Rolling Guidance Normal Filter for Geometric Processing

Peng-Shuai Wang[*†] Xiao-Ming Fu[‡†] Yang Liu[†] Xin Tong[†] Shi-Lin Liu[‡†] Baining Guo[†*]
[*]Tsinghua University [†]Microsoft Research [‡]University of Science and Technology of China

(a) *Input* **(b)** *Bilateral* **(c)** *Bilateral normal* **(d)** L_0 **(e)** *Laplacian* **(f)** *MH* **(g)** *APSS* **(h)** *Our RGN filter*

Figure 1: *Geometry texture removal: comparison with the state-of-the-art mesh smoothing algorithms. The input Merlion model (283k vertices) (a) contains rich small-scale features like fish scales on the body, brick pattern and text on the base. Left to right: smoothing results from (b) bilateral smoothing [Fleishman et al. 2003] ($\sigma_s = 3\,\bar{l}_{\mathbf{e}}$, 70 iterations), (c) bilateral normal denoising [Zheng et al. 2011] ($\sigma_c = \bar{l}_{\mathbf{e}}, \sigma_s = 0.5$, 30 iterations, local update scheme), (d) L_0 smoothing [He and Schaefer 2013] ($\lambda = 100\lambda_0$), (e) uniform-weighted Laplacian smoothing (80 iterations), (f) Manifold harmonic [Vallet and Lévy 2008] (400 basis on a simplified mesh with 5000 vertices), (g) APSS filter [Guennebaud and Gross 2007] ($h_i(x) = 20\,\bar{l}_{\mathbf{e}}$); (i) our rolling guidance normal filter ($\sigma_s = 5\,\bar{l}_{\mathbf{e}}, \sigma_r = 0.3$). Here $\bar{l}_{\mathbf{e}}$ is the average edge length and λ_0 is the default value suggested by He and Schafer [2013]. Methods like (b,c) can filter out the data noise but they are unable to remove the small features even with more iterations and larger parameters. L_0 smoothing cannot get rid of all the small-scale features. The Laplacian smoothing, manifold harmonic and APSS method can suppress the geometric details but with the price of over-smoothing. Our method removes the small-scale features entirely while preserving large-scale features.*

Abstract

3D geometric features constitute rich details of polygonal meshes. Their analysis and editing can lead to vivid appearance of shapes and better understanding of the underlying geometry for shape processing and analysis. Traditional mesh smoothing techniques mainly focus on noise filtering and they cannot distinguish different scales of features well, even mixing them up. We present an efficient method to process different scale geometric features based on a novel rolling-guidance normal filter. Given a 3D mesh, our method iteratively applies a joint bilateral filter to face normals at a specified scale, which empirically smooths small-scale geometric features while preserving large-scale features. Our method recovers the mesh from the filtered face normals by a modified Poisson-based gradient deformation that yields better surface quality than existing methods. We demonstrate the effectiveness and superiority of our method on a series of geometry processing tasks, including geometry texture removal and enhancement, coating transfer, mesh segmentation and level-of-detail meshing.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: Rolling guidance normal filter, geometry texture

1 Introduction

Geometric details (or *geometry texture*) like bumps, ridges, creases, structural patterns etc. are important features of 3D shapes. These features can be interpreted as geometric variations whose scales are much larger than the scale of noise. Small-scale geometric features characterize the detailed appearance of shapes while medium- and large-scale features represent intrinsic geometric properties of shapes. Analyzing and editing different scale geometric features has been received a lot of attention in computer graphics and geometric processing [Kobbelt et al. 1998; Guskov et al. 1999; Pauly et al. 2006; Nader et al. 2014].

Many geometric processing tasks, like coating transfer [Sorkine et al. 2004] and multi-resolution editing [Botsch and Sorkine 2008], require a decomposition of a 3D model into different scales, where a suitable mesh smoothing operator is needed. Traditional mesh smoothing techniques like Laplacian smoothing, bilateral and mean-shift smoothing [Fleishman et al. 2003; Zheng et al. 2011; Solomon et al. 2014] mainly focus on the removal of random noise arising from 3D acquisition and digitization processes. They cannot remove small-scale geometry features clearly and preserve medium- or large-scale features well (see Fig. 1-b,c,e). Therefore directly applying existing mesh smoothing algorithms cannot carry out the task of multi-scale geometry feature analysis and editing. Spectral based multi-scale methods [Vallet and Lévy 2008; Sun et al. 2009; Zhang et al. 2010] are useful tools for tackling this task. However, their costly computation and inability to support local surface editing are the main obstacles for practical use. Furthermore, they cannot preserve sharp features in mesh smoothing applications. Recently Nader et al. [2014] use the algebraic point set surfaces (APSS) method [Guennebaud and Gross 2007] to analyze point clouds and meshes at multiple scales. Their method efficiently associates the fitting regional scale with the geometric feature and supports both local and global editing. But due to the usage of spherical primitives, non-spherical features become sphere-like, for instance, a

high-curvature region can become rounded after processing (see the upper and bottom edges of the base in Fig. 1-g).

Inspired by the recent *rolling guidance filter* [Zhang et al. 2014] in image processing that filters images with complete control of detail smoothing under a scale measure, we propose a simple and effective scale-aware mesh smoothing filter to process different scales of geometry features of triangular meshes. The key idea is to smooth out small geometric variations while preserving large-scale features by applying an iterative joint bilateral filter to face normals. Our contributions are summarized as follows:

- We introduce a *rolling guidance normal filter* (RGN) for mesh smoothing which is a novel extension of the rolling guidance (RG) filter. Our RGN filter inherits good properties of RG that separates different scale geometric features and it is very efficient and intuitive for local and global editing.
- We propose a modified Poisson-based gradient algorithm to recover the surface geometry from the filtered face normals. Our algorithm significantly reduces triangle flips compared to existing linear vertex-updating approaches.
- We show the effectiveness and superiority of RGN on a series of applications, including geometry texture removal and enhancement, coating transfer, mesh segmentation and level-of-detail quadrilateral meshing.

2 Related Work

Mesh smoothing (or mesh denoising) techniques are essential in geometry processing (cf. detailed surveys [Botsch et al. 2010, Chapter 4]). Early low-pass filter methods like Taubin smoothing [Taubin 1995] and implicit mean curvature flow [Desbrun et al. 1999] focus on noise removal and handle the shape shrinkage problem that occurred in traditional Laplacian smoothing. These methods and their variations are intrinsically isotropic filters that cannot preserve surface features. Anisotropic diffusion algorithms [Clarenz et al. 2000; Tasdizen et al. 2002] are proposed to remove noise and preserve sharp edge. Their performance and results are highly dependent on the grid size and the time steps. It is unclear how to intuitively control the scale of the geometry details that the filter is applied to. Bilateral mesh filters were later introduced by Jones et al. [2003] and Fleishman et al. [2003] to retain features since bilateral filters are edge-preserving [Tomasi and Manduchi 1998]. However, these bilateral methods cannot well preserve sharp corners and features because the bilateral weights are not sensitive to the change of face/vertex normals. This issue was later resolved by normal-based bilateral mesh filters [Lee and Wang 2005; Zheng et al. 2011; Wei et al. 2015] which first filter face normals, then update mesh vertices to match the filtered face normals. Recently Solomon et al. [2014] provide a more theoretically sound generalization of bilateral and mean shift filter on images, meshes, and other domains within a single unified framework via heat diffusion, which can also be used for feature-preserving mesh denoising. Other techniques like L_0 smoothing [He and Schaefer 2013], total variation [Zhang et al. 2015] and l_1-analysis optimization [Wang et al. 2014] are also capable of keeping or enhancing sharp features while removing noise.

However, noise is usually defined in the literature as high-frequency geometric oscillation whose scale is much smaller than geometric features like bumps and creases. Most smoothing algorithms hardly work on geometric feature removal even on small-scale features as shown in Fig. 1-b,c,d. Multi-scale mesh analysis is an effective technique to process different scale features, which filters the mesh by using a series of Laplacian filters [Sorkine et al. 2004], or Moving-Least-Squares (MLS) [Adamson and Alexa 2003; Pauly et al. 2006], or spectral analysis [Vallet and Lévy 2008]. But these methods cannot well preserve large-scale features when removing small-scale features (see Fig. 1-e,f,g). Our method solves this problem by iteratively recovering large-scale features after removing small-scale features.

Image texture smoothing has gained great attention in image processing, the aim of which is to smooth an image while persevering different scale structures of the image. Subr et al. [2009] propose to average two manifolds from the local minimum and maximum to smooth a texture. Xu et al. [2012] employ relative total variation to preserve structural edges. Recently Zhang et al. [2014] introduce the Rolling Guidance Filter (RG) that first removes small structures then recovers structural edges iteratively. Their method separates different scale structures without artifacts and runs in realtime. Cho et al. [2014] present a joint bilateral texture filter that also preserves different scale structures. Their method has similar results and performance to Zhang et al. 's [2014]. We extend the rolling guidance filter to mesh processing by integrating it with the bilateral normal filter.

3 Rolling guidance normal filter on meshes

In Sec. 3.1 we first review the rolling guidance filter [Zhang et al. 2014] , then extend it to mesh processing and introduce our rolling guidance normal filter in Sec. 3.2. We develop an efficient vertex updating scheme that uses a modified Poisson-based gradient deformation with filtered normals in Sec. 3.3.

3.1 Rolling guidance filter

The bilateral filter [Tomasi and Manduchi 1998] for an image pixel $I(\mathbf{p})$, at coordinate $\mathbf{p} = (x, y)$, is defined as:

$$I(\mathbf{p}) = \frac{1}{K_{\mathbf{p}}} \sum_{\mathbf{q} \in N(\mathbf{p})} W_s(\|\mathbf{p} - \mathbf{q}\|)\, W_r(\|I(\mathbf{q}) - I(\mathbf{p})\|)\, I(\mathbf{q})$$

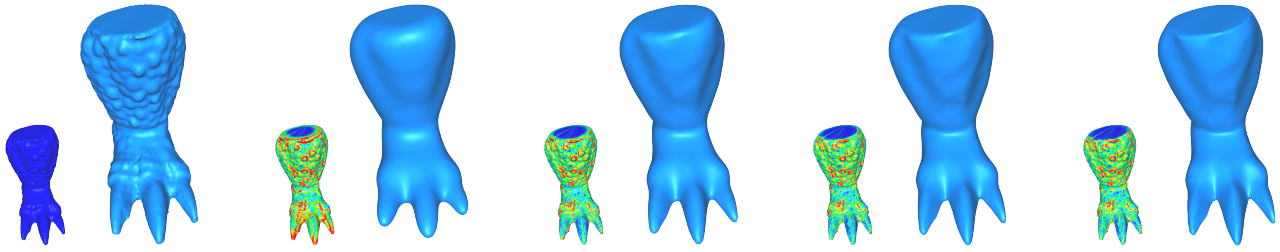
where $N(\mathbf{p})$ defines the neighborhood of $\mathbf{p}$, W_s and W_r are monotonically decreasing weighting functions characterizing the position similarity and the color similarity between $\mathbf{p}$ and $\mathbf{q}$ respectively, which are often chosen to be Gaussian functions, with standard deviations σ_s and σ_r. $K_{\mathbf{p}}$ is the normalization term, i.e. the summation of weights. When another image J, instead of I itself, is fed to the function W_r, then this filter is called a *joint bilateral filter* [Kopf et al. 2007]. The image J is referred to as the *guide image*. When σ_r approaches infinity or the values of J are all equal, this filter degenerates to a traditional Gaussian filter.

The *rolling guidance filter* [Zhang et al. 2014] is essentially an iterative joint bilateral filter, whose guide image is updated iteratively. Given the input image I, and the iteration number N_{iter}, the rolling guidance filter produces an output image $J^{N_{iter}}$. The iterative process can be summarized as:

$$J^{k+1}(\mathbf{p}) = \frac{1}{K_{\mathbf{p}}} \sum_{\mathbf{q} \in N(\mathbf{q})} W_s(\|\mathbf{p}-\mathbf{q}\|)\, W_r(\|J^k(\mathbf{p})-J^k(\mathbf{q})\|)\, I(\mathbf{q})$$

Here k denotes the iteration number and J^0 is set be *zero* in practice.

The principle of the rolling guidance filter can be understood as follows. In the first iteration, the filter is actually a Gaussian filter since $J^0 = \mathbf{0}$. Thus given some proper value of σ_s, the small image features whose scale are smaller than σ_s can be smoothed empirically as shown in [Zhang et al. 2014]. However, the edges of large structures are also smoothed to some degree. In the later iterations, J^t is no longer zero. For the removed small features, their guide pixels are almost equal. So the weight term W_r approximately equals 1, and the filter is still a Gaussian filter, resulting in small structures still being removed. For large-scale structures, they are recovered gradually since the joint bilateral filter sharpens the smoothed edges of large structures.



(a) *Input* **(b)** 1^{st} *iter* **(c)** 2^{nd} *iter* **(d)** 3^{th} *iter* **(e)** 4^{th} *iter*

Figure 2: *We apply the rolling guidance normal filter on a leg of the Armadillo model (courtesy of Stanford 3D Scanning repository). We color-code the angle difference between the filtered face normals and the input face normals (see the colored meshes, the angle difference increases from blue to red). We can see that the normals on the bumps have the maximal changes because the scale of corresponding features is under the specified σ_s. The blue models are meshes whose vertex positions are updated by our Poisson deformation method. In the first iteration small bumps are totally removed and large-scale features like toes are smoothed too. In the subsequent iterations, small features are kept removed, while large features are recovered gradually.*

3.2 Rolling guidance normal filter

When adapting the rolling guidance filter to mesh processing, one needs to generalize the bilateral filter to operate on a 3D mesh. There are already many attempts to deal with this problem (cf. [Solomon et al. 2014, Table 1]). Methods like [Fleishman et al. 2003; Jones et al. 2003] apply the filter to the vertex positions on the local tangential plane, thus the size of the neighborhood cannot be too large by nature. So it is not suitable for the removal of geometry features whose scale is not very small. Differently, Lee et al. [2005] and Zheng et al. [2011] propose to apply the bilateral filter to the mesh face normal field and evolve the mesh geometry to match the filtered normals. They regard the normal field as a geometric signal defined over the input mesh, discretized at face centers. In this way, the geometric features and spatial location of a mesh are decoupled. Additionally, since the normal signal changes abruptly over sharp features, the normal-based denoising algorithm can smooth out noise and keep the sharp features.

In our scenario, we want to smooth out geometry features, whose scale is much larger than the mesh noise. Essentially, a larger smoothing neighborhood is needed. So the algorithms of Fleishman et al. [2003] and Jones et al. [2003] do not fulfill the scenario. We follow the bilateral framework of Lee and Wang [2005] and Zheng et al. [2011] and apply the rolling guidance filter to the mesh face normal field.

Define the mesh as $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{N_v}$ represents the set of vertices, $\mathcal{F} = \{f_i\}_{i=1}^{N_f}$ the set of faces. Denote the centroid of triangle 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. As before, we choose the Gaussian function as the weight function, then the rolling guidance filter for mesh normals is written as:

$$\mathbf{n}_i^{k+1} = \frac{1}{K_i} \sum_{f_j \in \mathcal{F}} A_j \, W_s(\|\mathbf{c}_i - \mathbf{c}_j\|) \, W_r(\|\mathbf{n}_i^k - \mathbf{n}_j^k\|) \cdot \mathbf{n}_j$$

or more explicitly

$$\mathbf{n}_i^{k+1} = \frac{1}{K_i} \sum_{f_j \in \mathcal{F}} A_j \, \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{2\sigma_s^2} - \frac{\|\mathbf{n}_i^k - \mathbf{n}_j^k\|^2}{2\sigma_r^2} \right) \cdot \mathbf{n}_j \tag{1}$$

where K_i is the normalization term, and $\mathbf{n}^0 = \mathbf{0}$. The normal is normalized to length 1 after filtering in each iteration. We call Eq. 1 the *rolling guidance normal filter*. From the geometric point of view, Eq. 1 can be regarded as a Gaussian smoothing of the signal $\mathbf{n}_i$ defined at the points $(\mathbf{c}_i, \mathbf{n}_i^k) \in \mathbb{R}^6$.

In practice, σ_r can be chosen between 0.1 to 0.6. σ_s is related to the scale size of geometry features. If the geometry feature can be covered by a ball of radius r, then we could simply set σ_s to be r. These features are greatly smoothed in the first iteration by the Gaussian filter. Other large features are recovered by the later joint bilateral filtering, while small scale features are kept removed, similar to the original RG filter. The last parameter is the total rolling iteration number N_{iter}, which is often set to be 4, 5 or 6, which usually yields converged results in practice although there is no theoretical guarantee. Fig. 2 shows the filtered normals and reconstructed geometry of a model at different iterations.

As for implementation, one straightforward choice may be Zheng et al.'s [2011] local iterative normal update scheme, which is confined to a 1-ring or 2-ring neighborhood, and uses multiple iterations of updating to increase the influence of a filter from a small neighborhood to a wider region. Different from noise removal, the geometry feature removal that we consider needs the geometry information of a much larger neighborhood. Denote N_{σ_s} as the number of neighboring faces that fall into the support area of a Gaussian with kernel size σ_s. If we follow Zheng et al.'s framework, the time complexity in each iteration is at least $\mathcal{O}(N_{\sigma_s} N_f)$ which slows down the algorithm when N_{σ_s} is large. So we include all the faces into the weighted summation, and do normal updating only one time in each iteration. As presented before, Eq. 1 could be regarded as a high-dimensional Gaussian transformation. So we take advantage of *Permutohedral Lattice* [Adams et al. 2010] to reduce the time complexity in each iteration to $\mathcal{O}(d^2 N_f)$, where d is the dimensionality of the point space, which is 6 here. This approach is widely used in image processing, and has been proven to be much faster than the multi-pass local update method [Adams et al. 2010]. Other speed up techniques such as *Gaussian KD-Tree* [Adams et al. 2009] can be used as well.

Remark 1: Ideally the distance measurement between point $\mathbf{c}_i$ and $\mathbf{c}_j$ in Eq. 1 should be the 3D geodesic distance. We have tested the approach that uses Landmark MDS [Silva and Tenenbaum 2002; Panozzo et al. 2013] to efficiently compute a Euclidean embedding into higher dimensions for the input mesh, and then uses the Euclidean distance in higher dimensions to approximate the geodesic distance. But we have not found obvious differences using this approach for all the models in our experiments. So for simplicity and efficiency, we use the 6D Euclidean distance directly which is a kind of Riemannian embedding of meshes in 6D [Lévy and Bonneel 2013] and approximates the geodesic distance in some sense.

Remark 2: Recently Solomon et al. [2014] propose an implicit framework for bilateral filtering that supports large scale smoothing

and uses intrinsic and smooth distance weights respecting the domains metric. It would be possible to formulate the rolling guidance normal filter based on their formulation.

3.3 Vertex updating

After obtaining filtered face normals $\hat{\mathbf{n}}$, the mesh vertices should be updated to match the modified face normals. To this end, one common algorithm is to minimize the following quadratic energy function to get the new vertex positions [Sun et al. 2007]:

$$E(\mathbf{V}) = \sum_{f_i} \sum_{\mathbf{e}_{jk} \in \partial f_i} A_i \left[\hat{\mathbf{n}}_i \cdot (\mathbf{v}_j - \mathbf{v}_k)\right]^2$$

where $\mathbf{e}_{jk}$ is an edge of triangle f_i with end vertices $\mathbf{v}_j$ and $\mathbf{v}_k$.

Based on this energy, one can solve a least squares problem to update the vertex position, or simply follow Sun et al.'s local updating approach [2007], which is essentially the gradient descent method to minimize $E(\mathbf{V})$ with a fixed step size. For mesh denoising, the updated vertices will not deviate too far away from the expected positions, and this algorithm often performs quite well. However, when considering removing geometry features, whose scale is much larger than mesh noise, the normal change is so big that the recovered mesh vertex positions deviate greatly from the original. Thus directly optimizing this energy will cause frequent triangle flipping and will degrade surface quality (see Fig. 3-b). What is more, optimizing this energy function will lead to a linear system whose matrix coefficients depend on the filtered normals, so this linear system cannot be pre-factorized for later usage. When the users tweak the parameters of the rolling guidance normal filter for a desired result, the algorithm cannot give feedback interactively even for a moderately sized mesh due to the computational complexity.

To circumvent this problem, we regard this normal-guided vertex updating problem as a mesh deformation problem and adopt the *Poisson Mesh Deformation* [Yu et al. 2004] as our solver. Considering the triangle f_i with its original normal $\mathbf{n}_i$ and its filtered normal $\hat{\mathbf{n}}_i$, we can define a local rotation to align these two normals with a minimal rotation angle about the rotating axis $\mathbf{n}_i \times \hat{\mathbf{n}}_i$. This rotation is applied to triangle f_i to obtain a new triangle $\bar{f}$. on $\bar{f}$, we compute its gradient vector $\nabla \hat{f}_{i,j}, j = 1, 2, 3$ that are gradients of the piecewise-linear basis nodal function defined on $\bar{f}$. Recovering vertex positions amounts to minimizing the following energy function:

$$E_{Poisson} = \sum_{f_i} A_i \sum_{j=1}^{3} \|\nabla g_{i,j} - \nabla \hat{f}_{i,j}\|_2^2,$$

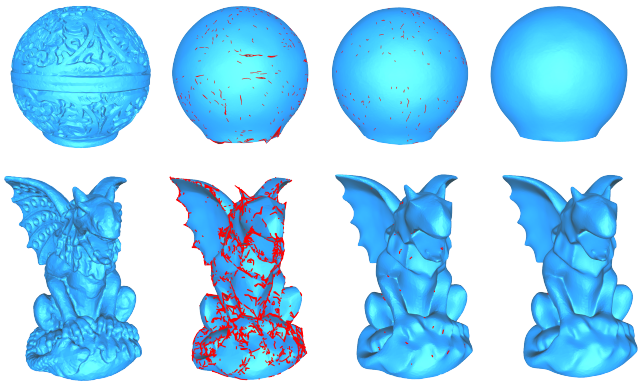
where $\nabla g_{i,j}, j = 1, 2, 3$ are the gradient vectors on the face f_i with unknown vertex positions. Minimizing this energy function with respect to vertex positions leads to the well-known Poisson equation, which can be solved efficiently.

In some cases, the geometry texture is so complicated that a rapid change of face normals will still cause triangle flipping in minimizing $E_{Poisson}$. To ease this problem, we add a gradient smoothness regularization term as follows.

$$E_{smooth} = \sum_{\mathbf{e}_{ij}} l_{e_{ij}} \sum_{k=1}^{3} \|\nabla g_{i,k} - \mathbf{R}_{ij} \nabla g_{j,k}\|_2^2,$$

where $\mathbf{e}_{ij}$ represents an edge adjacent to faces f_i and f_j, $l_{\mathbf{e}_{ij}}$ is the edge length, and $\mathbf{R}_{ij}$ is the rotation matrix that rotates triangle f_j along $\mathbf{e}_{ij}$ to be in the same plane of f_i.

E_{smooth} regularizes the smoothness of the target gradient vector field with regard to the Levi-Civita connection defined on the original mesh [Crane et al. 2013]. We observe that around flipped triangles, the resulting gradient vectors of the target mesh often change quite abruptly. Our smoothness term penalizes the quick changes thus reducing triangle flips.



(a) *Input* **(b)** *LS* **(c)** *Poisson* **(d)** *our method*

Figure 3: *Comparison of vertex updating algorithms. The flipped triangles are rendered in red. (a) the original mesh; (b) the mesh is updated by the least square approach [Sun et al. 2007]; (c) the mesh is updated by minimizing $E_{Poisson}$; (d) the mesh is updated by minimizing E_{update}.*

So the final energy function can be written as

$$E_{update} = \frac{E_{Poisson}}{\bar{A}} + \lambda \frac{E_{smooth}}{\bar{l}_{\mathbf{e}}} \tag{2}$$

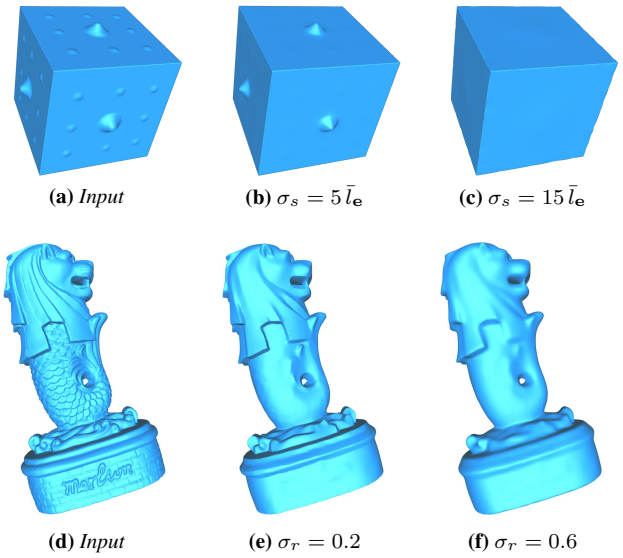
where average face area $\bar{A}$ and average edge length $\bar{l}_{\mathbf{e}}$ are used to normalize $E_{Poisson}$ and E_{smooth} under a unform scale, λ balances two energy terms. Larger λ value reduces more flipped elements, while increasing the approximation error of the target normals. We simply choose it as 0.5 in our experiments.

Fig. 3 shows that our algorithm can greatly reduce triangle flipping compared to the existing approaches. We also note that E_{update} is a quadratic function with respect to vertex positions, so it can be solved efficiently by a sparse linear solver and the resulting linear system can be pre-factorized by Cholesky decomposition for interactive editing.

4 Experiments and Applications

Implementation and timing For implementing the RGN filter efficiently, we use the Permutohedral Lattice [Adams et al. 2010], whose time complexity is $\mathcal{O}(d^2 N_f)$ in each iteration. Since the iteration number of the RGN filter is quite small (5 at most), the filtering process is quite fast. We use the Eigen library [Guennebaud et al. 2010] to compute the Cholesky decomposition. We conducted our experiments on a desktop PC with a 3.4 GHz Intel Core i7 CPU and 16 GB of RAM. Table 1 reports the timings and parameters of our algorithm on some models.

Parameter setting The choice of parameters is quite intuitive, as presented in Sec. 3.1. σ_r is related to the desired smoothness of the final results, σ_s is related to the scale size of geometry features. The total rolling iteration number N_{iter} is fixed to 5 in all our experiments. Fig. 4 shows the effect of different values of σ_r and σ_s on two models. Increasing the value of σ_r gradually increases the smoothness of final results. Increasing σ_s gradually smoothes geometry feature of different scales while preserving mostly large-scale features. Using these properties, we can decompose features of different scales from the original mesh, opening the door for more effective multi-scale analysis. Since the filtered normals do not affect



(a) *Input* **(b)** $\sigma_s = 5\,\bar{l}_{\mathbf{e}}$ **(c)** $\sigma_s = 15\,\bar{l}_{\mathbf{e}}$

(d) *Input* **(e)** $\sigma_r = 0.2$ **(f)** $\sigma_r = 0.6$

Figure 4: *Effects of different values of σ_s and σ_r. Upper row: (a) the input cube model; (b)$\sigma_s = 5\,\bar{l}_{\mathbf{e}}$; (c) $\sigma_s = 15\,\bar{l}_{\mathbf{e}}$. σ_r is fixed to 0.2, N_{iter} is fixed to 5. When increasing the value of σ_s , the geometry features of different scales are gradually removed and the large-scale geometry features are preserved. Bottom row: (d) the input Merlion model; (e) $\sigma_r = 0.2$; (f) $\sigma_r = 0.6$. Here σ_s is fixed to $5\,\bar{l}_{\mathbf{e}}$, N_{iter} is fixed to 5.*

the Cholesky decomposition, the user can change the parameters and get the updated mesh in realtime for a moderately sized mesh.

4.1 Geometry texture removal

Although there are already a lot of studies about mesh denoising, these algorithms cannot be directly adapted to large geometry texture removal, because of the inherent difference between geometry texture and model noise, as shown in Fig. 5. Our RGN filter is shown to produce superior results.

In this scenario, applying the Laplacian filter will not only smooth out the geometry texture, but also over-smooth the original sharp structure, and may even cause degeneration or volume shrinkage without care.

The methods of Lee and Wang [2005] and Zheng et al. [2011] apply the bilateral filter iteratively to the normal field for feature-preserving mesh denoising. Although these methods share the similar formula as RGN, they are designed for different purposes. The RGN is used for removing small scale features while preserving large scale structures. In contrast, the mentioned methods are designed for removing noise while preserving high *contrast* sharp features (here contrast means the amplitudes of normal oscillations). The high contrast sharp features have low similarity, so when averaging face normal signals they will not interfere with each other thus can be preserved. Small scale features are signals with small spatial oscillations. RGN first smoothes out small scale features with a Gaussian filter on normals. And in the following iterations, small scale features are kept removed and large scale features which are smoothed in the first iteration are recovered gradually. As shown in Fig. 5, small scale features we want to remove possess quite high contrast, bilateral filters preserve these features and can hardly work for our purpose.

The recent L_0 smoothing [He and Schaefer 2013] optimizes the sparseness of the edge Laplacian, and can remove all kinds of geometry features. This algorithm works better for CAD models, especially piecewise planar surfaces. However it is well known that L_0 will generate staircase artifact on smooth surfaces. Decreasing the iteration number can relieve the artifact, but will also keep the geometry textures that should be removed. Furthermore, when the mesh resolution is high, the geometry textures are hard to remove even with more iterations, as shown in Fig. 1 & 5. And the algorithm runs very slowly because it needs to solve large sparse linear systems many times.

Our method utilizes the power of the rolling guidance normal filter, and is specially designed for geometry texture removal. It can easily remove these geometry textures efficiently while preserving the original shape quite well. And because of the Poisson-based vertex updating algorithm, we have no volume shrinkage artifacts.

4.2 Geometry texture editing

Sculpting geometry appearance of 3D models is of great importance in practice. One may want to edit the surface geometry texture to achieve different visual effects. With the help of our RGN filter technique, we can edit the geometry texture easily.

Denote the gradient vectors of triangle f_i of the input mesh as $\nabla f^0_{i,j}, j = 1, 2, 3$. After applying our RGN filter, we get a new face normal field, and accordingly the new triangle gradient vector, $\nabla \hat{f}_{i,j}, j = 1, 2, 3$, as explained in Sec. 3.3. We can interpolate or extrapolate the gradient vectors $\nabla f^0_{i,j}$ and $\nabla \hat{f}_{i,j}$, and reconstruct the mesh by minimizing the following energy function without any computational overhead since only the right hand side of the resulting linear system is changed compared to Eq. 2.

$$E_{Edit} = \sum_{f_i} \frac{A_i}{\bar{A}} \sum_{j=1}^{3} \|\nabla g_{i,j} - (t \cdot \nabla f^0_{i,j} + (1-t) \cdot \nabla \hat{f}_{i,j})\|^2 + \lambda \frac{E_{smooth}}{\bar{l}_{\mathbf{e}}}$$

Fig. 6 shows interesting results with feature-exaggerated and engraved effects on two models.

Remark 3: The screened Poisson method [Chuang and Kazhdan 2011] also provides a solution to exaggerate geometry by scaling gradient vectors while trying to preserve the overall shape. However, our method can rotate and scale the target gradient and thus has more editing freedom, such as transforming embossed geometry to engraved geometry and vice versa. For instance, see results in the right column of Fig. 6. These transformation effects cannot be achieved by just scaling gradients.

Interactive editing One can also edit some regions of interest, and only recover vertex positions on selected regions. In this scenario, we fix one-ring or two-ring boundary vertices of selections when minimizing E_{Edit}. An editing result is shown in Fig. 7.

4.3 Coating transfer

Coating transfer [Sorkine et al. 2004] is a very useful mesh editing tool, with which we can smooth out the geometry texture of a 3D model and coat it with other geometry textures. When the commonly used Laplacian filter is used to peel the 3D model, features of different scales are mixed together. Unlike Laplacian smoothing, our RGN filter can decompose the mesh model in a structure-aware manner. As mentioned in Sec. 3.2, the features with scale smaller than σ_s tend to be removed by the filter, and the features with scale larger than σ_s are kept.

After applying the filter with some specified spatial kernel σ_s, we get the new Laplacian coordinates of the filtered mesh, and the difference of Laplacian coordinates with the original mesh can be encoded as the coating of the mesh. This coating can be transferred

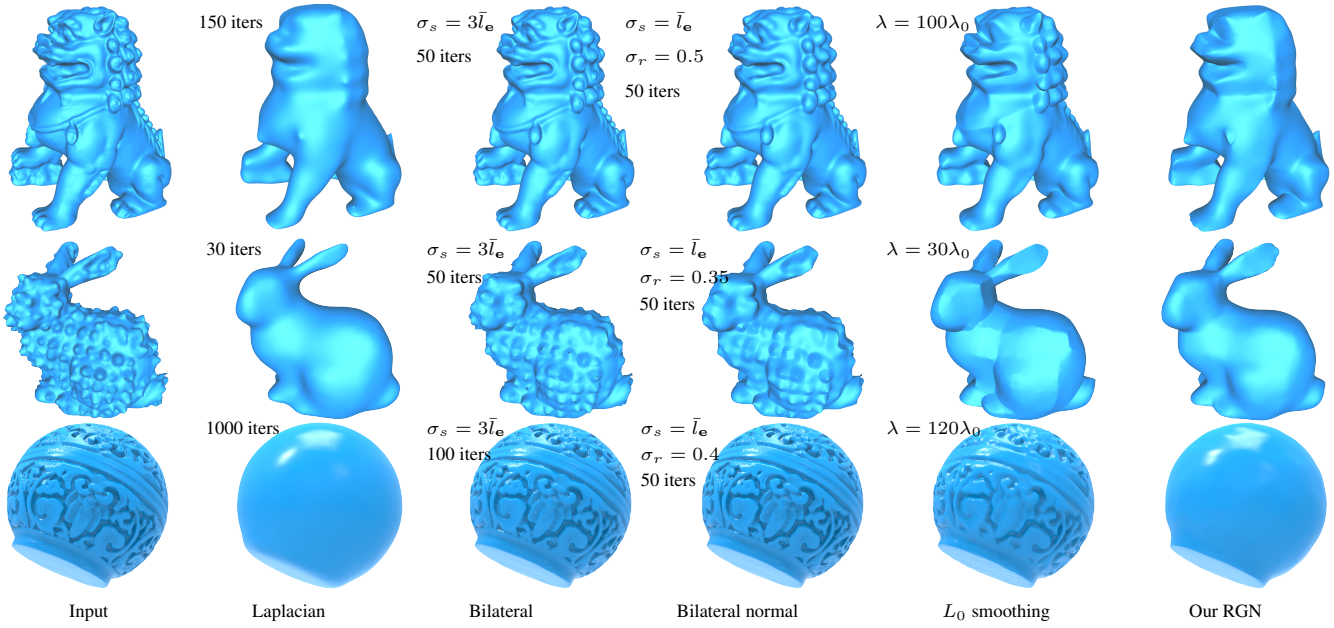


Figure 5: *Geometry texture removal. We applied uniform-weighted Laplacian smoothing, the bilateral filter [Fleishman et al. 2003], the bilateral normal filter [Zheng et al. 2011] using the local update scheme, L_0 smoothing [He and Schaefer 2013] and our RGN filter on the Chinese Lion, Bumpy Stanford Bunny and Circular Box models. We aim to remove the small features like the bumpy hair of Chinese Lion, the bumps of Bunny and the carved pattern on Circular Box. The parameters of each method are best tuned to remove the small features as much as possible.*

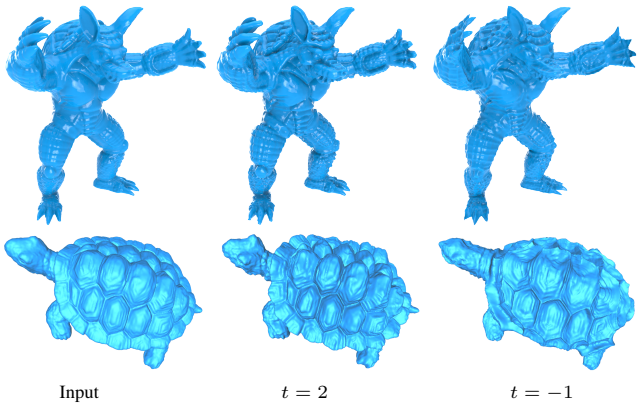
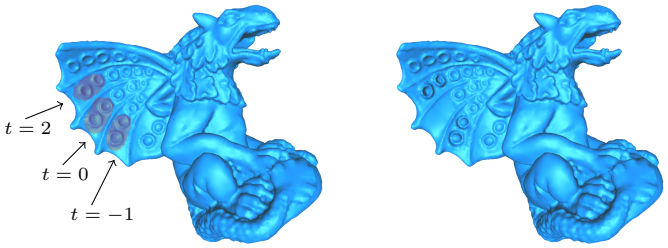


Figure 6: *Texture editing. Left column: original meshes (courtesy of AIM@SHAPE repository). Middle column: feature-exaggerated results. Right column: feature-engraved results.*

to other meshes (see Fig. 8). Compared with Laplacian smoothing, the RGN filter preserves large scale feature and will not shrink model volume, thus producing better results.

4.4 Mesh segmentation

The problem of 3D mesh segmentation is of great importance in computer graphics. Many algorithms rely heavily on surface curvature or the dihedral angle of faces [Golovinskiy and Funkhouser 2008; Chen et al. 2009; Zhang et al. 2012; Au et al. 2012]. When dealing with models full of geometry details, these algorithms often fail to give a satisfying results due to the sensitivity to small-scale geometric features. For instance, the Dinosaur model cannot be segmented into meaningful parts well as shown in [Au et al. 2012]. For these algorithms, our RGN filter can act as a preprocessing step to remove the distracting geometry textures and keep the large-scale sharp features.



(a) *Input with selected regions* **(b)** *Editing results*

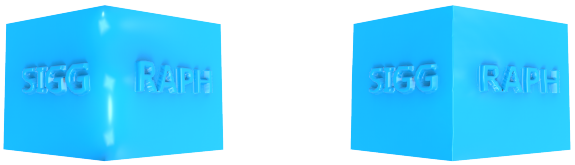
Figure 7: *ROI editing. Three regions are selected on the input mesh (a). We use different t values to edit these regions (b).*

We use the variational mesh decomposition [Zhang et al. 2012] to verify the effect of our filter. Since the models in the segmentation benchmark [Chen et al. 2009] are quite smooth and simple, our preprocessing has little effects on most shapes. Here we use the Dinosaur and Gargoyle models to demonstrate our method. As shown in Fig. 9, the segmentation results are greatly improved. The body, neck, and leg of the dinosaur are clearly cut out. The wing and arms of the gargoyle are also clearly segmented. Although Laplacian smoothing can also smooth out the geometry details, Laplacian smoothing over-smooths the large scale sharp features. So using Laplacian smoothing as a preprocessing can hardly help, and may even make the segmentation result worse.

4.5 Level-of-detail quadrilateral meshing

Directional field guided quadrilateral meshing (cf. detailed survey [Bommes et al. 2013]) is a powerful tool for achieving high-quality feature-aligned quad meshes. It usually consists of three steps: (1) directional field generation; (2) field-guided mesh parameterization; (3) quad mesh extraction. As pointed out by Ray et al. [2009] and Ebke et al. [2014], the incompatibility between small-

(a) *Source mesh* **(b)** *Target mesh*



(c) *Laplacian coating* **(d)** *Our method*

Figure 8: *Coating transfer. We transfer the "SIGGRAPH" text on the bumpy source mesh (a) to the target mesh (b). The Laplacian coating method [Sorkine et al. 2004] cannot preserve the sharp edge of the target mesh because the sharp feature is smoothed out (c). Our method does not have this artifact (d).*

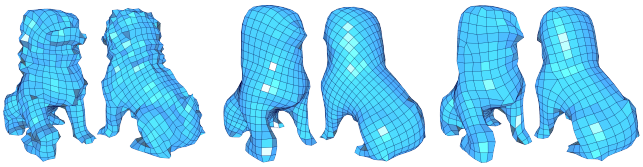
(a) Original **(b)** Laplacian **(c)** RGN filter

Figure 9: *Mesh segmentation. We applied Zhang et al. 's [2012] method on the Dinosaur and Gargoyle model (courtesy of AIM@SHAPE repository). The input models (see the blue insets) are (a) the original model; (b) the smoothed model by Laplacian smoothing; (c) the model with small features removed by our RGN filter. The segmentation results are shown on the original meshes.*

scale geometric features and the desired quad edge length set in step (2) can introduce large distortion or make the meshing algorithm fail.

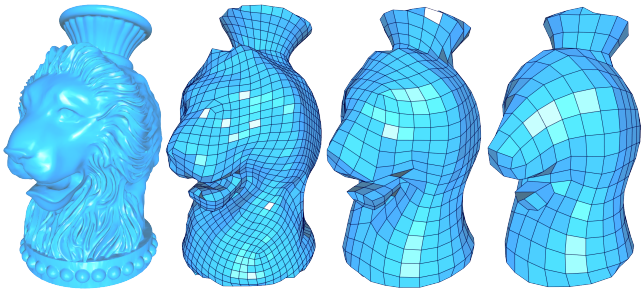
As shown in Ebke et al. 's work [2014], two different paradigms can potentially address this problem: (1) Pre-process the input mesh in order to make it satisfy the requirements of the quad meshing algorithms; (2) Make the meshing algorithms capable of dealing with the input geometry. There are a few attempts for the second paradigm. Ray et al. [2009] propose to smooth the Gaussian curvature in the cross field computation to reduce the number of singularities and avoid singularities from being closely placed. Ebke et al. [2014] apply a scale-aware Gaussian smoothing to face normals and construct a feature-aligned field on the filtered normal planes. They also modify the parameterization to integrate the scale and the effects of filtered normals. Compared to Ray et al. 's work [2009], their method can achieve lower distortion.

Note that Ebke et al. [2014] use a Gaussian normal filter as their first step. Our RGN filter can replace their Gaussian normal filter to suppress small-scale features more effectively. Here we leave such



(a) *Ebke et al. 's method* **(b)** *Laplacian* **(c)** *Ours*

Figure 10: *Quad meshing of the Chinese Lion model. Left: [Ebke et al. 2014]'s result (60 singularities). Middle: meshing result using Laplacian preprocessing (25 singularities). Right: our result (40 singularities).*



original model 33 singularities 24 singularities 24 singularities

Figure 11: *Meshing results on our RGN filtered meshes on the Vase-lion model. From left to right: σ_s is $3\,\bar{l}_e$, $5\,\bar{l}_e$ and $7\,\bar{l}_e$ respectively.*

direct integration as future work. In this paper, we follow the first paradigm and use our resulting RGN mesh as the input for quad meshing. We follow the standard mixed-integer approach [Bommes et al. 2009] to get the parameterization result and then use QEx [Ebke et al. 2013] to extract the quad mesh robustly. The cross field we used is dominated by the principal-curvature directions.

Fig. 10 shows quad meshing results of the Chinese Lion model. We set σ_s and the target edge length as $5\,\bar{l}_e$. Our method preserves the global shape while removing most bumps with fewer singularities than Ebke et al. 's [2014]. We also replace our RGN filter by Laplacian smoothing in our framework and the resulting mesh has the least singularities but large-scale structures are not preserved.

Fig. 11 shows level-of-details quad meshing of the Vaselion model using different values of σ_s. Target edge lengths are set to σ_s. We can see that the resulting meshes have few singularities due to the removal of complex small features.

5 Conclusion and Discussion

We present a novel rolling guidance normal filter for geometry processing. Our RGN filter can efficiently smooth out small-scale geometric features and preserve large-scale features of 3D models. We demonstrate the effectiveness and the efficiency of our filter in various computer graphics applications. We believe that other geometric algorithms and applications can also benefit from our work.

As a limitation, our RGN method is not well suited for removing large noise because the mixed geometric feature and noise cannot be distinguished by RGN. Fig. 12 shows comparisons with state-of-the-art denoising methods. It is clear that our method cannot recover the sharp feature well. Thus for removing geometry textures from a very noisy model, we recommend to denoise the model first using other methods.

Another limitation is that our smoothing results might still contain flip triangles and self-intersection if the changes of filtered face

Model	N_v	σ_s	σ_r	Filter	Decomp	VUpdate
Fig. 9-Dinosaur	14k	2 $\bar{l}_e$	0.13	0.39 s	0.04 s	0.02 s
Fig. 5-Bunny	14k	3 $\bar{l}_e$	0.7	0.29 s	0.08 s	0.03 s
Fig. 8 (a)	33k	12 $\bar{l}_e$	0.5	0.36 s	0.23 s	0.06 s
Fig. 8 (b)	33k	13 $\bar{l}_e$	0.2	0.32 s	0.23 s	0.06 s
Fig. 2-Leg	69k	6 $\bar{l}_e$	0.3	0.71 s	0.41 s	0.12 s
Fig. 6-Turtle	134k	14 $\bar{l}_e$	0.4	0.94 s	1.15 s	0.22 s
Fig. 5-Chinese lion	153k	8 $\bar{l}_e$	0.4	1.23 s	1.47 s	0.35 s
Fig. 6-Armadillo	173k	5 $\bar{l}_e$	0.3	1.51 s	0.90 s	0.30 s
Fig. 1-Merlion	283k	5 $\bar{l}_e$	0.3	2.39 s	3.00 s	0.53 s
Fig. 5-Circular box	701k	12 $\bar{l}_e$	0.4	5.14 s	25.0 s	1.37 s

Table 1: *Timings of applying the rolling guidance filter on different models. The number of iterations is 5. The timings of normal filtering (Filter), Cholesky decomposition (Decomp) and solving the vertex positions (VUpdate) are in seconds. $\bar{l}_e$ is the average edge length of each model. We choose appropriate σ_r and σ_s to remove the small-scale features.*

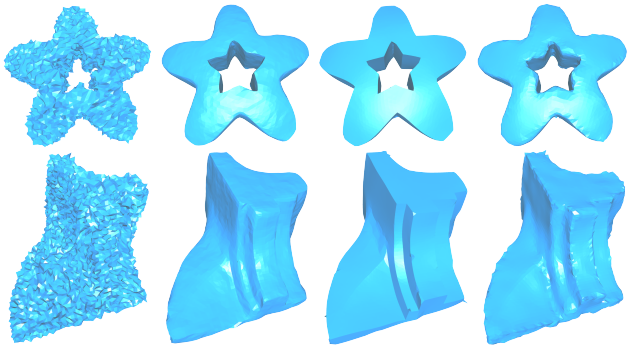


Figure 12: *Noisy mesh denoising. We apply the mean-shift method [Solomon et al. 2014], L_0 smoothing and our RGN filter on two noisy meshes (Noisy models and the mean-shift results are from the project page of the work [Solomon et al. 2014]). The former two methods remove noise and recover the sharp features much better than our RGN.*

normals are extremely large. One possible solution is to integrate face-orientation checking and self-collision detection explicitly in the vertex update step.

In our experiments, the values of σ_s and σ_r at different locations of a surface are set to be constant. In some cases, the geometric features to be removed may be of different scales and using spatially varying parameters may result in more satisfying results. Although the problem can be partially resolved by processing interesting regions separately and interactively, it is desirable to determine appropriate scales and parameters automatically. We leave this for future work.

Another interesting future direction is to account for the color or texture information of surfaces in the rolling guidance filter.

Acknowledgements

We would like to thank Juyong Zhang for providing the Merlion model, Hans-Christian Ebke for providing level-of-detail quad meshing results, Youyi Zheng for sharing the implementation of bilateral normal smoothing, Huayan Zhang for generating mesh segmentation results and the anonymous reviewers for their constructive feedback.

References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian KD-trees for fast high-dimensional filtering. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 21:1–21:12.

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.

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Symp. Geom. Proc.*, 230–239.

AU, O.-C., ZHENG, Y., CHEN, M., XU, P., AND TAI, C.-L. 2012. Mesh segmentation with concavity-aware fields. *IEEE. T. Vis. Comput. Gr. 18*, 7, 1125–1134.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 77:1–77:10.

BOMMES, D., LÉVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2013. Quad-mesh generation and processing: a survey. *Comput. Graph. Forum 32*, 6, 51–76.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE. T. Vis. Comput. Gr. 14*, 1, 213–230.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. A K Peters/CRC Press.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A benchmark for 3D mesh segmentation. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 73:1–73:12.

CHO, H., LEE, H., KANG, H., AND LEE, S. 2014. Bilateral texture filtering. *ACM Trans. Graph. (SIGGRAPH) 33*, 4, 128:1–128:8.

CHUANG, M., AND KAZHDAN, M. 2011. Interactive and anisotropic geometry processing using the screened Poisson equation. *ACM Trans. Graph. (SIGGRAPH) 30*, 4, 57:1–57:10.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *Proceedings of the conference on Visualization'00*, 397–405.

CRANE, K., DE GOES, F., DESBRUN, M., AND SCHRÖDER, P. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses*.

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.

EBKE, H.-C., BOMMES, D., CAMPEN, M., AND KOBBELT, L. 2013. QEx: robust quad mesh extraction. *ACM Trans. Graph. (SIGGRAPH ASIA) 32*, 6, 168:1–168:10.

EBKE, H.-C., CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2014. Level-of-detail quad meshing. *ACM Trans. Graph. (SIGGRAPH ASIA) 33*, 6, 184:1–184:11.

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

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2008. Randomized cuts for 3D mesh analysis. *ACM Trans. Graph. (SIGGRAPH ASIA) 27*, 5, 145:1–145:12.

GUENNEBAUD, G., AND GROSS, M. 2007. Algebraic point set surfaces. *ACM Trans. Graph. (SIGGRAPH) 26*, 3, 23:1–23:9.

GUENNEBAUD, G., JACOB, B., ET AL., 2010. Eigen v3. http://eigen.tuxfamily.org.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *SIGGRAPH*, 325–334.

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

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

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH*, 105–114.

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*.

LÉVY, B., AND BONNEEL, N. 2013. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In *Int. Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds., 349–366.

NADER, G., GUENNEBAUD, G., AND MELLADO, N. 2014. Adaptive multi-scale analysis for point-based surface editing. *Comput. Graph. Forum (PG) 33*, 7, 171–179.

PANOZZO, D., BARAN, I., DIAMANTI, O., AND SORKINE-HORNUNG, O. 2013. Weighted averages on surfaces. *ACM Trans. Graph. (SIGGRAPH) 32*, 4, 60:1–60:12.

PAULY, M., KOBBELT, L., AND GROSS, M. 2006. Point-based multiscale surface representation. *ACM Trans. Graph. 25*, 2, 177–193.

RAY, N., VALLET, B., ALONSO, L., AND LÉVY, B. 2009. Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1, 1:1–1:11.

SILVA, V. D., AND TENENBAUM, J. B. 2002. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, 705–712.

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

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Symp. Geom. Proc.*, 175–184.

SUBR, K., SOLER, C., AND DURAND, F. 2009. Edge-preserving multiscale image decomposition based on local extrema. *ACM Trans. Graph. (SIGGRAPH ASIA) 28*, 5, 147:1–147:9.

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, J., OVSJANIKOV, M., AND GUIBAS, L. 2009. A concise and provably informative multi-scale signature based on heat diffusion. *Comput. Graph. Forum (SGP) 28*, 5, 1383–1392.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings of the conference on Visualization'02*, 125–132.

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

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 839 – 846.

VALLET, B., AND LÉVY, B. 2008. Spectral geometry processing with manifold harmonics. *Comput. Graph. Forum (EG) 27*, 2, 251–260.

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., YAN, Q., XIA, Y., AND JIA, J. 2012. Structure extraction from texture via relative total variation. *ACM Trans. Graph. (SIGGRAPH ASIA) 31*, 6, 139:1–139:10.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Trans. Graph. (SIGGRAPH) 23*, 3, 644–651.

ZHANG, H., VAN KAICK, O., AND DYER, R. 2010. Spectral mesh processing. *Comput. Graph. Forum 29*, 6, 1865–1894.

ZHANG, J., ZHENG, J., WU, C., AND CAI, J. 2012. Variational mesh decomposition. *ACM Trans. Graph. 31*, 3, 21:1–21:14.

ZHANG, Q., SHEN, X., XU, L., AND JIA, J. 2014. Rolling guidance filter. In *ECCV*, 815–830.

ZHANG, H., WU, C., ZHANG, J., AND DENG, J. 2015. Variational mesh denoising using total variation and piecewise constant function space. *IEEE. T. Vis. Comput. Gr. 21*, 7.

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.