint bilateral filter averages normals that belong to a common smooth region, while preserving sharp changes of normals that indicate features. To do so, we compute the guidance normal in a patch-based approach (see Fig. 2): for a given face, we search among a set of candidate patches that contain the face, and pick the one with the most consistent normal directions; the average normal of the chosen patch is then used as the guidance normal for the face. Such guidance provides a robust estimation for the true normal in the presence of noise, enabling our denoising algorithm to handle highly noisy meshes. The effectiveness of our approach is illustrated by extensive experimental results. Moreover, we provide the source code to ensure reproducibility of our results.

1.1. Related work

Bilateral and joint bilateral filters. Due to its simplicity and feature-preserving capability, the bilateral filter [TM98] has been used in numerous applications in image processing [OCDD01, DD02, CLKL14], video processing [BM05, WOG06], and computer vision [XCS*06]. An overview of bilateral filtering and its applications can be found in [PKTD07]. As an extension of the bilateral filter, the joint bilateral filter can achieve even better results by employing a proper guidance signal. It has been successfully applied for image processing applications such as flash/no-flash image filtering [PSA*04, ED04] and image upsampling [KCLU07]. The feature-preserving capability of the standard bilateral filter has also been utilized in geometry processing such as mesh denoising [FDCO03, JDD03, ZFAT11], normal improvement for point rendering [JDZ04], surface reconstruction [DDD04], and mesh feature enhancement [Wan06]. Recently, [SCBW14] propose a framework for applying the bilateral and joint bilateral filters to signals with general domains and codomains.

Our construction of guidance normals is closely related to the joint bilateral filtering approach of [CLKL14] for extracting image structures, which also computes the guidance image via patch-based analysis of the input image. In particular, we extend the patch selection criteria from [CLKL14], to make them applicable for multi-dimensional signals defined on non-planar domains with irregular sampling.

Mesh denoising. Meshes obtained from 3D scanning devices often contain high-frequency noises. Thus mesh denoising is an important tool in geometry processing, and has been an active research topic for a long time [BPK*08]. Due to the large amount of research work in this domain, here we only review the work that are the most relevant to our method.

A major challenge in mesh denoising is to remove the noise without destroying the true features. Due to the feature-

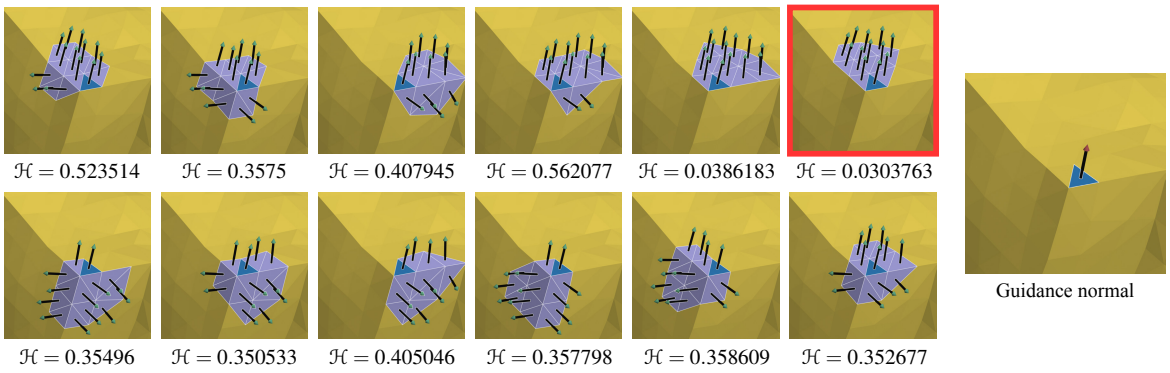

Figure 2: *Patch-based construction of a guidance normal. For a given face (in blue), the face normals from each candidate patch that contains the given face are shown, together with the consistency measure $\mathcal{H}$ computed from Equation (7). The patch with the smallest value of $\mathcal{H}$ (highlighted in red) is chosen, and its average normal is used as the guidance normal for the given face.*

preserving property of the bilateral filter, it is adopted in [FDCO03] and [JDD03] for mesh denoising, where the bilateral filter is applied to the mesh vertex positions. [ZFAT11] employ the bilateral filter to the face normals instead, followed by a vertex update according to the normals. [SCBW14] also perform mesh denoising by filtering the face normals, using a general formulation of the bilateral filter. Our denoising algorithm follows a similar approach, iterating between normal filtering and vertex update. Our method differs from [ZFAT11] and [SCBW14] as we apply a joint bilateral filter on the normals, using a proper computed guidance normal field that is robust for noisy meshes. In comparison, [ZFAT11] and [SCBW14] can be considered as applying the joint bilateral filter, with the guidance being the input face normals. Such guidance can be corrupted by the noises on the mesh and lead to unsatisfactory results.

The approach of filtering face normals and updating vertex positions has also been adopted by many previous work [YOB02, CC05, SRML07, SRML08]. The difference between these methods is in their normal filtering strategies: [YOB02] apply the mean and median filters, [CC05] select filters based on local sharpness, while [SRML07] and [SRML08] adopt trimmed quadratic weights and random walk based weights for averaging the normals respectively. Similar ideas have also been applied in point cloud consolidation [WXL*13, HWG*13, SSW15], where they first construct reliable normals for all points especially for points around the edges, and then update the point positions accordingly.

Another type of mesh denoising methods is to first classify the vertices according to their incident features such as corner, edge, and flat areas, and then apply specific denoising strategies to each type of vertices. Different vertex classification criteria have been proposed before, such as volume integral invariant [BT11], distribution of dihedral angles [CC05], quadric surface fitting [FYP10], difference between face normals [WZY12] and normal tensor voting [WYP*15].

Recently, sparsity optimization is also gaining popularity in denoising, due to the fact that sharp features are usually sparse on the ground truth mesh. [HS13] employ L_0 minimization to induce sparsity for an edge-based Laplacian operator, which is effective for preserving the sharp features; on the hand, the optimization is in favor of piecewise flat shapes, and may not be suitable for non-CAD models. A similar approach is employed in [SSW15] for denoising point clouds. Recently, [WYL*14] perform L_1 optimization to recover sharp features from noisy meshes. This method is only guaranteed to work for independent and identically distributed noises, and is computationally expensive for large meshes.

2. Guided filters for meshes

In this section, we provide an overview of our guided mesh normal filtering framework, starting from a review of the classical bilateral and joint bilateral filters.

2.1. Classical bilateral and joint bilateral filtering

Suppose we are interested in signals that are defined on domain Ω and with values in domain Γ. In its general form, the bilateral filter for a signal $I : \Omega \mapsto \Gamma$ can be written as

$$J_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \int_{\mathcal{N}(\mathbf{p})} K_s(\mathbf{p},\mathbf{q})\, K_r(I_{\mathbf{p}}, I_{\mathbf{q}})\, I_{\mathbf{q}}\, d\mathbf{q}, \quad \text{for } \mathbf{p} \in \Omega. \quad (1)$$

Here $J : \Omega \mapsto \Gamma$ is the output signal, $J_{\mathbf{p}}$ and $I_{\mathbf{p}}$ are the values of the signals J and I at point $\mathbf{p}$, respectively; $\mathcal{N}(\mathbf{p}) \subset \Omega$ is a neighborhood of $\mathbf{p}$; $K_s : \Omega \times \Omega \mapsto \mathbb{R}$ and $K_r : \Gamma \times \Gamma \mapsto \mathbb{R}$ are non-negative kernel functions for the *spatial* and *range* weights respectively, and $W_{\mathbf{p}} = \int_{\mathcal{N}(\mathbf{p})} K_s(\mathbf{p},\mathbf{q})\, K_r(I_{\mathbf{p}}, I_{\mathbf{q}})\, d\mathbf{q}$ is a normalization factor. Intuitively, $J_{\mathbf{p}}$ is a weighted average of I within the neighborhood $\mathcal{N}(\mathbf{p})$. The spatial kernel

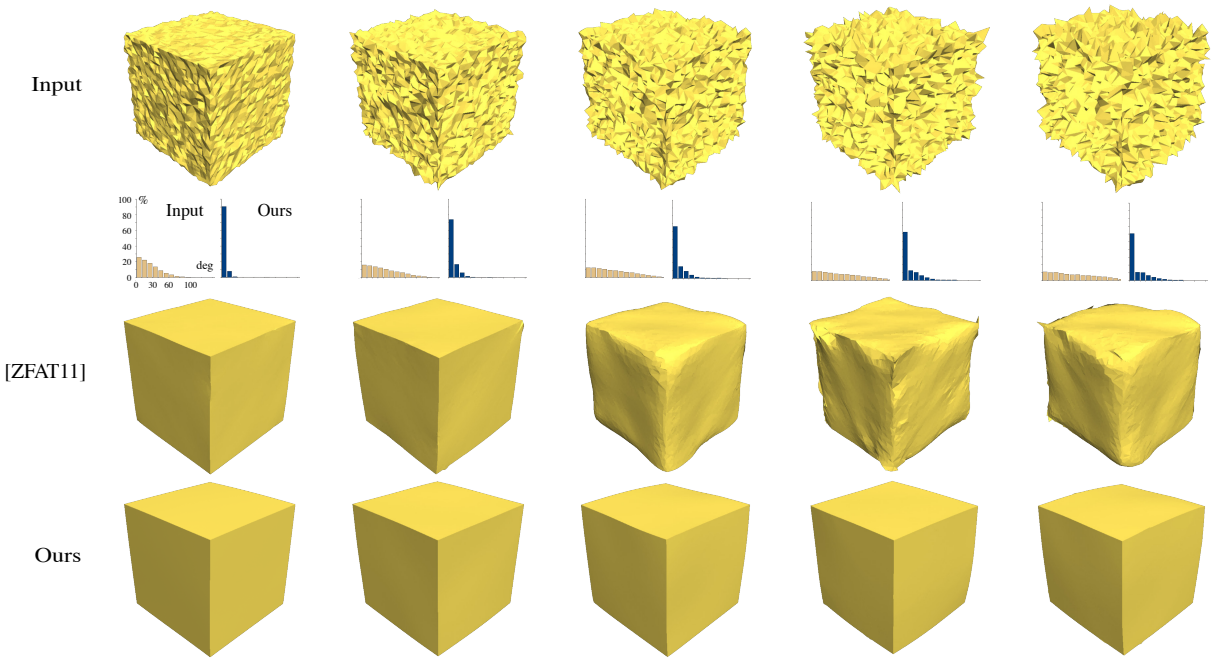

Figure 3: *Comparison between our method and the bilateral filtering scheme from [ZFAT11], which can be considered as joint bilateral normal filtering with the input face normals as guidance. The histograms show statistics of the angle error function* $\theta(\cdot)$ *in Equation (12) across all edges. The robustness of our guidance normal field helps to correctly preserve sharp features even for highly noisy meshes. The intensity* σ_E *of the Gaussian noise is from left to right 0.2, 0.4, 0.6, 0.8, and 1.0 (see Equation 14).*

K_s is monotonically decreasing with respect to the distance between $\mathbf{p}$ and $\mathbf{q}$, while the range kernel K_r is monotonically decreasing with respect to the distance between $I_\mathbf{p}$ and $I_\mathbf{q}$. A classical choice of the kernels are the Gaussian functions evaluated from the Euclidean distance [TM98]:

$$\begin{aligned} K_s(\mathbf{p},\mathbf{q}) &= \exp(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma_s^2}), \\ K_r(I_\mathbf{p},I_\mathbf{q}) &= \exp(-\frac{\|I_\mathbf{p}-I_\mathbf{q}\|^2}{2\sigma_r^2}), \end{aligned} \tag{2}$$

where σ_s, σ_r are variance parameters. As the kernel values fall off quickly with increasing distance values, the filtered signal $J_\mathbf{p}$ is influenced by the input signal value $I_\mathbf{q}$ only if points $\mathbf{p}$ and $\mathbf{q}$ are close to each other in terms both their spatial distance and their input signal value difference. As a result, unlike the Gaussian filter which smooths the whole signal, the bilateral filter is able to perform smoothing while preserving high-frequency features such as image edges [PKTD07].

The effectiveness of the bilateral filter relies on the range kernel that relates the averaging weights to the signal intensity difference. In the classical scheme (1), the weight functions are evaluated from the input signal I. In doing so, it makes an implicit assumption that for nearby points $\mathbf{p}$ and $\mathbf{q}$, the proximity between the input signal values $I_\mathbf{p}$ and $I_\mathbf{q}$ provides reliable prediction for the proximity between the desired output signal values $J_\mathbf{p}$ and $J_\mathbf{q}$. However, this assumption is not always valid. For example, when recovering a signal from a noisy measurement, the noise in the input signal may render it unreliable for predicting the features of the output signal, which can lead to erroneous denoising results using the scheme (1). This issue can be resolved using the joint bilateral filter [ED04, PSA*04], which evaluates the range kernel using a guidance signal $G : \Omega \mapsto \Gamma$ such that

$$J_\mathbf{p} = \frac{\int_{\mathcal{N}(\mathbf{p})} K_s(\mathbf{p},\mathbf{q})\, K_r(G_\mathbf{p},G_\mathbf{q})\, I_\mathbf{q}\, d\mathbf{q}}{\int_{\mathcal{N}(\mathbf{p})} K_s(\mathbf{p},\mathbf{q})\, K_r(G_\mathbf{p},G_\mathbf{q})\, d\mathbf{q}}. \tag{3}$$

The idea is that the guidance G can provide more reliable information about the structure of the desired output, thus producing more suitable range weights that direct the filter towards desirable results. On the other hand, the effectiveness of the joint bilateral filter depends on how well the guidance indicates the features of the desired output. While some previous work such as [ED04, PSA*04] use the same measurement device in different settings to produce the input signal and the guidance, for many applications it is not straightforward to find a suitable guidance. This is exactly the problem we tackle in this paper, within the context of mesh denoising.

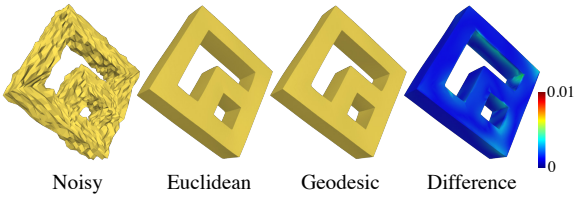

Figure 4: *Denoising results using the Euclidean distance and the geodesic distance for the joint bilateral filter, respectively. The color coding shows the closest distance from the vertices of the Euclidean result mesh to the surface of the geodesic result mesh, normalized by the bounding box diagonal length of the input mesh.*

2.2. Filtering mesh geometry

Given a noisy triangle mesh, our goal is to smooth the surface while preserving the sharp features. We achieve this in a two-stage process similar to [ZFAT11]:

- First, we apply a joint bilateral filter to the face normals, by considering them as a signal defined over the mesh surface.
- Afterwards, the vertex positions are updated according to the filtered face normals.

One benefit of filtering the normals instead of the vertices is that the normals provide convenient descriptors for geometric features of a mesh. For example, a sharp edge is indicated by a large difference between the normals of its two incident faces. To achieve desirable results, it is crucial to employ a proper guidance for the joint bilateral filtering of face normals. In the following, we provide an overview of our guided normal filtering framework. Details of our method for constructing the guidance can be found in Section 3.

Filtering face normals. For a face f_i of an orientable triangle mesh, its outward unit normal can be computed as $\mathbf{n}_i = \frac{(\mathbf{v}_{i_2} - \mathbf{v}_{i_1}) \times (\mathbf{v}_{i_3} - \mathbf{v}_{i_1})}{\|(\mathbf{v}_{i_2} - \mathbf{v}_{i_1}) \times (\mathbf{v}_{i_3} - \mathbf{v}_{i_1})\|}$, where $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \mathbf{v}_{i_3}$ are the positions of its vertices in a fixed orientation. We consider $\mathbf{n}_i$ as a signal associated with the face centroid $\mathbf{c}_i$. To filter the face normals,we first find a unit vector $\mathbf{g}_i$ for each face f_i as the guidance. Then a filtered normal $\bar{\mathbf{n}}_i$ for face f_i is computed from joint bilateral filtering:

$$\bar{\mathbf{n}}_i = \frac{1}{W_i} \sum_{f_j \in \mathcal{N}_i} A_j \, K_s(\mathbf{c}_i, \mathbf{c}_j) \, K_r(\mathbf{g}_i, \mathbf{g}_j) \, \mathbf{n}_j, \qquad (4)$$

where $\mathcal{N}_i$ is the set of faces in a neighborhood of f_i; A_j is the area of f_j; K_s, K_r are the Gaussian kernels as defined in (2); and the normalization factor $W_i = \|\sum_{f_j \in \mathcal{N}_i} A_j \, K_s(\mathbf{c}_i, \mathbf{c}_j) \, K_r(\mathbf{g}_i, \mathbf{g}_j) \, \mathbf{n}_j\|$ ensures that $\bar{\mathbf{n}}_i$ is a unit vector. The set $\mathcal{N}_i$ consists of the f_i and a set of surrounding faces, and can be defined in different ways:

- A simple choice is a *topological* neighborhood, consisting of all faces that share at least one vertex with f_i:

$$\mathcal{N}_i = \{f_k \mid v_{i_1} \in f_k \text{ or } v_{i_2} \in f_k \text{ or } v_{i_3} \in f_k\}, \qquad (5)$$

  where $v_{i_1}, v_{i_2}, v_{i_3}$ are the vertices of f_i. This definition is similar to [ZFAT11].
- Alternatively, we can choose a *geometrical* neighborhood, defined as the maximal set of faces that contains f_i and satisfies the following conditions:
  1. For each face in $\mathcal{N}_i$, its centroid is within distance r from the centroid of f_i, where r is a parameter specified by the user: $\forall f_k \in \mathcal{N}_i, \|\mathbf{c}_k - \mathbf{c}_i\| \le r$;
  2. The faces in $\mathcal{N}_i$ form a single connected component via shared vertices between neighboring faces.

  Such a neighborhood can be computed using breadth-first search starting from f_i.

Algorithm 1 Guided mesh normal filtering framework.

Input: Initial mesh M_{in}, number of iterations k_{iter}.
Output: Filtered mesh M_{out}.
1: $M^{(0)} = M_{\text{in}}$;
2: **for** $s = 1$ to k_{iter} **do**
3: Compute face normals $\{\mathbf{n}_i\}$ of mesh $M^{(s-1)}$;
4: Construct guidance normals $\{\mathbf{g}_i\}$;
5: Compute filtered normals $\{\bar{\mathbf{n}}_i\}$ according to (4);
6: Compute updated mesh $M^{(s)}$ according to $\{\bar{\mathbf{n}}_i\}$;
7: **end for**
8: $M_{\text{out}} = M^{(k_{\text{iter}})}$.

For meshes with highly non-uniform sampling, the geometrical neighborhood can provide better results, since it is able to include all neighboring faces with similar spatial weights.

Remark. Using the Gaussian kernels in (2), the weights for $\mathbf{n}_j$ in our filtering scheme depends on the proximity between the centroids $\mathbf{c}_i, \mathbf{c}_j$ and the proximity between the guidance normals $\mathbf{g}_i, \mathbf{g}_j$, both measured using the Euclidean distance. Since $\mathbf{c}_i, \mathbf{c}_j$ lie on the mesh surface, and $\mathbf{g}_i, \mathbf{g}_j$ lie on the unit sphere, we can also evaluate the kernels K_s, K_r using the geodesic distance instead of the Euclidean distance. In our experiments, using the geodesic distance does not make a significant difference to the final results (see Fig. 4), while the evaluation of geodesic distance can be much more involved. Thus in this paper, the kernels are always evaluated from the Euclidean distance, unless stated otherwise.

Updating vertices. After filtering the face normals, the vertex positions need to be updated to match new normal directions $\{\bar{\mathbf{n}}_i\}$. We adopt the iterative scheme from [SRML07] for the vertex update. Specifically, for a face f_i, we compute its updated vertex positions $\bar{\mathbf{v}}_{i_1}, \bar{\mathbf{v}}_{i_2}, \bar{\mathbf{v}}_{i_3}$ via the iteration

$$\bar{\mathbf{v}}_i^{(t+1)} = \bar{\mathbf{v}}_i^{(t)} + \frac{1}{|\mathcal{F}_i|} \sum_{j \in \mathcal{F}_i} \bar{\mathbf{n}}_j \left[\bar{\mathbf{n}}_j \cdot \left(\bar{\mathbf{c}}_j^{(t)} - \bar{\mathbf{v}}_i^{(t)} \right) \right], \qquad (6)$$

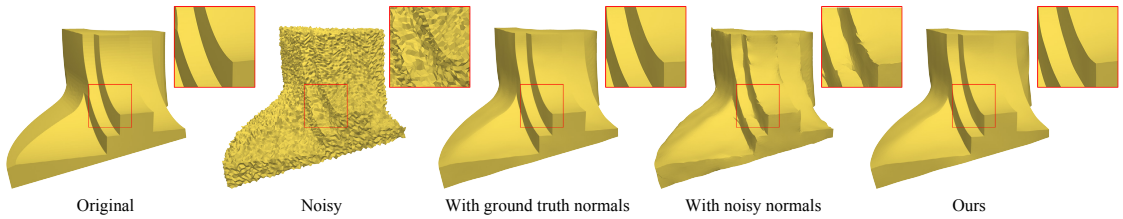

Figure 5: *Comparison between joint bilateral normal filtering for denoising, using different guidance normal fields. Our patch-based guidance normals produce similar results as using the ground truth normals for guidance.*

where $\overline{\mathbf{v}}_i^{(t)}$ is the value of $\overline{\mathbf{v}}_i$ in the t-th iteration, $\mathcal{F}_i$ is the index set of the incident faces for $\overline{\mathbf{v}}_i$, and $\overline{\mathbf{c}}_j^{(t)} = (\overline{\mathbf{v}}_{j_1}^{(t)} + \overline{\mathbf{v}}_{j_2}^{(t)} + \overline{\mathbf{v}}_{j_3}^{(t)})/3$. This scheme is actually a gradient descent process for the ℓ_2 error of the compatibility conditions $\overline{\mathbf{n}}_i \cdot (\overline{\mathbf{v}}_{i_k} - \overline{\mathbf{c}}_i) = 0$ $(k = 1,2,3)$ across all faces. In our experiments, 10 to 20 iterations are sufficient for the mesh to approximately satisfy the above conditions while being close to the initial shape. We iterate between the face normal filtering step and the vertex update step, until a desired result is obtained. A summary of our framework is given in Algorithm 1. In the next section, we present a method for computing guidance normals that are suitable for mesh denoising.

3. Guidance normals for mesh denoising

Many meshes used in computer graphics are piecewise smooth, i.e., their surfaces contain smooth regions separated by sharp edges. When filtering the normal on a face incident with a sharp edge, we should avoid combining it with the face normal across the edge, since a sharp edge is indicated by a large difference between its two incident face normals. When no noise is present, the input face normals $\{\mathbf{n}_i\}$ can provide good guidance for joint bilateral filtering: across a sharp edge, the large difference between the two face normals results in a small value of the range kernel, which inhibits the two normals from influencing each other. However, with more and more noise on the mesh, the noisy normals will become less and less reliable for indicating the features of the ground truth shape, eventually leading to erroneous results if the input normal field is employed as the guidance (Fig. 3). It is thus necessary to compute a guidance normal field that provides robust indication of features in the presence of noise.

Our method is based on the following observation: a triangle mesh can be decomposed into many small patches, each consisting of multiple faces with similar normal directions. Thus for each face f_i, we can search for such a small patch that contains f_i, and choose its average normal as the guidance normal at f_i. Specifically, for each face f_k, we define a patch $\mathcal{P}_k$ as the union of f_k with all surrounding faces that share vertices with it. To compute the guidance normal at a face f_i, we search among all patches that contain f_i, and pick the one with the most consistent normal directions. More specifically, the set of candidate patches is $\mathcal{C}(f_i) = \{\mathcal{P}_k \mid f_i \in \mathcal{P}_k\}$. For each candidate patch $\mathcal{P} \in \mathcal{C}(f_i)$, we measure the consistency of its normals using a function

$$\mathcal{H}(\mathcal{P}) = \Phi(\mathcal{P}) \cdot \mathcal{R}(\mathcal{P}). \tag{7}$$

Here $\Phi(\mathcal{P})$ measures the maximum difference between two face normals from the patch:

$$\Phi(\mathcal{P}) = \max_{f_j, f_k \in \mathcal{P}} \|\mathbf{n}_j - \mathbf{n}_k\|. \tag{8}$$

And $\mathcal{R}(\mathcal{P})$ is a relative measure of edge saliency in the patch:

$$\mathcal{R}(\mathcal{P}) = \frac{\max_{e_j \in E_\mathcal{P}} \varphi(e_j)}{\varepsilon + \sum_{e_j \in E_\mathcal{P}} \varphi(e_j)}, \tag{9}$$

where $E_\mathcal{P}$ is the set of mesh edges with both incident faces contained in patch $\mathcal{P}$, $\varphi(e_j)$ measures the saliency of an edge e_j using the difference between the normals of the two incident faces f_{j_1}, f_{j_2}:

$$\varphi(e_j) = \|\mathbf{n}_{j_1} - \mathbf{n}_{j_2}\|, \tag{10}$$

and ε is a small positive value for avoiding division by zero. Note that a small value of $\Phi(\mathcal{P})$ indicates similar face normals within the patch, while a small value of $\mathcal{R}(\mathcal{P})$ indicates similar saliency among all interior edges of the patch. Thus the consistency function $\mathcal{H}(\mathcal{P})$ in Equation (7) is in favor of patches with a small range of normal directions, and without edges that are much 'sharper' than other edges. Finally, among all candidate patches for f_i, we pick the one $\mathcal{P}^*$ with the smallest consistency function, and compute its area-weighted average normal as the guidance $\mathbf{g}_i$ for face f_i:

$$\mathbf{g}_i = \frac{\sum_{f_j \in \mathcal{P}^*} A_j \mathbf{n}_j}{\|\sum_{f_j \in \mathcal{P}^*} A_j \mathbf{n}_j\|}, \tag{11}$$

where A_j is the area of face f_j. This construction process is illustrated in Fig. 2. Algorithm 2 provides the pseudocode for efficient computation of the guidance normals for all faces of a mesh. The pipeline of our denoising method is illustrated in Fig. 1. The zoom-in parts show the chosen patches and the guidance normals, for two faces that are adjacent to sharp features on the ground truth mesh. The color coding shows that across the whole mesh, the guidance normals get gradually closer to the ground truth normals during the iterations, which helps to recover the original shape. Fig. 3 further shows the effectiveness of our guidance construction method in the

Algorithm 2 Computation of guidance normals on a mesh.

Input: A mesh with faces $\mathcal{F}$.
Output: A guidance normal $\mathbf{g}_i$ for each face $f_i \in \mathcal{F}$.

For each face $f_i \in \mathcal{F}$, initialize the minimum consistency function value $\mathcal{H}^i_{\min} = +\infty$;
for each face $f_k \in \mathcal{F}$ **do**
  Construct a patch
    $$\mathcal{P}_k = \{f_j \mid f_j \text{ shares a vertex with } f_k\};$$
  Compute the consistency function $\mathcal{H}(\mathcal{P}_k)$ and the average normal $\widehat{\mathbf{n}}_k$ for patch $\mathcal{P}_k$, according to Equations (7) and (11), respectively;
  for each face $f_j \in \mathcal{P}_k$ **do**
    if $\mathcal{H}(\mathcal{P}_k) < \mathcal{H}^j_{\min}$ **then**
      $\mathcal{H}^j_{\min} = \mathcal{H}(\mathcal{P}_k)$, $\mathbf{g}_j = \widehat{\mathbf{n}}_k$;
    end if
  end for
end for

presence of noise. For a given guidance normal field $\{\eth_i\}$, we evaluate its robustness at an edge e_j by comparing the angle between the guidance normals $\mathbf{g}_{j_1}, \mathbf{g}_{j_2}$ on its incident faces against the angle between the corresponding ground truth face normals $\mathbf{n}_{j_1}, \mathbf{n}_{j_2}$, using a function

$$\theta(e_j) = \left| \angle(\mathbf{g}_{j_1}, \mathbf{g}_{j_2}) - \angle(\mathbf{n}_{j_1}, \mathbf{n}_{j_2}) \right|. \tag{12}$$

Smaller values of the function θ implies better consistency between the guidance and the ground truth normals for indicating feature edges. Fig. 3 provides histograms of the function θ among all edges, for the same ground truth mesh with different levels of noises, and using the input noisy normals and our patch-based normals as the guidance respectively. The histograms show that our patch-based normals provide more robust indication of feature edges even for highly noisy models. This leads to more accurate results using our guidance normals, as shown in Figs. 3 and 5. More results for our denoising algorithm can be found in Section 4.

Remark. Our guidance construction method is similar to the patch-shift approach proposed in [CLKL14] for separating textures from structures in images using joint bilateral filtering. At each pixel, they select a neighboring square patch, and use the average intensity of the patch as the guidance intensity for the current pixel. One of their patch selection criteria is the so-called modified relative total variation (mRTV). Our consistency function $\mathcal{H}(\cdot)$ can be considered as a generalization of mRTV, from scalar signal defined on regular grids, to vector-valued signals defined on surfaces with irregular samples. In particular, our angle range function $\Phi(\cdot)$ generalizes the image tonal range function in mRTV, while our saliency function $\mathcal{R}(\cdot)$ corresponds to the image gradient norm. Indeed, there is a natural analogy between mesh denoising and texture extraction: the underlying mesh surface can be considered as structure, while the high-frequency noise can be considered as texture. Interestingly, the consistency

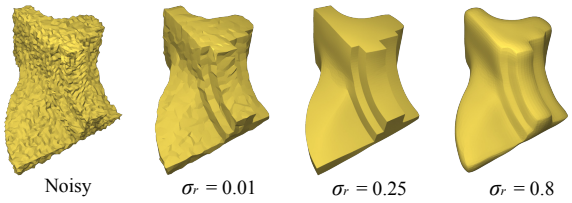

Figure 6: *Denoising results using different values of σ_r, with other parameters fixed.*

function $\mathcal{R}(\cdot)$ can also be interpreted using sparsity. Ignoring the small constant ε, we can see that $\mathcal{R}(\mathcal{P})$ is the reciprocal of the ℓ_1-norm of a vector

$$\Psi(\mathcal{P}) = \left(\frac{\varphi(e_1)}{\varphi_{\max}}, \frac{\varphi(e_2)}{\varphi_{\max}}, \ldots, \frac{\varphi(e_m)}{\varphi_{\max}} \right), \tag{13}$$

where $\varphi(\cdot)$ is the edge saliency function in Equation (10), $e_1, e_2, \ldots, e_m$ are the interior edges with respect to the patch $\mathcal{P}$, and $\varphi_{\max} = \max_{e_j \in E_{\mathcal{P}}} \varphi(e_j)$. Namely, the components of $\Psi(\mathcal{P})$ are the saliency values for the interior edges, normalized by the maximum edge saliency across the patch $\mathcal{P}$. It is well known that the ℓ_1-norm for a vector is related to its sparsity, i.e., the property that many of its components are close to zero [BJMO12]. Thus a small ℓ_1-norm of $\Psi(\mathcal{P})$ indicates that a small number of edges are much more salient than the others. The function $\mathcal{R}(\mathcal{P})$, being the reciprocal of the reciprocal of the ℓ_1-norm for $\Psi(\mathcal{P})$, inhibits such patches from being chosen for the guidance normal computation.

4. Implementation and results

4.1. Choice of parameters

Our method involves a set of parameters: the number of normal filtering iterations k_{iter}, the number of iterations v_{iter} for vertex update (6) according to a given normal field, the radius parameter r for finding a geometrical neighborhood, and the variance parameters σ_s and σ_r for the spatial and the range kernels, respectively. We observe in our experiments that $k_{\text{iter}} \leq 75$ and $v_{\text{iter}} \leq 20$ are enough for achieving nice results while avoiding unnecessary iterations. We always set the spatial variance σ_s as the average distance between neighboring face centroids across the whole mesh, as suggested in [ZFAT11]. The range variance σ_r controls the denoising effect and the smoothness of the final results (Fig. 6): if σ_r is too small, some noise may not be removed; if σ_r is too large, the results may be over-smoothed and lose some features. We observe that $\sigma_r \in [0.2, 0.6]$ provides a good compromise between noise removal and feature preservation. For all results in this paper, we employ geometrical neighborhoods for applying the filter unless stated otherwise. To achieve desirable results, we choose $r \in [2 \times \sigma_s, 3 \times \sigma_s]$ to ensure the geometrical neighborhood includes all faces with large enough spatial kernel values. Setting r to a larger value does

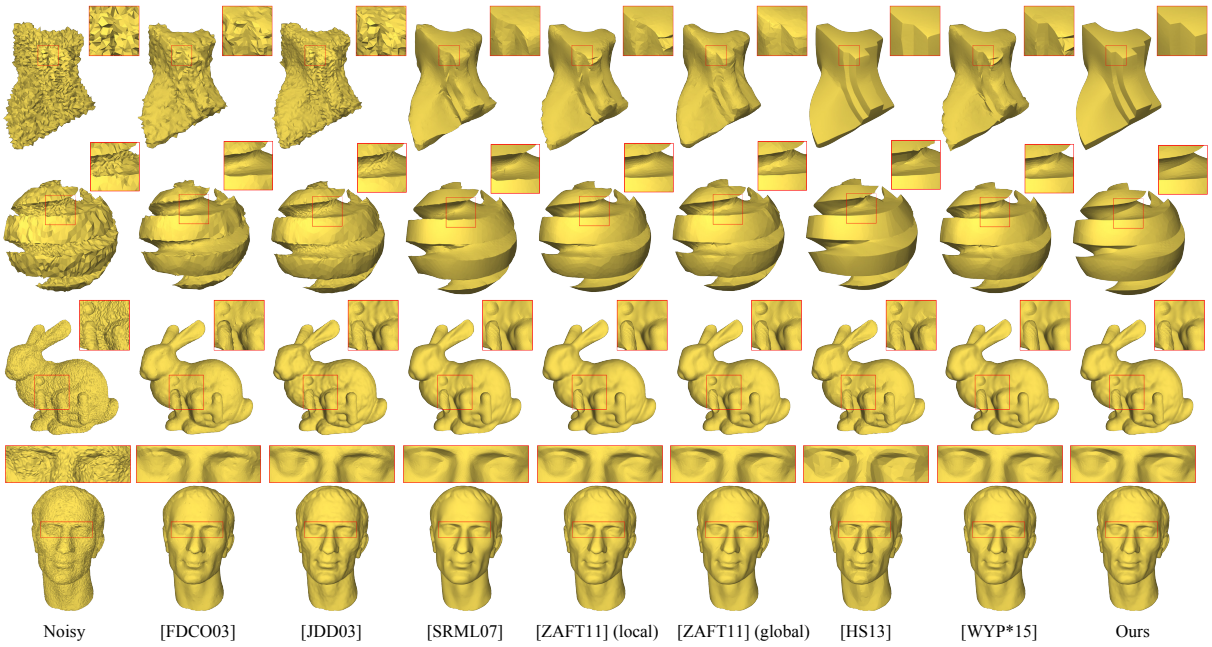

Figure 7: *Comparison of denoising algorithms for meshes with additive Gaussian noise. The intensity σ_E of the noise is from top to bottom 0.7, 0.3, 0.2 and 0.2.*

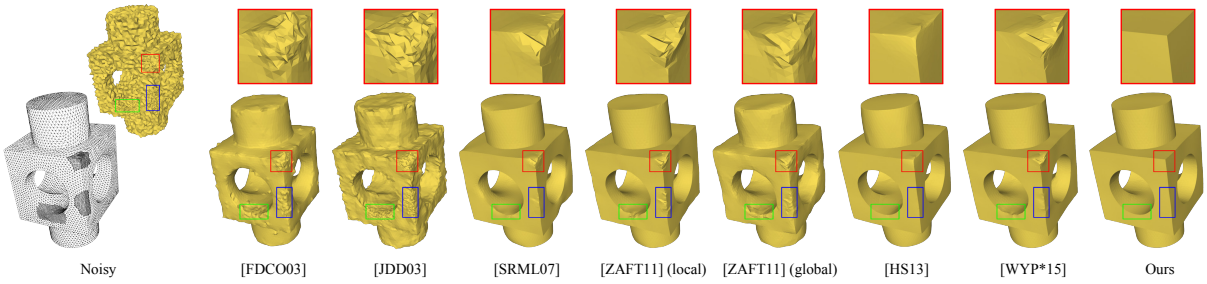

Figure 8: *Denoising a mesh with non-uniform sampling. For the resulting meshes, regions with denser triangulation are highlighted with colored rectangles. The intensity σ_E of the Gaussian noise applied to this model is 0.4.*

not make a significant difference but increases the computational cost, because the additionally included faces have very small spatial kernel values and make almost no contribution to the filtering.

4.2. Results and comparisons

In the following, we show denoising results computed using our framework, and compare them against the results using other methods. We provide the source code of a program that implements our method as well as other denoising algorithms [FDCO03, JDD03, SRML07, ZFAT11, HS13], on `https://github.com/bldeng/GuidedDenoising`. Note that [ZFAT11] propose two denoising schemes: a local scheme that directly applies bilateral filtering to the face normals, and a global scheme that filters the face normals by solving a global optimization problem. Both schemes are compared in our examples. Note that each method involves a set of parameters that need to be set by the user, and the best parameters often depend on the input model. For a fair comparison, for each method we enumerate a dense set of samples in its parameter space, and choose the best result from the sample parameters. The parameter values we choose can be found in the supplementary material. For models whose ground truth shapes are known, we evaluate the quality of a denoising result using three error metrics:

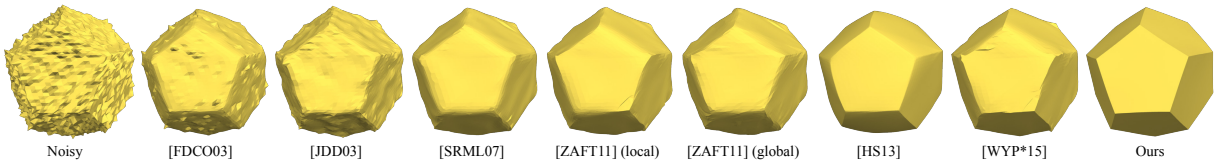

Figure 9: *Comparison of denoising algorithms on a mesh with impulsive noise.*

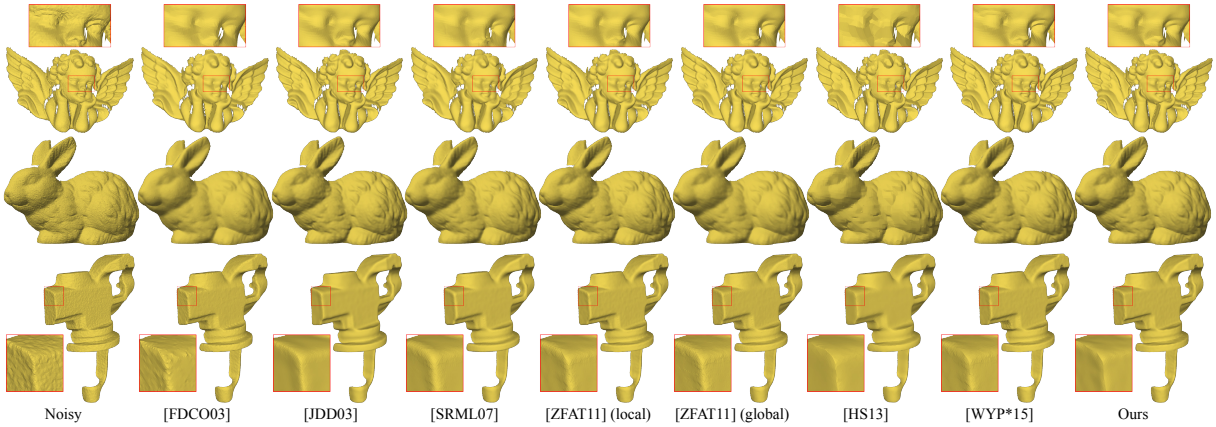

Figure 10: *Comparison of denoising algorithms applied on real world 3D objects.*

- δ: the angle between the ground truth face normals and the resulting face normals, averaged over all faces.
- $\mathcal{D}_{\text{mean}}$: the average distance from the resulting mesh vertices to the ground truth mesh surface, weighted by the associated area for the vertices on the ground truth mesh.
- $\mathcal{D}_{\text{max}}$: the maximum distance from the resulting mesh vertices to the ground truth mesh surface.

In Figures 7, 8, 9, and 10 we compare our denoising algorithm against other methods based on normal filtering [FDCO03, JDD03, SRML07, ZFAT11, WYP*15] and L_0 minimization [HS13]. In Figs. 7 and 8, the input noisy model is generated by adding Gaussian noise to the vertices of a ground truth mesh along the vertex normals. More comparison for such models can be found in the supplementary material. In this paper, the intensity of the noise is described using a relative variance parameter

$$\sigma_E = \frac{\sigma}{E_{\text{mean}}}, \tag{14}$$

where σ is the variance of the Gaussian function, and E_{mean} is the average edge length of the ground truth mesh. For most of the models, our method achieves better results according to the error metrics shown in Table 1, with a notable exception of the sphere model. Although our method successfully recovers the sharp features of the ground truth mesh, it also induces sharp features in the concave regions that are smooth on the ground truth, which causes higher error metric values.

Fig. 8 also shows the benefit of using geometrical neighborhoods for the filtering normals in our framework. Here some regions of the input mesh are of higher triangulation density. By applying the filter in geometrical neighborhoods, our method ensures successfully recover the shape regardless of the triangulation density. In comparison, the other normal filtering methods rely on 1-ring neighbors for filtering face normals, which leads to undesirable results in the regions with denser triangles.

Our method can also work for meshes with non-Gaussian

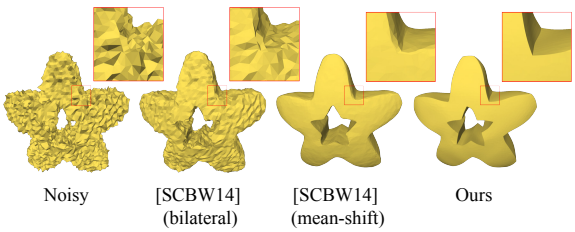

Figure 11: *Comparison between our method and [SCBW14].*

Model	Error	[FDCO03]	[JDD03]	[SRML07]	[ZFAT11](l)	[ZFAT11](g)	[HS13]	[WYP*15]	Ours
Fandisk Fig 7	δ	22.49	27.55	13.19	15.05	14.25	6.22	17.27	**6.05**
	$\mathcal{D}_{mean}$	$2.91\cdot10^{-2}$	$3.03\cdot10^{-2}$	$1.76\cdot10^{-2}$	$2.09\cdot10^{-2}$	$1.93\cdot10^{-2}$	$1.61\cdot10^{-2}$	$1.96\cdot10^{-2}$	$\mathbf{1.02\cdot10^{-3}}$
	$\mathcal{D}_{max}$	$4.74\cdot10^{-1}$	$4.22\cdot10^{-1}$	$3.65\cdot10^{-1}$	$3.24\cdot10^{-1}$	$3.84\cdot10^{-1}$	$2.83\cdot10^{-1}$	$4.05\cdot10^{-1}$	$\mathbf{2.57\cdot10^{-1}}$
Sphere Fig 7	δ	12.58	17.36	11.89	**6.70**	9.70	12.96	8.74	11.57
	$\mathcal{D}_{mean}$	$1.08\cdot10^{-1}$	$7.01\cdot10^{-2}$	$7.36\cdot10^{-2}$	$\mathbf{3.58\cdot10^{-2}}$	$3.92\cdot10^{-2}$	$1.05\cdot10^{-1}$	$3.69\cdot10^{-2}$	$6.62\cdot10^{-2}$
	$\mathcal{D}_{max}$	$8.38\cdot10^{-1}$	$6.68\cdot10^{-1}$	$6.86\cdot10^{-1}$	$\mathbf{4.82\cdot10^{-1}}$	$5.58\cdot10^{-1}$	$5.72\cdot10^{-1}$	$5.72\cdot10^{-1}$	$5.97\cdot10^{-1}$
Bunny Fig 7	δ	6.93	5.81	5.89	5.67	5.57	6.33	5.95	**5.28**
	$\mathcal{D}_{mean}$	$1.20\cdot10^{-3}$	$7.60\cdot10^{-4}$	$7.69\cdot10^{-4}$	$7.24\cdot10^{-4}$	$6.78\cdot10^{-4}$	$7.60\cdot10^{-4}$	$6.93\cdot10^{-4}$	$\mathbf{6.00\cdot10^{-4}}$
	$\mathcal{D}_{max}$	$1.47\cdot10^{-1}$	$7.01\cdot10^{-2}$	$8.31\cdot10^{-2}$	$8.08\cdot10^{-2}$	$8.41\cdot10^{-2}$	$\mathbf{6.96\cdot10^{-2}}$	$8.42\cdot10^{-2}$	$7.89\cdot10^{-2}$
Julius Fig 7	δ	7.70	7.63	7.01	6.21	6.11	7.98	6.15	**6.10**
	$\mathcal{D}_{mean}$	$8.44\cdot10^{-4}$	$6.20\cdot10^{-4}$	$5.02\cdot10^{-4}$	$4.39\cdot10^{-4}$	$4.48\cdot10^{-4}$	$6.91\cdot10^{-4}$	$\mathbf{4.26\cdot10^{-4}}$	$4.44\cdot10^{-4}$
	$\mathcal{D}_{max}$	$7.78\cdot10^{-2}$	$6.81\cdot10^{-2}$	$6.76\cdot10^{-2}$	$5.45\cdot10^{-2}$	$5.71\cdot10^{-2}$	$9.32\cdot10^{-2}$	$5.63\cdot10^{-2}$	$\mathbf{5.25\cdot10^{-2}}$
Block Fig 8	δ	12.71	13.85	6.39	5.31	8.13	4.97	5.25	**3.60**
	$\mathcal{D}_{mean}$	$1.43\cdot10^{-1}$	$1.06\cdot10^{-1}$	$7.33\cdot10^{-2}$	$5.08\cdot10^{-2}$	$7.29\cdot10^{-2}$	$9.83\cdot10^{-2}$	$4.81\cdot10^{-2}$	$\mathbf{4.31\cdot10^{-2}}$
	$\mathcal{D}_{max}$	$9.16\cdot10^{-1}$	$7.77\cdot10^{-1}$	$7.84\cdot10^{-1}$	$7.17\cdot10^{-1}$	$6.74\cdot10^{-1}$	$6.20\cdot10^{-1}$	$7.32\cdot10^{-1}$	$\mathbf{5.31\cdot10^{-4}}$
Twelve Fig 9	δ	11.72	11.09	7.45	7.37	7.27	8.46	8.23	**3.31**
	$\mathcal{D}_{mean}$	$1.41\cdot10^{-2}$	$1.34\cdot10^{-2}$	$8.99\cdot10^{-3}$	$9.34\cdot10^{-3}$	$9.26\cdot10^{-3}$	$1.55\cdot10^{-2}$	$8.22\cdot10^{-3}$	$\mathbf{4.99\cdot10^{-3}}$
	$\mathcal{D}_{max}$	$3.68\cdot10^{-1}$	$3.27\cdot10^{-1}$	$2.39\cdot10^{-1}$	$2.65\cdot10^{-1}$	$2.37\cdot10^{-1}$	$3.19\cdot10^{-1}$	$3.02\cdot10^{-1}$	$\mathbf{1.99\cdot10^{-1}}$

Table 1: *Error metrics for different methods. For each model, the best error metric value is highlighted in bold.*

additive noise, as well as real scanned models. This is illustrated in Fig. 9 (for meshes with additive impulsive noise), and in Fig. 10, respectively.

In Figs. 11 and 12 , our method is compared against [SCBW14] and [WYL*14], using input models and denoising results provided by the authors of the respective papers. In both figures, our method successfully smooths the meshes while retaining the features.

Table 2 provides the timing of our method for the shown examples, on a PC with an Intel i7-3770K CPU. Even with our naïve implementation, our method is able to denoise a large mesh efficiently.

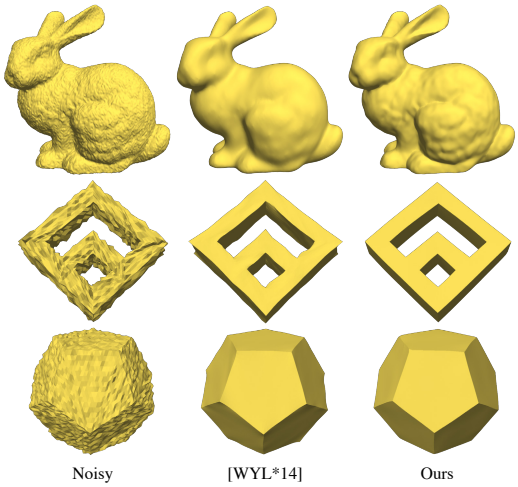

Figure 12: *Comparison between our method and [WYL*14].*

4.3. Limitations and discussion

Although the effectiveness of our method has been verified by the extensive experiments, it still has some limitations: First, for models with extremely irregular sampling, our method can produce undesirable results (Fig. 13), because neither the topology neighborhood nor the geometrical neighborhood can properly account for the contribution from nearby faces in the normal filtering. Second, for very noisy models, the sharp features recovered with our method might not be as smooth or straight as expected, although already better than other methods (Fig. 3); this is partly because the vertex update is driven by the filtered normals instead of the target feature lines. Moreover, our vertex update step only aims at

Model	Vertices	Faces	Time (ms)	k_{iter}
Fandisk (Fig 7)	6475	12946	76	50
Sphere (Fig 7)	10443	20882	162	30
Bunny (Fig 7)	34834	69451	698	4
Julius (Fig 7)	36201	71912	452	5
Block (Fig 8)	8771	17550	104	40
Twelve (Fig 9)	4610	9216	42	75
Angel (Fig 10)	24566	48090	783	3
Rabbit (Fig 10)	37394	73679	699	4
Iron (Fig 10)	85574	168285	1571	20

Table 2: *Time per iteration and number of iterations used for the different results. The timings have been measured on an Intel i7-3770K.*

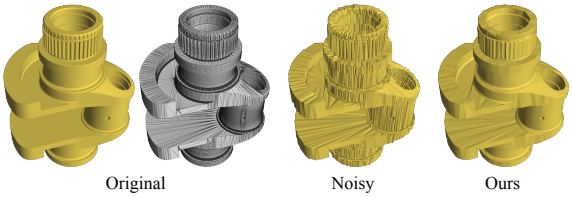

Figure 13: *Our method may fail on meshes with extremely irregular sampling.*

the orthogonality between the filtered normals and the new edges without considering the face orientation, which can introduce flipped triangles in some cases. To improve the results, we can potentially incorporate other criteria into the filtering process, such as shape priors of the feature lines and the quality of triangulation. Third, our method can be sensitive to the resolution of the models, as shown in Fig. 14: here we employ mesh decimation to create four ground truth meshes with the same underlying shape but different resolutions, and add the same Gaussian noise on the four models. Our method decreases the error metrics for all models initially, but increases the error metrics for models of lower resolutions during further iterations. This is because for such models, the normals are filtered using larger neighborhoods and the guidance normals are computed on larger patches, which can lead to over-smoothing. In the future, we would like to investigate how to make our method adaptive to mesh resolutions. Finally, similar to the local scheme of [ZFAT11], our method does not guarantee convergence (see Fig. 14), and requires parameter tuning to achieve nice results. The main issue is that we apply local operations to filter the normals, while all face normals are indeed globally coupled by certain integrability conditions. One potential solution is to apply a global filtering scheme like [ZFAT11].

The success of our method lies on the construction of guidance normals that provide reliable estimation for the ground truth. In Fig. 15, we modify our denoising method, by performing the vertex update based on the guidance normals instead of the filtered normals. Interestingly, such a variant is able to remove high frequency noises, although it also leads to undesirable flat regions due to the discrepancy between the guidance and the ground truth normals. In comparison, the joint bilateral filter from our method makes the final results smoother while retaining the sharp features. This observation sheds lights on a rigorous proof for the effectiveness of our method, which will require further investigation.

5. Conclusion

In this paper, we propose a guided normal filtering framework for denoising triangular meshes. Our framework applies a joint bilateral filter to the face normals, followed by an update

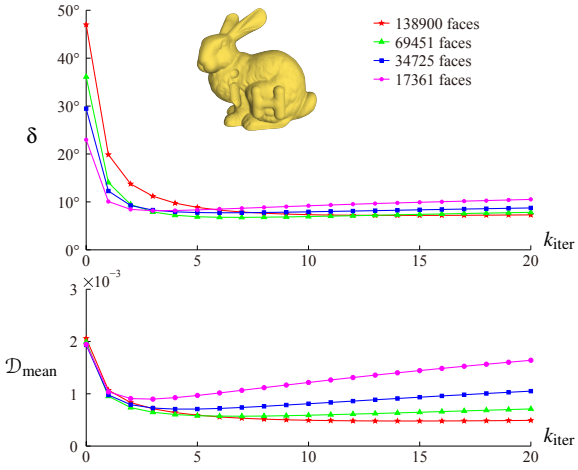

Figure 14: *Convergence plots of error metrics* δ *and* $\mathcal{D}_{mean}$ *on bunny models with different resolutions. The values of* $\mathcal{D}_{mean}$ *are normalized by the bounding box diagonal length of the ground truth model. Our method achieves desired results on lower resolution models faster than higher resolution ones.*

of vertex positions according to the filtered normals. Since the quality of the result depends heavily on the guidance for the joint bilateral filter, we propose a method for constructing a suitable guidance normal field that reliably indicates surface features in the presence of noise. Our method extends previous work on image filtering to the domain of geometry processing, and their effectiveness is validated by extensive experimental results. Our work shows the potential of joint bilateral filtering as a tool for geometry processing, thanks to its flexibility via the guidance signal. Indeed, such flexibility allows the joint bilateral filter to be applied for other geometry processing tasks as long as a proper guidance is provided, which also provides interesting avenues for further research.

Acknowledgements We thank the anonymous reviewers for their comments and suggestions, Justin Solomon and Ruimin Wang for providing the results of [SCBW14] and [WYL*14], and Mingqiang Wei for providing the program of [WYP*15]. We also thank Hao Li, Hongwei Yang and Chao Hu for their help to prepare the results and video. This work was supported by the NSF of China (Nos. 61303148, 61222206), NSF of AnHui Province, China (No. 1408085QF119), Specialized Research Fund for the Doctoral Program of Higher Education under contract (No. 20133402120002), One Hundred Talent Project of the Chinese Academy of Sciences, and Swiss National Science Foundation (grant 200021_137626).

References

[BJMO12] BACH F. R., JENATTON R., MAIRAL J., OBOZINSKI G.: Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning 4*, 1 (2012), 1–106. 7

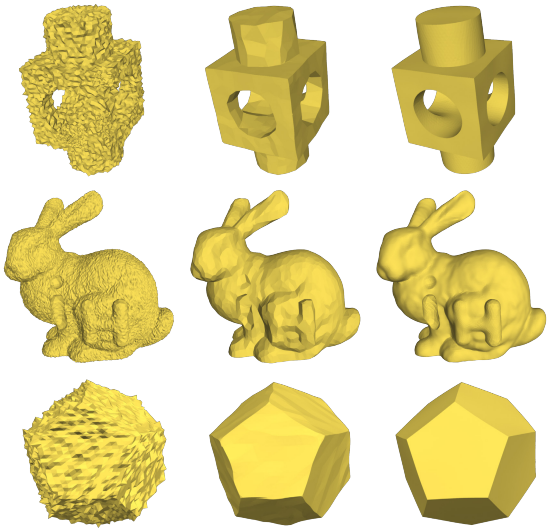

Figure 15: *Performing the vertex update according to the guidance normals instead of the filtered normals can also remove high-frequency noises, but at the same time introduces undesirable flat regions.*

[BM05] BENNETT E. P., MCMILLAN L.: Video enhancement using per-pixel virtual exposures. *ACM Trans. Graph. 24*, 3 (2005). 2

[BPK*08] BOTSCH M., PAULY M., KOBBELT L., ALLIEZ P., LEVY B., BISCHOFF S., RÖSSL C.: Geometric modeling based on polygonal meshes. Eurographics 2008 Course Notes, 2008. 2

[BT11] BIAN Z., TONG R.: Feature-preserving mesh denoising based on vertices classification. *Computer Aided Geometric Design 28*, 1 (2011), 50 – 64. 3

[CC05] CHEN C.-Y., CHENG K.-Y.: A sharpness dependent filter for mesh smoothing. *Computer Aided Geometric Design 22*, 5 (2005), 376 – 391. 3

[CLKL14] CHO H., LEE H., KANG H., LEE S.: Bilateral texture filtering. *ACM Trans. Graph. 33*, 4 (2014), 128:1–128:8. 2, 7

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. 21*, 3 (2002), 257–266. 2

[DDD04] DUGUET F., DURAND F., DRETTAKIS G.: *Robust Higher-Order Filtering of Points*. Tech. Rep. RR-5165, INRIA, 2004. 2

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (2004), 673–678. 1, 2, 4

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003). 1, 2, 3, 8, 9, 10

[FYP10] FAN H., YU Y., PENG Q.: Robust feature-preserving mesh denoising based on consistent subneighborhoods. *IEEE Trans. Vis. Comput. Graphics 16*, 2 (2010), 312–324. 3

[HS13] HE L., SCHAEFER S.: Mesh denoising via l0 minimization. *ACM Trans. Graph. 32*, 4 (2013), 64:1–64:8. 3, 8, 9, 10

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H. R.: Edge-aware point set resampling. *ACM Trans. Graph. 32*, 1 (2013), 9:1–9:12. 3

[JDD03] JONES T. R., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. 22*, 3 (2003), 943–949. 1, 2, 3, 8, 9, 10

[JDZ04] JONES T., DURAND F., ZWICKER M.: Normal improvement for point rendering. *IEEE Computer Graphics and Applications 24*, 4 (2004), 53–56. 2

[KCLU07] KOPF J., COHEN M. F., LISCHINSKI D., UYTTENDAELE M.: Joint bilateral upsampling. *ACM Trans. Graph. 26*, 3 (2007). 2

[OCDD01] OH B. M., CHEN M., DORSEY J., DURAND F.: Image-based modeling and photo editing. SIGGRAPH '01. 2

[PKTD07] PARIS S., KORNPROBST P., TUMBLIN J., DURAND F.: A gentle introduction to bilateral filtering and its applications. In *ACM SIGGRAPH 2007 Courses* (2007). 1, 2, 4

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Trans. Graph. 23*, 3 (2004), 664–672. 1, 2, 4

[SCBW14] SOLOMON J., CRANE K., BUTSCHER A., WOJTAN C.: A general framework for bilateral and mean shift filtering. *arXiv preprint arXiv:1405.4734* (2014). 1, 2, 3, 9, 10, 11

[SRML07] SUN X., ROSIN P. L., MARTIN R. R., LANGBEIN F. C.: Fast and effective feature-preserving mesh denoising. *IEEE Trans. Vis. Comput. Graphics 13*, 5 (2007), 925–938. 3, 5, 8, 9, 10

[SRML08] SUN X., ROSIN P. L., MARTIN R. R., LANGBEIN F. C.: Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design 25*, 7 (2008), 437 – 456. 3

[SSW15] SUN Y., SCHAEFER S., WANG W.: Denoising point sets via L_0 minimization. *Computer Aided Geometric Design 35-36* (2015), 2–15. 3

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. ICCV '98. 1, 2, 4

[Wan06] WANG C. C.: Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Trans. Vis. Comput. Graphics 12*, 4 (2006), 629–639. 2

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. *ACM Trans. Graph. 25*, 3 (2006). 2

[WXL*13] WANG J., XU K., LIU L., CAO J., LIU S., YU Z., GU X. D.: Consolidation of low-quality point clouds from outdoor scenes. *Computer Graphics Forum 32*, 5 (2013), 207–216. 3

[WYL*14] WANG R., YANG Z., LIU L., DENG J., CHEN F.: Decoupling noise and features via weighted ℓ_1-analysis compressed sensing. *ACM Trans. Graph. 33*, 2 (2014), 18:1–18:12. 3, 10, 11

[WYP*15] WEI M., YU J., PANG W.-M., WANG J., QIN J., LIU L., HENG P.-A.: Bi-normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graphics 21*, 1 (2015), 43–55. 3, 9, 10, 11

[WZY12] WANG J., ZHANG X., YU Z.: A cascaded approach for feature-preserving surface mesh denoising. *Computer-Aided Design 44*, 7 (2012), 597 – 610. 3

[XCS*06] XIAO J., CHENG H., SAWHNEY H., RAO C., ISNARDI M.: Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV 2006*. 2006, pp. 211–224. 2

[YOB02] YAGOU H., OHTAKE Y., BELYAEV A.: Mesh smoothing via mean and median filtering applied to face normals. GMP '02. 3

[ZFAT11] ZHENG Y., FU H., AU O.-C., TAI C.-L.: Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graphics 17*, 10 (2011), 1521–1530. 1, 2, 3, 4, 5, 7, 8, 9, 10, 11