Feature-Preserving Surface Mesh Smoothing via Suboptimal Delaunay Triangulation

Zhanheng Gao[a,b], Zeyun Yu[b,*], Michael Holst[c]

[a]College of Computer Science and Technology, Jilin University, China
[b]Department of Computer Science, University of Wisconsin at Milwaukee, USA
[c]Department of Mathematics, University of California, San Diego, USA

Abstract

A method of triangular surface mesh smoothing is presented to improve angle quality by extending the original optimal Delaunay triangulation (ODT) to surface meshes. The mesh quality is improved by solving a quadratic optimization problem that minimizes the approximated interpolation error between a parabolic function and its piecewise linear interpolation defined on the mesh. A suboptimal problem is derived to guarantee a unique, analytic solution that is significantly faster with little loss in accuracy as compared to the optimal one. In addition to the quality-improving capability, the proposed method has been adapted to remove noise while faithfully preserving sharp features such as edges and corners of a mesh. Numerous experiments are included to demonstrate the performance of the method.

Keywords: surface mesh denoising, mesh quality improvement, feature-preserving, optimal Delaunay triangulation

1. Introduction

Triangular surface meshes are widely used in computer graphics, industrial design and scientific computing. In computer graphics and design, people are typically interested in the smoothness (low variation in curvature) and sharp features (edges, corners, etc.) of a mesh. In many applications of scientific computing, however, the quality of a mesh is a key factor that significantly affects the numerical result of finite or boundary element analysis.

*Corresponding author: yuz@uwm.edu

One of the most common criteria for mesh quality is the uniformity of angles, although this may not be the best in some cases where anisotropic meshes are desired [1]. For its popularity, however, we shall adopt the angle-based criterion in the present work. In many real applications, the input meshes often have low quality, containing angles close or even equal to 0° or 180°. The main interest and contribution of the present work is to improve the quality of triangular surface meshes. Additionally our method will be extended to be able to remove noise and preserve sharp features on surface meshes. For simplicity, we refer to both mesh quality improvement and mesh denoising as mesh smoothing unless otherwise specified.

Mesh denoising has a long history in computer graphics and the related methods include three main categories: (1) geometric flows [2, 3, 4, 5, 6], (2) spectral analysis [7, 8], and (3) optimization methods [9, 10]. Due to its simplicity and low computational cost, Laplacian smoothing has established itself as one of the most common methods among all the geometric flow-based methods. In this method, every node is updated towards the barycenter of the neighborhood of the node. However, volume shrinkage often occurs during this process. The shrinkage problem may be tackled by methods utilizing spectral analysis of the mesh signal, which is the main idea of the second category. Optimization-based methods guarantee the smoothness of the mesh by minimizing different types of energy functions. But the iterative process searching for optimal solutions can be time-consuming.

A variety of techniques on mesh quality improvement have been developed [11, 12]. Some of the existing techniques include: (1) inserting/deleting vertices [13], (2) swapping edges/faces [14], (3) remeshing [15, 16, 17, 18], and (4) moving vertices without changing mesh topology [19, 20, 21, 22]. Two or more of the above techniques are sometimes combined to achieve better performance. For instance, Dyer et al. [23] integrate edge flipping, remeshing and decimation into one framework for generating high-quality Delaunay meshes. In the current work, however, we shall restrict ourselves to the methods in the last category that only adjust the nodes' coordinates. Among these methods, Laplacian smoothing in its simplest form that moves a vertex to the center or barycenter of the surrounding vertices [19] is one of the fastest methods but it may fail in improving mesh quality and is often equipped with other techniques such as optimizations [24, 25]. Ohtake et al. [26] presented a method of simultaneously improving and denoising a mesh based on a combination of mean curvature flow and Laplacian smoothing. Nealen et al. [27] introduced a framework for mesh improving and denoising using Laplacian-based least-squares techniques. Both methods, as shown in [28], cannot warrant mesh quality or feature-preservation. Wang et al. [28] presented a method for mesh denoising and quality improvement by local surface fitting and maximum inscribed circles but it was heuristic and lacked mathematical foundations.

Among all the repositioning-based methods for mesh quality improvement, the optimal Delaunay triangulation (ODT) [29, 1, 30] has been proved to be effective on 2D triangular meshes. However, the extension from 2D meshes to 3D surface meshes is nontrivial in both mathematical analysis and algorithm design. For 3D surface meshes we need to consider not only angle quality but also mesh noise that causes bumpiness on surfaces, which was not taken into account in the original ODT method or its variants in tetrahedral mesh smoothing [31, 32]. In addition, sharp surface features must be well preserved during the processes of mesh denoising and quality improvement. There have been extensive studies on feature-preserving surface mesh processing [33, 34, 35, 36, 37, 38]. However, most of the previous work was focused on the mesh denoising problem but only a few dealt with both mesh denoising and quality improvement with feature preservation [28].

The main goal of the present paper is to generalize the 2D ODT idea to 2-manifold surface meshes by formulating the mesh quality improvement as an optimization problem that minimizes the interpolation error between a parabolic function and its piecewise linear interpolation at each vertex of the surface mesh. Unfortunately there is no analytical solution to this optimization problem. To solve the minimization problem faster, we consider a suboptimal problem by simplifying the objective function into a quadratic formula such that an analytical solution can be derived. The proposed suboptimal Delaunay triangulation (or **S-ODT**) is then extended to include two other capabilities: removing mesh noise as well as preserving sharp features on the original meshes. These two goals are achieved by using two standard techniques: curve/surface fitting [39] and local structure tensors [33].

The remainder of this paper is organized as follows. In Section 2, we extend the original ODT method [29, 1] to improve the angle quality of a surface mesh. Several variants of the new algorithm are also introduced to warrant additional desirable properties such as noise removal and feature preservation. Numerous mesh examples are included and comparisons are given in Section 3 to demonstrate the performance of the proposed algorithms, followed by our conclusions in Section 4. Some mathematical details of the algorithms are provided in the Appendices.

2. Method

Like many other mesh smoothing approaches, our method is iterative and vertex-based, meaning that all mesh vertices are repositioned in each iteration and the process is repeated until the mesh quality meets some predefined criteria or a maximum number of iterations is reached. In this section we shall describe three algorithms with the basic one addressing the mesh quality improvement using the proposed sub-optimization formulation and two extended algorithms dealing additionally with the issues of feature preservation and noise removal. For completeness, we shall begin with a brief introduction to Delaunay triangulation and the original ODT method [29]. More details on ODT-based 2D/3D and local/global mesh smoothing algorithms can be found in [1, 30].

2.1. Brief introduction to ODT

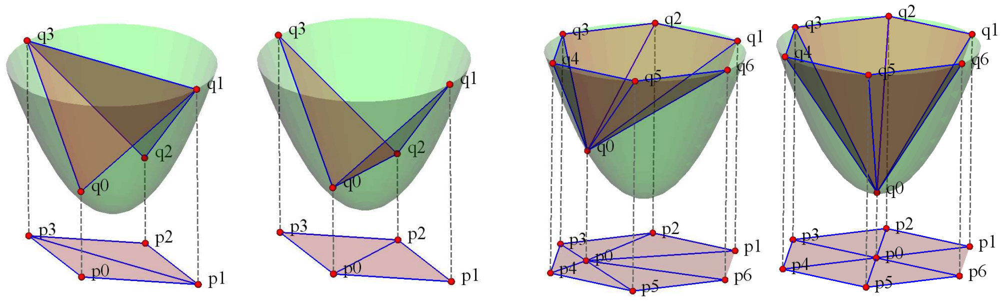
In computational geometry, Delaunay triangulation (DT) is a well known scheme to triangulate a finite set of fixed points P, satisfying the so-called *empty sphere condition*. That is, no point in P can be inside the circumsphere of any simplex (e.g., triangle) in $DT(P)$. Consider, for example, the four points $p0$, $p1$, $p2$ and $p3$ in Figure 1(a-b). There are obviously two ways to triangulate this point set, but only the one in Figure 1(b) is a Delaunay triangulation that produces a larger minimum angle than that in Figure 1(a) and thus is preferable according to the angle-based criterion. Figure 1(a-b) also tells us another interpretation of Delaunay triangulation. If we lift the point set onto a parabolic function $||\mathbf{x}||^2$, any triangulation on the lifting points $q0$, $q1$, $q2$ and $q3$ will result in a unique piecewise linear interpolation of the parabolic function. The one that minimizes the interpolation error can be projected back to the original point set and makes the Delaunay triangulation. From this example, we can see that Delaunay triangulation of a fixed point set is equivalent to minimizing the following interpolation error, which can be achieved by swapping edges:

$$Q(DT, ||\mathbf{x}||^2, q) = \min_{\mathcal{T} \in \mathcal{T}_p} Q(\mathcal{T}, ||\mathbf{x}||^2, q), \quad \forall 1 \leq q \leq \infty, \tag{1}$$

where $Q(\mathcal{T}, ||\mathbf{x}||^2, q)$ is the L^q distance between the parabolic function $||\mathbf{x}||^2$ and its piecewise linear interpolation $||\mathbf{x}||_I^2$ based on a particular triangulation $\mathcal{T}$ of a fixed point set P. $\mathcal{T}_p$ is the set of all possible triangulations of P.

Although Delaunay triangulation is optimal for a fixed set of points, it does not necessarily produce a high quality mesh if the given points are not nicely distributed. In addition to edge-swapping, there is actually another way, called vertex-repositioning, to minimize the error between a parabolic function and its piecewise linear interpolation. Consider for example the point set in Figure 1(c). The triangulation is already optimal in terms of the DT criterion. However, the interpolation error can be further reduced by moving the vertex $p0$ to a better position as shown in Figure 1(d) and hence the mesh quality is improved. This strategy constitutes the core of the optimal Delaunay triangulation (ODT) method as detailed in [1, 29, 30].

It is worth noting that the vertex-repositioning alone does not produce a Delaunay-like triangulation. For better mesh quality improvement, it is always wise to combine edge-swapping into vertex-repositioning, as in the original ODT method [29]. In the rest of the current paper, we shall extend the ODT method to surface meshes to improve the angle quality. However, we will not consider the edge-swapping technique in the descriptions of our algorithms as well as results, simply because our main focus in the current paper is how vertices are repositioned to achieve quality improvement and two other goals (noise removal and feature preservation).



(a) (b) (c) (d)

Figure 1: Illustration of minimizing interpolation error in two ways: edge swapping (a-b) and vertex-repositioning (c-d). The mesh quality in (b) is improved by swapping the edges but keeping the vertices fixed. The interpolation error can also be reduced (hence mesh quality is improved) by moving the vertex $p0$ in (c) to a new position in (d), where the edge connections are kept unchanged.

2.2. Optimal Delaunay triangulation on surfaces

Suppose $\mathcal{M}$ is a triangular surface mesh in $\mathbb{R}^3$ and the sets of vertices (or nodes) and faces are $\mathcal{V}$ and $\mathcal{K}$ respectively. Let $\mathbf{x}_*$ be the optimal position of a vertex $\mathbf{x}_0 \in \mathcal{V}$ in the sense that the following interpolation error is minimized:

$$\begin{aligned} E(\mathbf{x}') =& \int_{\mathbf{x} \in \mathcal{N}'} |f_I(\mathbf{x}-\mathbf{x}') - f(\mathbf{x}-\mathbf{x}')| \, \mathrm{d}\mathbf{x} \\ =& \sum_{k=1}^{N} \int_{\mathbf{x} \in \tau'_k} f_I(\mathbf{x}-\mathbf{x}') - f(\mathbf{x}-\mathbf{x}') \, \mathrm{d}\mathbf{x}, \end{aligned} \tag{2}$$

where $\mathbf{x}'$ is the varying (new) position of $\mathbf{x}_0$, $\mathcal{N}' \subset \mathcal{K}$ is the set of N neighboring triangles around $\mathbf{x}'$, $f(\mathbf{x}) = ||\mathbf{x}||^2$ is a parabolic function in $\mathbb{R}^3$, $f_I(\mathbf{x})$ is the piecewise linear interpolation of $f(\mathbf{x})$ based on $\mathcal{N}'$, and τ'_k is the k-th triangle in $\mathcal{N}'$. Note that $f_I(\mathbf{x})$ is always no less than $f(\mathbf{x})$ so that we can remove the absolute-value operation in the first equation of (2).

The key of minimizing (2) is to compute the sum of the surface integrals in all the neighboring triangles around $\mathbf{x}'$. Suppose τ'_k is formed by $< \mathbf{x}'$, $\mathbf{x}_k$, $\mathbf{x}_{k+1} >$ (let $\mathbf{x}_{N+1} = \mathbf{x}_1$), the integral $\int_{\mathbf{x} \in \tau'_k} f_I(\mathbf{x}-\mathbf{x}') - f(\mathbf{x}-\mathbf{x}')\mathrm{d}\mathbf{x}$ can be computed by replacing $\mathbf{x}$ with $\mathbf{x}' + \lambda_1(\mathbf{x}_k - \mathbf{x}') + \lambda_2(\mathbf{x}_{k+1} - \mathbf{x}')$, where λ_1, $\lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$. Thus (2) becomes the following equation (see Appendix A for details):

$$E(\mathbf{x}') = \sum_{k=1}^{N} [(\mathbf{x}_k - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}_k)^2] \mathrm{S}'_k, \tag{3}$$

where S'_k is the area of τ'_k. Note that S'_k depends on $\mathbf{x}'$ introducing additional non-linearity to the error function.

The minimizer of (3) in general does not admit a closed-form expression. Although numerical methods may be used for solving (3), it can be computationally inefficient, as will be demonstrated in Section 3. For this reason, we shall take another strategy by replacing S'_k with other types of weights, yielding a suboptimal problem that can be analytically and more efficiently solved. The simplest case is that, if we set $\mathrm{S}'_k \equiv 1$ for $k = 1, 2, \cdots, N$, the solution of (3) is equivalent to the Laplacian smoothing that moves $\mathbf{x}'$ towards the barycenter of its neighborhood in $\mathcal{K}$. Therefore, Laplacian smoothing is just a special case of (3).

2.3. Suboptimal Delaunay triangulation on surfaces

In this work, we replace each S'_k in (3) with $\mathrm{D}'_k = \det(\mathbf{x}_k - \mathbf{x}', \mathbf{x}_{k+1} - \mathbf{x}', \mathbf{n})$, where $\mathbf{n}$ is the unit normal vector of a plane Π_t on which $\mathbf{x}'$ is allowed to move. As an approximation to the tangent plane at $\mathbf{x}_0$, Π_t is computed as follows:

$$\mathbf{n} = \frac{\sum_{k=1}^{N} \mathrm{S}_k \mathbf{n}_k}{||\sum_{k=1}^{N} \mathrm{S}_k \mathbf{n}_k||}, \tag{4}$$

where S_k and $\mathbf{n}_k$ are the area and unit normal vector of the k^{th} neighboring triangle of $\mathbf{x}_0$ in the original mesh. As shown in Appendix D, when $\mathbf{x}'$ is restricted to the tangent plane defined this way, the volume of a closed mesh can be exactly preserved. Please note that at this moment, we assume that the original mesh is smooth enough and noise-free, such that the tangent plane is well defined as above. For meshes with sharp features or noise, special care must be taken to calculate tangent planes (or feature lines) as will be discussed in the subsequent subsections. In these cases, the volume preservation is not guaranteed.

Note that D'_k is the area of the projection of τ'_k onto Π_t, the ratio between any two D'_k's is a good approximation of the ratio between the two corresponding S'_k's. With this in mind, we replace each S'_k in (3) with D'_k and have the following approximated, suboptimal Delaunay triangulation (S-ODT) problem:

$$\mathbf{x}_* = \mathrm{argmin}\overline{E}(\mathbf{x}') \text{ with}$$
$$\overline{E}(\mathbf{x}') = \sum_{k=1}^{N} [(\mathbf{x}_k - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}_k)^2]\mathrm{D}'_k. \tag{5}$$

We shall see in Section 3, especially Figure 6, that the approximation of S'_k with D'_k makes sense (i.e., with significantly less computational time but little loss in mesh quality).

As each D'_k also has linear dependence on $\mathbf{x}'$, the error $\overline{E}$ seems to have cubic dependence on $\mathbf{x}'$ and thus the minimizer of (5) does not seem to admit a closed form expression. Fortunately, the sum of all D'_k is a constant (i.e., $\sum_{k=1}^{N} \mathrm{D}'_k \equiv C$; see Appendix B for the proof) if all the neighbors around $\mathbf{x}'$ are fixed (i.e., we smooth the mesh locally). This property makes the sum of all cubic terms in (5) a constant and thus minimizing (5) becomes an unconstrained quadratic optimization problem such that an analytic solution can be obtained.

In order to preserve the local shape (and volume too) of the original mesh near $\mathbf{x}'$, we restrict $\mathbf{x}'$ to moving only in the tangent plane Π_t. Thus $\mathbf{x}'$ in (5) can be written as a parametric representation as follows:

$$\mathbf{x}' = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t}, \tag{6}$$

where $\mathbf{s}$ and $\mathbf{t}$ are two orthogonal unit vectors on Π_t, and u, v are the coordinates of $\mathbf{x}'$ corresponding to $\mathbf{s}$ and $\mathbf{t}$ respectively.

Algorithmically the optimal coordinates u_*, v_* can be computed by solving the following system of linear equations:

$$\begin{pmatrix} 2\mathcal{E} & \mathcal{G} \\ \mathcal{G} & 2\mathcal{F} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \mathcal{H} \\ \mathcal{I} \end{pmatrix}, \tag{7}$$

where $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ are determined in the following way:

$$\begin{cases} \mathcal{E} = C + \sum_{k=1}^{N} [\mathbf{s}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{s}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \\ \mathcal{F} = C + \sum_{k=1}^{N} [\mathbf{t}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{t}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \\ \mathcal{G} = \sum_{k=1}^{N} [\mathbf{s}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{t}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n}) \\ \quad + \mathbf{t}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{s}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \\ \mathcal{H} = \sum_{k=1}^{N} [\mathbf{s}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) \\ \quad + (\mathbf{X}_k^2 + \mathbf{X}_{k+1}^2 - \mathbf{X}_k \mathbf{X}_{k+1}) \det(\mathbf{s}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \\ \mathcal{I} = \sum_{k=1}^{N} [\mathbf{t}(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) \\ \quad + (\mathbf{X}_k^2 + \mathbf{X}_{k+1}^2 - \mathbf{X}_k \mathbf{X}_{k+1}) \det(\mathbf{t}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \end{cases} \tag{8}$$

where $\mathbf{X}_i = \mathbf{x}_i - \mathbf{x}_0$ for $i = 1, 2, \cdots, N$. The details of calculating $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ are provided in Appendix C. The basic S-ODT algorithm for surface mesh quality improvement by minimizing (5) is summarized in Algorithm 1.

2.4. Feature-preserving mesh quality improvement

Algorithm 1 performs well for surface meshes without sharp features such as creases or corners. In reality, however, sharp features are commonly seen and crucial in precisely representing geometric features of a mesh. To this end, we classify the surface nodes into three categories: (1) smooth nodes

Algorithm 1: Suboptimal Delaunay triangulation (S-ODT)

Input: A surface mesh $\mathcal{M}$ with vertices $\mathcal{V}$ and faces $\mathcal{K}$

for every $\mathbf{x}_0$ in $\mathcal{V}$ **do**

  Find all the neighboring nodes $\{\mathbf{x}_k\}$ around $\mathbf{x}_0$

  Compute the unit normal vector $\mathbf{n}$ of Π_t at $\mathbf{x}_0$

  Choose two vectors $\mathbf{s}$ and $\mathbf{t}$ on Π_t

  Compute $\{D_k'\}$ and C, and then $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ in (8)

  Solve the matrix equation in (7)

  Compute the optimal $\mathbf{x}_*$ with $\mathbf{x}_0 + u_*\mathbf{s} + v_*\mathbf{t}$

end for

Output: The smoothed mesh $\mathcal{M}_s$

with low curvature in the neighborhood, (2) crease nodes with low curvature in one direction and high curvature in another (typically perpendicular to the first direction), and (3) corner nodes, where at least three creases intersect. We define crease and corner nodes as feature nodes and impose some special restrictions on them during the mesh smoothing process. Specifically, a crease node moves only along the direction of the crease and a corner node remains unchanged.

Motivated by [33] and [22], we distinguish between smooth and feature nodes by using the local structure tensor $\mathbf{T}$ at $\mathbf{x}_0$ as defined below:

$$\mathbf{T} = \sum_{k=1}^{N} \omega_k \mathbf{n}_k \mathbf{n}_k^{\mathrm{T}}. \tag{9}$$

Here $\mathbf{n}_k$ is the unit normal vector of τ_k, calculated by $< \mathbf{x}_0, \mathbf{x}_k, \mathbf{x}_{k+1} >$. The weight ω_k is determined by $\frac{\mathrm{S}_k}{\mathrm{S}_{\max}} \exp(-g_k/\sigma)$, where S_k is the area of τ_k, $\mathrm{S}_{\max} = \max\limits_{i=1,\cdots,N} \mathrm{S}_i$, g_k is the distance from x_0 to the barycenter of τ_k, and σ is the average edge length of the surface mesh.

Note that $\mathbf{T}$ is a semi-positive definite symmetric matrix and has three real eigenvalues. We decompose $\mathbf{T}$ using the eigen-analysis method and decide the type of $\mathbf{x}_0$ based on the distribution of the eigenvalues of $\mathbf{T}$. Let $\nu_1 \geq \nu_2 \geq \nu_3$ be the eigenvalues of $\mathbf{T}$ and $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ be the corresponding eigenvectors. Let $\mathcal{S}_s = \nu_1 - \nu_2$, $\mathcal{S}_e = \nu_2 - \nu_3$ and $\mathcal{S}_c = \nu_3$, the type of $\mathbf{x}_0$ is determined by the following scheme:

$$\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \begin{cases} \mathcal{S}_s : \mathbf{x}_0 \text{ is a smooth node} \\ \epsilon\mathcal{S}_e : \mathbf{x}_0 \text{ lies on a crease curve} \\ \qquad\quad \text{with direction } \mathbf{e}_3 \\ \epsilon\eta\mathcal{S}_c : \mathbf{x}_0 \text{ is a corner node} \end{cases}. \tag{10}$$

Here, the sensitivity parameters ϵ and η are both set to be 2 according to [33].

In the mesh smoothing process, Algorithm 1 is still applicable when $\mathbf{x}_0$ is a smooth node. When $\mathbf{x}_0$ is a corner node, we just keep it unchanged. When $\mathbf{x}_0$ is a crease node, however, we move $\mathbf{x}_0$ to the optimal position by solving (5) along the direction of the crease. Therefore, we assume $\mathbf{x}' = \mathbf{x}_0 + d\mathbf{e}_3$ and compute the optimal value d_* by minimizing (5) along $\mathbf{e}_3$. The computation of d_* is similar to that of u_*, v_* in Algorithm 1. First, we compute the corresponding coefficients in the following way:

$$\begin{cases} A = C + \sum_{k=1}^{N} [\mathbf{e}_3(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{e}_3, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \\ B = \sum_{k=1}^{N} [\mathbf{e}_3(\mathbf{X}_k + \mathbf{X}_{k+1}) \det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) \\ \qquad + (\mathbf{X}_k^2 - \mathbf{X}_k\mathbf{X}_{k+1} + \mathbf{X}_{k+1}^2) \det(\mathbf{e}_3, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})] \end{cases} \tag{11}$$

Then the scalar d_* is computed by $d_* = -\frac{B}{2A}$. The process is summarized in Algorithm 2.

Algorithm 2: Feature-preserving S-ODT

Input: A surface mesh $\mathcal{M}$ with vertices $\mathcal{V}$ and faces $\mathcal{K}$

for every $\mathbf{x}_0$ in $\mathcal{V}$ **do**
    Find all the neighboring nodes $\{\mathbf{x}_k\}$ around $\mathbf{x}_0$
    Compute the unit normal vector $\mathbf{n}$ of Π_t at $\mathbf{x}_0$
    Compute the tensor matrix $\mathbf{T}$ using (9)
    Compute the eigen-pairs of $\mathbf{T}$: ν_1, $\mathbf{e}_1$, ν_2, $\mathbf{e}_2$, ν_3, $\mathbf{e}_3$
    Set $\mathcal{S}_s = \nu_1 - \nu_2$, $\mathcal{S}_e = \nu_2 - \nu_3$, $\mathcal{S}_c = \nu_3$
    if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \mathcal{S}_s$ **do**
        Set $\mathbf{x}_0$ as a smooth node
    else if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \epsilon\mathcal{S}_e$, **do**
        Set $\mathbf{x}_0$ as a crease node
    else if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \epsilon\eta\mathcal{S}_c$ **do**

Set $\mathbf{x}_0$ as a corner node
end if
if $\mathbf{x}_0$ is a corner node **do**
continue
else if $\mathbf{x}_0$ is a smooth node **do**
Choose two vectors $\mathbf{s}$ and $\mathbf{t}$ on Π_t
Compute $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ in (8) and solve (7)
Compute the optimal $\mathbf{x}_*$ with $\mathbf{x}_0 + u_*\mathbf{s} + v_*\mathbf{t}$
else if $\mathbf{x}_0$ is a crease node **do**
Compute A, B in (11) and set $d_* = -\frac{B}{2A}$
Compute the optimal $\mathbf{x}_*$ with $\mathbf{x}_0 + d_*\mathbf{e}_3$
end if
end for
Output: The smoothed mesh $\mathcal{M}_s$

2.5. Feature-preserving, noise-removing mesh quality improvement

Our method can be readily adapted to remove mesh noise while improving mesh quality and still retaining the feature-preserving property. In the basic S-ODT algorithm (Algorithm 1), the optimal position is assumed to be on the tangent plane at $\mathbf{x}_0$ of the surface mesh. When there is noise on the surface mesh, a common strategy is to fit a plane (or higher order polynomials) to the neighboring nodes of each vertex and project the vertex onto the plane [28]. Sharp features may be preserved by considering anisotropic local neighborhoods [35]. In our current work, we utilize a weighted least squares fitting strategy as detailed below [39].

As described in Section 2.4, each mesh node can be classified into either a smooth node, a crease node or a corner node. We always keep the corner nodes unchanged. Suppose $\mathbf{x}_0$ is a smooth node with neighboring nodes $\{\mathbf{x}_k\}_{k=1}^N$. The corresponding unit normal vectors at $\mathbf{x}_0$ and its neighbors are $\{\mathbf{n}_k\}_{k=0}^N$. Then a plane can be fitted by solving the following weighted least squares problem:

$$\min_{\bar{\mathbf{x}},\bar{\mathbf{n}}} \sum_{k=0}^{N} w_k((\mathbf{x}_k - \bar{\mathbf{x}})\bar{\mathbf{n}})^2 \tag{12}$$

where w_k is the weight of $\mathbf{x}_k$, $\bar{\mathbf{x}}$ is a point on the fitting plane Π_f and $\bar{\mathbf{n}}$ is the unit normal vector of Π_f. The weights are set as follows:

$$w_0 = 1, w_k = L(r_k), \text{where } r_k = \mathbf{n}_0 \cdot \mathbf{n}_k, k = 1, 2, \cdots, N.$$

$L(r)$ is a linear function on $[\cos(\pi/4), 1]$ with $L(\cos(\pi/4)) = 0$ and $L(1) = 1$.

The fitting plane Π_f can be computed by first determining $\bar{\mathbf{x}}$ and then $\bar{\mathbf{n}}$. Specifically, $\bar{\mathbf{x}}$ is the weighted average of $\mathbf{x}_0$ and its neighbors:

$$\bar{\mathbf{x}} = \sum_{k=0}^{N} w_k \mathbf{x}_k / \sum_{k=0}^{N} w_k. \tag{13}$$

$\bar{\mathbf{n}}$ is chosen to be the eigenvector corresponding to the smallest eigenvalue of the following matrix $\mathbf{M}$ [39]:

$$\mathbf{M} = \sum_{k=0}^{N} w_k (\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{x}_k - \bar{\mathbf{x}})^{\mathrm{T}}.$$

Simply projecting $\mathbf{x}_0$ onto the fitting plane Π_f can suppress the mesh noise around $\mathbf{x}_0$ but the mesh angle quality may not be improved and sometimes may become even worse. To achieve both mesh denoising and quality improvement, we replace the tangent plane Π_t in Algorithm 1 or Algorithm 2 with the fitting plane Π_f and accordingly replace $\mathbf{x}_0$ with $\bar{\mathbf{x}}$ in the minimization of (5).

When $\mathbf{x}_0$ is a crease node, a similar procedure is applied. The difference is that we fit a line instead of a plane by using some 2-ring neighboring nodes of $\mathbf{x}_0$ and then project $\mathbf{x}_0$ onto the fitted line. The neighboring nodes selected include $\mathbf{x}_0$ itself, two neighbors along one direction of the crease line and two neighbors along the other direction of the crease line, where the crease line passing $\mathbf{x}_0$ is defined as the crease direction determined by the tensor analysis procedure. The two neighbors along each direction are selected so that they are the closest to the crease line. The overall algorithm for feature-preserving mesh denoising and quality improvement is given in Algorithm 3.

Algorithm 3: Feature-preserving & noise-removing S-ODT

Input: A surface mesh $\mathcal{M}$ with vertices $\mathcal{V}$ and faces $\mathcal{K}$
for every node $\mathbf{x}_0$ in $\mathcal{V}$ **do**
  Find all the neighboring nodes $\{\mathbf{x}_k\}$ around $\mathbf{x}_0$
  Compute the normal tensor $\mathbf{T}$ using (9)
  Compute the eigen-pairs of $\mathbf{T}$: ν_1, $\mathbf{e}_1$, ν_2, $\mathbf{e}_2$, ν_3, $\mathbf{e}_3$
  Set $\mathcal{S}_s = \nu_1 - \nu_2$, $\mathcal{S}_e = \nu_2 - \nu_3$, $\mathcal{S}_c = \nu_3$
  if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \mathcal{S}_s$ **do**
    Set $\mathbf{x}_0$ as a smooth node

else if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \epsilon\mathcal{S}_e$, **do**
    Set $\mathbf{x}_0$ as a crease node
else if $\max\{\mathcal{S}_s, \epsilon\mathcal{S}_e, \epsilon\eta\mathcal{S}_c\} = \epsilon\eta\mathcal{S}_c$ **do**
    Set $\mathbf{x}_0$ as a corner node
end if
if $\mathbf{x}_0$ is a corner node **do**
    continue
else if $\mathbf{x}_0$ is a smooth node **do**
    Compute $\bar{\mathbf{x}}$ and $\bar{\mathbf{n}}$ for the fitting plane Π_f
    Set $\mathbf{x}_0 = \bar{\mathbf{x}}$ and $\mathbf{n} = \bar{\mathbf{n}}$
    Compute $\mathbf{s}$ and $\mathbf{t}$ which are perpendicular to $\mathbf{n}$
    Compute the $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, $\mathcal{I}$ in (8) and solve (7)
    Compute the optimal $\mathbf{x}_*$ with $\mathbf{x}_0 + u_*\mathbf{s} + v_*\mathbf{t}$
else if $\mathbf{x}_0$ is a crease node **do**
    Find four more neighboring nodes along $\mathbf{e}_3$ near $\mathbf{x}_0$
    Fit a line based on these five nodes
    Set $\mathbf{x}_0$ to be any point on the fitting line
    Compute A, B in (11) and set $d_* = -\frac{B}{2A}$
    Compute the optimal $\mathbf{x}_*$ with $\mathbf{x}_0 + d_*\mathbf{e}_3$
end if
end for
Output: The smoothed mesh $\mathcal{M}_s$

3. Results and Discussions

The presented algorithms have been tested on numerous surface meshes and we shall show some of the results below. We first apply the basic S-ODT algorithm (Algorithm 1) to several surface meshes without noise or sharp features. The bimba and elephant models are shown in Figure 2(a) and Figure 3(a) respectively. A closer look at the original bimba mesh and the angle histograms of these meshes are given in Figure 2(b-c) and Figure 3(b-c). By applying Algorithm 1 to each mesh for 20 times, the mesh qualities are significantly improved, as can be seen from Figure 2(d-f) and Figure 3(d-f).

In Figure 4 we show how the minimum and maximum angles of the bimba and elephant models change with respect to the number of iterations by applying Algorithm 1. We can see that the mesh qualities are largely improved in the first five iterations, and further smoothing does not help much on mesh quality improvement. In order to measure the shape change between

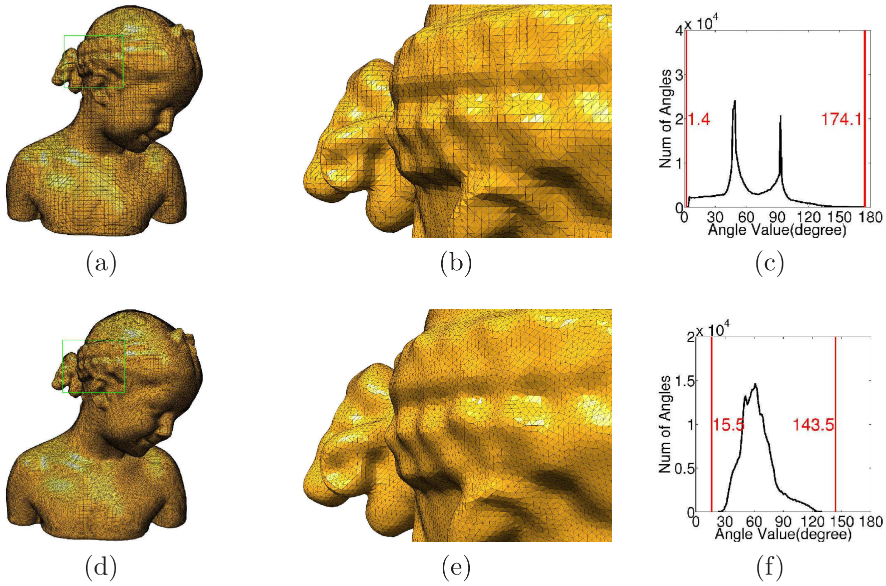


Figure 2: The bimba mesh model. (a-c) show the original surface mesh, a closer view and the angle histogram of the mesh. (d-f) show the smoothed mesh and the corresponding histogram after applying the S-ODT (Algorithm 1) 20 times. The minimum and maximum angles in both meshes are indicated in red in the histograms. The original model is provided courtesy of IMATI and INRIA by the AIM@SHAPE Shape Repository.

the original and smoothed meshes, we compute the symmetric Hausdorff distance between the meshes using the M.E.S.H. tool [40], and the results are illustrated in Figure 5. The histograms in Figure 5 show the absolute differences between the original and smoothed meshes. The maximal relative differences, defined as the ratio of the maximal absolute difference over the diagonal of the bounding box of a mesh, are 0.09% and 0.13% for the *bimba* and *elephant* models respectively.

As mentioned in Section 2.2, the analytically-based S-ODT algorithm is a suboptimal solution to the ODT method on surfaces. Here we compare the S-ODT method with the numerical solution of the optimal problem (the ODT method on surfaces). The model we use is a triangular surface mesh of a biomedical molecule called RyR with 129K vertices. Algorithm 1 is applied to this mesh for 20 times and it takes about 36 seconds. By contrast, a numerical method by using L-BFGS [41] is adopted to solve the original

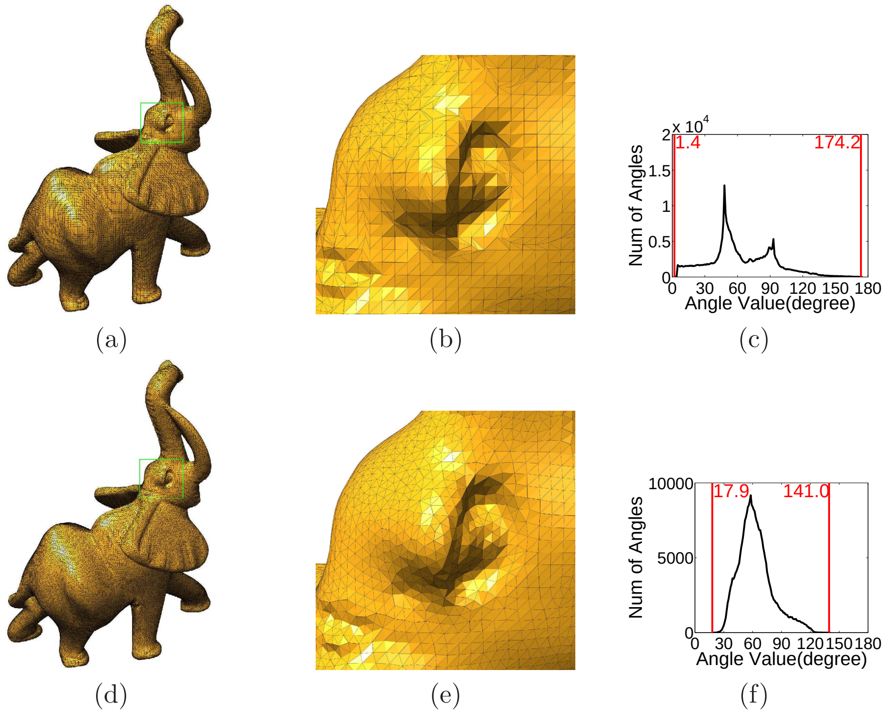
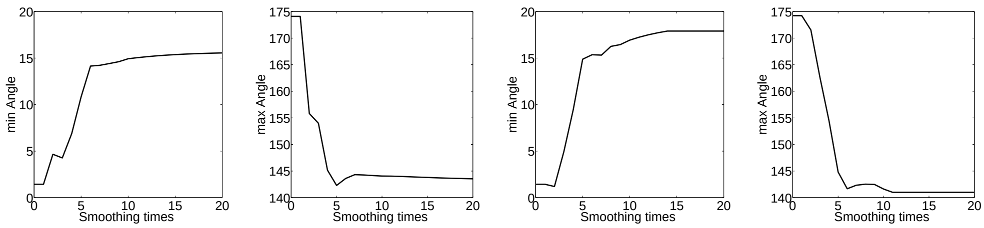


Figure 3: The elephant mesh model. (a-c) show the original surface mesh, a closer view and the angle histogram of the mesh. (d-f) show the smoothed mesh and the corresponding histogram after applying the S-ODT (Algorithm 1) 20 times. The minimum and maximum angles in both meshes are indicated in red in the histograms. The model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository.

optimal problem in Equation (3) and it takes about 2 minutes for 5 iterations, after which no significant improvement was observed. The resulting meshes as well as their qualities, however, are similar by using the two methods, as we can see from Figure 6. A conclusion from this experiment is that the proposed S-ODT method can smooth a mesh as effectively as the numerically-based optimalization method but it takes much less time than the latter.

The feature-preserving S-ODT method (Algorithm 2) is tested on the noise-free fandisk model containing crease edges and corners. The algorithm is applied on the mesh for 20 times and the results are shown in Figure 7(a-f), where both angle histograms and curvature distributions before and after mesh smoothing are provided. Besides the significantly improved mesh

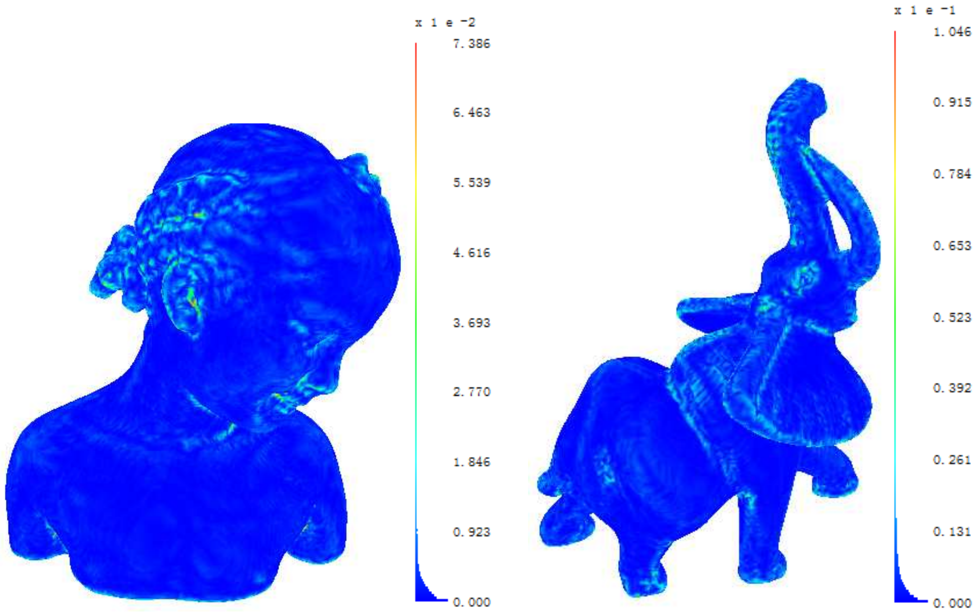


(a) (b) (c) (d)

Figure 4: The convergence of Algorithm 1. (a-b) show how the minimal and maximal angles (in degrees) change with respect to the number of iterations applied to the bimba model. (c-d) show how the minimal and maximal angles (in degrees) change with respect to the number of iterations applied to the elephant model.

quality, the sharp features are well preserved too. Note that the sensitivity parameters ϵ and η in Eq. (10) are both set to be 2 according to [33]. However, we believe that this value should be controlled by the user as the user is the best person to define the "noise" and "feature" in his/her data. Different parameter values would give different results of the node classification, which would produce different smoothing results as the three types of nodes (smooth, crease, and corner) are subject to different smoothing procedures in our algorithms. For example, larger sensitivity parameters would better preserve small "features" that may otherwise be treated as "noise" and smoothed out when small parameters are chosen. When both sensitivity parameter are small, almost all vertices are smooth nodes and no feature preservation can be achieved.

The feature-preserving and noise-removing S-ODT method (Algorithm 3) is also applied for 20 times to three noisy meshes: the dragon head (Figure 8), the Chinese lion (Figure 9), and the noisy fandisk (Figure 10). The mesh quality improvement is clearly demonstrated by the angle histograms in all these models and Figure 11. The curvature distribution maps in Figures 9 and 10 show high-performance mesh denoising effects. In addition, the noisy fandisk model (Figure 10) confirms the feature-preserving property of our method. The bilateral filtering denoising technique is utilized and compared with our approach as shown in Figure 10. In the figure, the mesh quality by the bilateral filtering is poor, and the curvature distribution is also worse than our method. It is worth pointing out that our method performs better



(a) (b)

Figure 5: The symmetric Hausdorff distance between the original and smoothed meshes. The distance is computed using the M.E.S.H. tool [40].

than the bilateral filter because we pre-classify every vertex using the tensor analysis technique.

Tables 3 and 4 show quantitative comparisons between our S-ODT algorithms (with 20 iterations) and two other representative methods, Sun's [35] and Ohtake's [26], running on a Pentium IV PC of 2.0 GHz. Note that the dancer model (not shown due to space limit), also downloaded from the AIM@SHAPE Shape Repository, was included to fill the size gap of the other models shown in this paper. The mesh qualities after smoothing are provided in Tables 3, where Sun's method is excluded because it performs worse than Ohtake's method on all the models considered. From Table 4 we can see that Sun's method is fast but, like Ohtake's, it lacks the ability of mesh quality improvement. While the running time of our method can be much reduced if only $5 \sim 10$ iterations are applied, the biggest gain of our approach is the tremendously improved mesh quality. As shown in Figure 12, the running time of the three variants of our algorithm are approximately linear to the number of vertices in the meshes.

Table 3: Comparisons of min-angle improvement

models	original	ours	Ohtake's
dancer	$0.8°$	$18.3°$ (Alg.1)	$0.4°$
elephant	$1.4°$	$17.9°$ (Alg.1)	$0.2°$
bimba	$1.4°$	$15.5°$ (Alg.1)	$0.2°$
RyR	$4.9°$	$17.3°$ (Alg.1)	$0.2°$
noise-free fandisk	$0.0°$	$17.7°$ (Alg.2)	$0.1°$
noisy fandisk	$16.0°$	$18.4°$ (Alg.3)	$0.4°$
Chinese lion	$0.2°$	$16.8°$ (Alg.3)	$0.0°$
dragon head	$0.3°$	$17.5°$ (Alg.3)	$0.1°$
venus	$1.0°$	$16.8°$ (Alg.3)	$0.0°$
angel	$0.1°$	$16.3°$ (Alg.3)	$0.0°$

Table 4: Comparisons of running time (in seconds)

models	vertex number	ours	Ohtake's	Sun's
dancer	24,998	12.0 (Alg.1)	3.3	0.8
elephant	52,099	16.6 (Alg.1)	7.6	1.0
bimba	83,887	24.5 (Alg.1)	11.8	1.7
RyR	129,346	35.8 (Alg.1)	16.8	2.6
noise-free fandisk	6,475	4.5 (Alg.2)	2.0	0.1
noisy fandisk	6,475	6.3 (Alg.3)	3.0	0.1
Chinese lion	99,289	49.8 (Alg.3)	13.0	2.1
dragon head	99,777	50.5 (Alg.3)	14.8	3.1
venus	100,759	53.3 (Alg.3)	15.7	8.8
angel	236,979	120.4 (Alg.3)	35.1	15.7

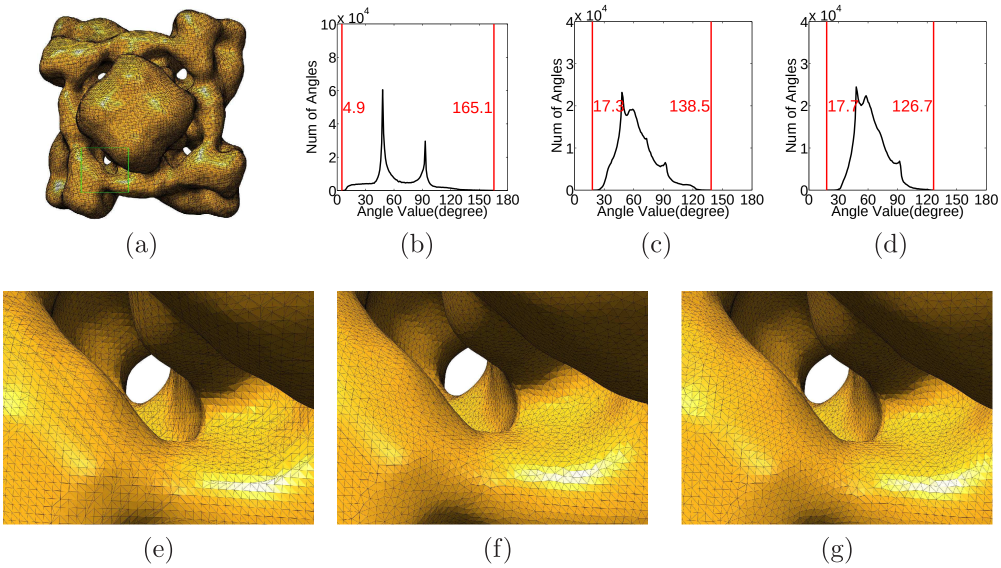


Figure 6: Performance comparison between the analytical solution to the suboptimal problem (the proposed S-ODT method: Algorithm 1) and the numerical solution to the optimal problem (Equation (3)). (a) The original RyR mesh. (b-d) show respectively the angle histograms of the original mesh, the mesh smoothed by the S-ODT method, and the mesh smoothed by the numerically-based ODT method. (e-g) show a closer look at the three meshes respectively. While little difference is observed between the two smoothed meshes, the computational time is only about 36 seconds by using the analytical method for 20 iterations, in contrast to 1 minute and 56 seconds by using the L-BFGS method for 5 iterations.

Finally we would like to compare our S-ODT method with the surface remeshing technique [15, 16, 17, 18], as both aim to generate meshes with high quality. There are two main differences between the two methods: (1) the S-ODT always keeps the connectivity between vertices in a mesh while the remeshing method does not because of the re-sampling on the mesh; (2) for a smooth, closed surface mesh, the S-ODT algorithm preserves exactly the volume of the original mesh while the remeshing method typically does not. In addition, we demonstrate in Table 5 some quantitative comparisons between our S-ODT method and several recent remeshing algorithms (Valette [16], Wang [17] and Fuhrmann [18]). From the table we can see that

Table 5: Comparisons of min-angle improvement between our method and remeshing techniques

model	ours	Valette's	Wang's	Fuhrmann's
dancer	18.3° (Alg.1)	6.0°	20.9°	30.7°
elephant	17.9° (Alg.1)	0.0°	14.9°	30.5°
bimba	15.5° (Alg.1)	0.0°	21.55°	32.8°
RyR	17.3° (Alg.1)	0.2°	N/A	31.0°
noise-free fandisk	17.7° (Alg.2)	0.0°	28.8°	0.0°
noisy fandisk	18.4° (Alg.3)	0.0°	35.6°	34.0°
Chinese lion	16.8° (Alg.3)	0.0°	15.12°	4.23°
dragon head	17.5° (Alg.3)	0.0°	33.11°	0.55°
venus	16.8° (Alg.3)	0.0°	N/A	2.77°
angel	16.3° (Alg.3)	0.0°	N/A	0.92°

the methods in [16] does not guarantee improvement of min angles. The method in [17] fails when the size of the input mesh is too large (e.g., RyR). In addition, the remeshed results by this method are not as smooth as ours, as shown in Figure 8(g-i). Although high-quality meshes generally can be achieved by using the method in [18] when the original meshes are noise-free and error-free, the quality is not guaranteed for noisy meshes. When the original mesh contains self-intersecting triangles (e.g., the noise-free fandisk model in Table 5), the method in [18] cannot fix the errors and often results in low-quality meshes. The two problems of [18] are further demonstrated in Figure 7(g-i) and Figure 9(g-i) respectively. Although our algorithms seem to perform better in dealing with self-intersections, there is no guarantee of mesh quality either. This is because self-intersections introduce inverted normal vectors to some triangles, which usually results in inaccurate estimation of tangent planes (see Eq. (4)). In some cases, our algorithms may fail in improving the minimal and maximal angles of some meshes (see Figure 13 for example), where poorly-shaped triangles are formed by vertices mostly lying on sharp edges. In these cases, other methods such as remeshing [17] or vertex insertion/deletion may work better.

4. Conclusion

In this paper, we present a novel, analytical approach that shows excellent performance in simultaneously denoising a surface mesh, improving the mesh quality, and preserving sharp features. Although the proposed S-ODT method is a suboptimal solution to the original ODT formulation, it can generate comparable results to the latter one but with much less computational time. Our method has fast convergence: typically 5 iterations are sufficient to observe good mesh quality and smoothness. In addition, the symmetric Hausdorff distances show that the smoothed mesh undergoes little shape deformation from the original mesh.

Acknowledgment

The work described was supported in part by an NIH Award (Number R15HL103497) from the National Heart, Lung, and Blood Institute (NHLBI) and by a subcontract from the National Biomedical Computation Resource (NIH Award Number P41 RR08605). We are grateful to Dr. Fuhrmann, Dr. Ohtake, Dr. Sun, and Dr. Valette for making their source code available, and to Dr. Wang who helped generate the remeshed models using their algorithms for comparisons.

Appendix A. From (2) to (3)

For any given $\mathbf{x}'$, note that τ'_k is the triangle formed by $< \mathbf{x}', \mathbf{x}_k, \mathbf{x}_{k+1} >$. We compute

$$\int_{\mathbf{x}\in\tau'_k} f_I(\mathbf{x}-\mathbf{x}') - f(\mathbf{x}-\mathbf{x}')\mathrm{d}\mathbf{x} \tag{A.1}$$

by replacing $\mathbf{x}$ with $\mathbf{x}' + \lambda_1(\mathbf{x}_k - \mathbf{x}') + \lambda_2(\mathbf{x}_{k+1} - \mathbf{x}')$, where λ_1, $\lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$. Let $\mathbf{Y}_k = \mathbf{x}_k - \mathbf{x}'$ and $\mathbf{Y}_{k+1} = \mathbf{x}_{k+1} - \mathbf{x}'$, we can rewrite $f(\mathbf{x}-\mathbf{x}')$ into the following form:

$$f(\mathbf{x}-\mathbf{x}') = \lambda_1^2\mathbf{Y}_k^2 + 2\lambda_1\lambda_2\mathbf{Y}_k\mathbf{Y}_{k+1} + \lambda_2^2\mathbf{Y}_{k+1}^2 \tag{A.2}$$

Note that $f_I(\mathbf{x}-\mathbf{x}')$ is the linear interpolation of $f(\mathbf{x}-\mathbf{x}')$ in τ'_k, $f(\mathbf{x}-\mathbf{x}')$ takes the following form:

$$\begin{aligned} f_I(\mathbf{x}-\mathbf{x}') &= f(\mathbf{0}) + \lambda_1 f(\mathbf{Y}_k) + \lambda_2 f(\mathbf{Y}_{k+1}) \\ &= \lambda_1\mathbf{Y}_k^2 + \lambda_2\mathbf{Y}_{k+1}^2 \end{aligned} \tag{A.3}$$

By substituting (A.2) and (A.3) for $f(\mathbf{x}-\mathbf{x}')$ and $f_I(\mathbf{x}-\mathbf{x}')$ respectively in (A.1), we have

$$\begin{aligned}
&\int\limits_{\mathbf{x}\in\tau'_k} f_I(\mathbf{x}-\mathbf{x}') - f(\mathbf{x}-\mathbf{x}')\mathrm{d}\mathbf{x} \\
=\ & \int_0^1 \mathrm{d}\lambda_1 \int_0^{1-\lambda_1} [(\lambda_1-\lambda_1^2)\mathbf{Y}_k^2+(\lambda_2-\lambda_2^2)\mathbf{Y}_{k+1}^2 \\
& -2\lambda_1\lambda_2\mathbf{Y}_k\mathbf{Y}_{k+1}\mathrm{d}\lambda_2]||\mathbf{Y}_k \times \mathbf{Y}_{k+1}|| \\
=\ & \tfrac{1}{12}(\mathbf{x}_k^2+\mathbf{x}_{k+1}^2+\mathbf{x}'^2-\mathbf{x}'\mathbf{x}_k-\mathbf{x}'\mathbf{x}_{k+1}-\mathbf{x}_k\mathbf{x}_{k+1})\mathrm{S}'_k \\
=\ & \tfrac{1}{24}[(\mathbf{x}_k-\mathbf{x}')^2+(\mathbf{x}_{k+1}-\mathbf{x}')^2+(\mathbf{x}_{k+1}-\mathbf{x}_k)^2]\mathrm{S}'_k
\end{aligned} \tag{A.4}$$

where $\mathrm{S}'_k = ||\mathbf{Y}_k \times \mathbf{Y}_{k+1}||/2$ is the area of τ'_k and depends on the current vertex $\mathbf{x}'$.

By dropping the constant that does not affect the optimal solution, we can rewrite the error function in (2) as follows:

$$E(\mathbf{x}')=\sum_{k=1}^{N}[(\mathbf{x}_k-\mathbf{x}')^2+(\mathbf{x}_{k+1}-\mathbf{x}')^2+(\mathbf{x}_{k+1}-\mathbf{x}_k)^2]\mathrm{S}'_k$$

Appendix B. Proof of $\sum\limits_{k=1}^{N} \mathrm{D}'_k \equiv C$

The determinant D'_k in (5) has another form:

$$\mathrm{D}'_k=\det(\mathbf{x}_k-\mathbf{x}', \mathbf{x}_{k+1}-\mathbf{x}', \mathbf{n})=[(\mathbf{x}_k-\mathbf{x}')\times(\mathbf{x}_{k+1}-\mathbf{x}')]\mathbf{n}$$

Thus we have

$$\begin{aligned}
\sum_{k=1}^{N}\mathrm{D}'_k &= \sum_{k=1}^{N}[(\mathbf{x}_k-\mathbf{x}')\times(\mathbf{x}_{k+1}-\mathbf{x}')]\mathbf{n} \\
&= \sum_{k=1}^{N}[(\mathbf{x}_k\times\mathbf{x}_{k+1})+(\mathbf{x}_{k+1}-\mathbf{x}_k)\times\mathbf{x}']\mathbf{n} \\
&= \mathbf{n}\sum_{k=1}^{N}(\mathbf{x}_k\times\mathbf{x}_{k+1})+\mathbf{n}\left[\sum_{k=1}^{N}(\mathbf{x}_{k+1}-\mathbf{x}_k)\times\mathbf{x}'\right]
\end{aligned}$$

Note that $\sum\limits_{k=1}^{N}(\mathbf{x}_{k+1}-\mathbf{x}_k)=\mathbf{0}$, thus the sum of all D'_k is a constant.

Appendix C. Computing the coefficients in (8)

Note that $\mathbf{x}' = \mathbf{x}_0 + u\mathbf{s} + v\mathbf{t}$, the objective function in (5) is equivalent to:

$$\begin{aligned}\overline{E}(\mathbf{x}') &= \sum_{k=1}^{N}[(\mathbf{x}_k - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}')^2 + (\mathbf{x}_{k+1} - \mathbf{x}_k)^2]\mathrm{D}'_k \\ &= \sum_{k=1}^{N}[(\mathbf{X}_k - \mathbf{X}')^2 + (\mathbf{X}_{k+1} - \mathbf{X}')^2 + (\mathbf{X}_{k+1} - \mathbf{X}_k)^2]\mathcal{D}'_k\end{aligned}$$

where $\mathbf{X}_i = \mathbf{x}_i - \mathbf{x}_0$, $\mathbf{X}' = \mathbf{x}' - \mathbf{x}_0 = u\mathbf{s} + v\mathbf{t}$ and $\mathcal{D}'_k = \mathrm{D}'_k = \det(\mathbf{X}_k - \mathbf{X}', \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n}) = \det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) - \det(\mathbf{X}', \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})$. Let $\mathcal{S}_k$ denote $\mathbf{X}_k^2 - \mathbf{X}_k\mathbf{X}_{k+1} + \mathbf{X}_{k+1}^2$, we have:

$$\begin{aligned}&\overline{E}(\mathbf{x}') \\ &= 2\sum_{k=1}^{N}[\mathbf{X}'^2 - (\mathbf{X}_k + \mathbf{X}_{k+1})\mathbf{X}' + (\mathbf{X}_k^2 - \mathbf{X}_k\mathbf{X}_{k+1} + \mathbf{X}_{k+1}^2)]\mathcal{D}'_k \\ &= 2\{C\mathbf{X}'^2 - \sum_{k=1}^{N}[(\mathbf{X}_k + \mathbf{X}_{k+1})\mathbf{X}'\mathcal{D}'_k + \mathcal{S}_k\mathcal{D}'_k]\} \\ &= 2\{C(u^2 + v^2) - (u\mathbf{s} + v\mathbf{t})\sum_{k=1}^{N}(\mathbf{X}_k + \mathbf{X}_{k+1})\det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) \\ &\quad + (u\mathbf{s} + v\mathbf{t})\sum_{k=1}^{N}(\mathbf{X}_k + \mathbf{X}_{k+1})\det(u\mathbf{s} + v\mathbf{t}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n}) \\ &\quad + \sum_{k=1}^{N}\mathcal{S}_k[\det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n}) - \det(u\mathbf{s} + v\mathbf{t}, \mathbf{X}_{k+1} - \mathbf{X}_k, \mathbf{n})]\} \\ &= 2(\mathcal{E}u^2 + \mathcal{F}v^2 + \mathcal{G}uv - \mathcal{H}u - \mathcal{I}v + \mathcal{J})\end{aligned} \tag{C.1}$$

where $\mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \mathcal{I}$ take the same forms as in (8) and $\mathcal{J} = \sum_{k=1}^{N}\mathcal{S}_k\det(\mathbf{X}_k, \mathbf{X}_{k+1}, \mathbf{n})$.

Note that (C.1) is a quadratic function, it has a unique minimum if the Hessian matrix is positive definite:

$$\begin{aligned}\mathcal{E} &> 0 \\ 4\mathcal{E}\mathcal{F} &> \mathcal{G}^2\end{aligned}$$

In the implementation of our algorithms, these conditions were checked, but interestingly they were never violated on all the meshes we had tested.

Thus the optimal solution of (5) can be computed by solving the following linear system:

$$\begin{pmatrix} 2\mathcal{E} & \mathcal{G} \\ \mathcal{G} & 2\mathcal{F} \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \mathcal{H} \\ \mathcal{I} \end{pmatrix}$$

Appendix D. Proof of the volume-preserving property

We shall prove that the constraint of moving the vertex $\mathbf{x}_0$ on a specially-defined tangent plane can preserve the volume of a smooth, closed surface mesh. In our algorithms, the normal of the tangent plane at $\mathbf{x}_0$ is defined as:

$$\mathbf{n} = \sum_{i=1}^{N} \mathrm{S}_i \mathbf{n}_i, \tag{D.1}$$

where S_i and $\mathbf{n}_i$ are the area and unit normal vector of the incident triangle τ_i formed by $\{\mathbf{x}_0, \mathbf{x}_i, \mathbf{x}_{i+1}\}$. Suppose all $\mathbf{n}_i$'s point to the outside of the closed mesh.

In order to define a "local" volume around $\mathbf{x}_0$ for the surface mesh, we need to have an "anchor" point $\mathbf{y}$, which can be any point. For simplicity, we can choose $\mathbf{y}$ as the centroid of all the neighboring vertices of $\mathbf{x}_0$. By connecting $\mathbf{x}_0$ and $\mathbf{y}$ with all the neighboring vertices of $\mathbf{x}_0$, we get a local, closed domain denoted by Ω. Using a similar idea, when $\mathbf{x}_0$ moves to any new position $\mathbf{x}'$ in the tangent plane defined by (D.1), the points $\mathbf{x}'$, $\mathbf{y}$ and all neighboring vertices of $\mathbf{x}_0$ form another local closed domain Ω'. We shall prove that $|\Omega| \equiv |\Omega'|$ for any $\mathbf{x}'$ in the tangent plane, where $||$ denotes the volume of a closed domain. Note that both Ω and Ω' can be divided into N tetrahedra. For example, the N tetrahedra forming Ω are $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{y}\}$, $\{\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3, \mathbf{y}\}$, $\cdots$, $\{\mathbf{x}_0, \mathbf{x}_N, \mathbf{x}_1, \mathbf{y}\}$. Therefore, the volumes of Ω and Ω' are the total volumes of the tetrahedra forming Ω and Ω' respectively.

We now prove that the volume of Ω' is independent of $\mathbf{x}'$ (or equivalently $|\Omega'| \equiv |\Omega|$). For any tetrahedron σ_k in Ω', the volume $|\sigma_k| = \frac{1}{3}\mathrm{S}_k < \mathbf{n}_k, \mathbf{x}' - \mathbf{y} >$. Thus, we have the following formula for the volume of Ω':

$$\begin{aligned} |\Omega'| &= \sum_{k=1}^{N} \left(\frac{1}{3}\mathrm{S}_k < \mathbf{n}_k, \mathbf{x}' - \mathbf{y} > \right) & \text{(D.2)} \\ &= < \frac{1}{3}\mathbf{n}, \mathbf{x}' - \mathbf{y} > & \text{(D.3)} \\ &= < \frac{1}{3}\mathbf{n}, \mathbf{x}' - \mathbf{x}_0 > + < \frac{1}{3}\mathbf{n}, \mathbf{x}_0 - \mathbf{y} > . & \text{(D.4)} \end{aligned}$$

Note that $\mathbf{x}'$ is restricted in the tangent plane that passes through $\mathbf{x}_0$ and takes $\mathbf{n}$ as the normal vector. Hence we have $< \mathbf{n}, \mathbf{x}' - \mathbf{x}_0 > \equiv 0$. The volume becomes $|\Omega'| = < \frac{1}{3}\mathbf{n}, \mathbf{x}_0 - \mathbf{y} >$, which is independent of $\mathbf{x}'$.

Please note that the above volume-preserving property does not apply to surface meshes with noise or sharp features. As described in Algorithm 2, we consider crease lines instead of tangent planes for sharp features. For surface meshes with noise (see Algorithm 3), tangent planes are approximated by using a fitting technique. The equation (D.1) does not apply to either case.

References

[1] L. Chen, Mesh smoothing schemes based on optimal Delaunay triangulations, in: 13th International Meshing Roundtable, 2004, pp. 109–120.

[2] M. Desbrun, M. Meyer, P. Schröder, A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, in: SIGGRAPH 1999 Papers, New York, NY, USA, 1999, pp. 317–324.

[3] Y. Ohtake, A. Belyaev, H. P. Seidel, Mesh smoothing by adaptive and anisotropic Gaussian filter applied to mesh normals, in: Vision, Modeling, and Visualization 2002, Erlangen, Germany, 2002, pp. 203–210.

[4] J. Wang, Z. Yu, A novel method for surface mesh smoothing: applications in biomedical modeling, in: Proceedings of the 18th International Meshing Roundtable, 2009, pp. 195–210.

[5] M. Nociar, A. Ferko, Feature-preserving mesh denoising via attenuated bilateral normal filtering and quadrics, in: Proceedings of the 26th Spring Conference on Computer Graphics, ACM, New York, NY, USA, 2010, pp. 149–156.

[6] C. L. Bajaj, G. Xu, Anisotropic diffusion of surfaces and functions on surfaces, ACM Trans. Graph. 22 (2003) 4–32.

[7] G. Taubin, A signal processing approach to fair surface design, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 1995, pp. 351–358.

[8] P. Choudhury, J. Tumblin, The trilateral filter for high contrast images and meshes, in: ACM SIGGRAPH 2005 Courses, ACM, New York, NY, USA, 2005, pp. 186–196.

[9] J. Peng, V. Strela, D. Zorin, A simple algorithm for surface denoising, in: Proceedings of the conference on Visualization '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 107–112.

[10] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, in: ACM SIGGRAPH 2003 Papers, ACM, New York, NY, USA, 2003, pp. 950–953.

[11] R. Bade, J. Haase, B. Preim, Comparison of fundamental mesh smoothing algorithms for medical surface models, in: Simulation und Visualisierung (2006), 2006, pp. 289–304.

[12] Y. Zhang, G. Xu, C. Bajaj, Quality meshing of implicit solvation models of biomolecular structures, Computer Aided Geometric Design 23 (6) (2006) 510–30.

[13] W. Yue, Q. Guo, J. Zhang, G. Wang, 3D triangular mesh optimization in geometry processing for CAD, in: Proceedings of the 2007 ACM symposium on Solid and physical modeling, ACM, New York, NY, USA, 2007, pp. 23–33.

[14] J. a. F. Mari, J. H. Saito, G. Poli, M. R. Zorzan, A. L. M. Levada, Improving the neural meshes algorithm for 3D surface reconstruction with edge swap operations, in: Proceedings of the 2008 ACM symposium on Applied computing, ACM, New York, NY, USA, 2008, pp. 1236–1240.

[15] P. Alliez, Éric Colin de Verdière, O. Devillers, M. Isenburg, Isotropic surface remeshing, Shape Modeling and Applications, International Conference on (2003) 49–58.

[16] S. Valette, J. M. Chassery, R. Prost, Generic remeshing of 3D triangular meshes with metric-dependent discrete voronoi diagrams, IEEE Transactions on Visualization and Computer Graphics 14 (2008) 369–381.

[17] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang, Isotropic remeshing with fast and exact computation of restricted voronoi diagram, Computer Graphics Forum 28 (5) (2009) 1445–1454.

[18] S. Fuhrmann, J. Ackermann, T. Kalbe, M. Goesele, Direct resampling for isotropic surface remeshing, in: Proceedings of Vision, Modeling and Visualization 2010, Siegen, Germany, 2010, pp. 9–16.

[19] D. Field, Laplacian smoothing and Delaunay triangulations, Communications in Applied Numerical Methods 4 (6) (1988) 709–712.

[20] N. Amenta, M. Bern, D. Eppstein, Optimal point placement for mesh smoothing, in: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, 1997, pp. 528–537.

[21] T. Zhou, K. Shimada, An angle-based approach to two-dimensional mesh smoothing, in: Proceedings, 9th International Meshing Roundtable, 2000, pp. 373–384.

[22] Z. Yu, M. J. Holst, J. A. McCammon, High-fidelity geometric modeling for biomedical applications, Finite Elements in Analysis and Design 44 (11) (2008) 715–723.

[23] R. Dyer, H. Zhang, T. Möler, Delaunay mesh construction, in: Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, 2007, pp. 273–282.

[24] L. A. Freitag, On combining Laplacian and optimization-based mesh smoothing techniques, in: Trends in unstructured mesh generation, 1997, pp. 37–43.

[25] S. A. Canann, J. R. Tristano, M. L. Staten, An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes, in: Proceedings of the 7th International Meshing Roundtable, 1998, pp. 479–494.

[26] Y. Ohtake, A. G. Belyaev, I. A. Bogaevski, Polyhedral surface smoothing with simultaneous mesh regularization, in: Geometric Modeling and Processing 2000, IEEE, 2000, pp. 229–237.

[27] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Laplacian mesh optimization, in: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, ACM, New York, NY, USA, 2006, pp. 381–389.

[28] J. Wang, Z. Yu, Quality mesh smoothing via local surface fitting and optimum projection, Graphical Models 73 (4) (2011) 127–139.

[29] L. Chen, J. Xu, Optimal Delaunay triangulations, Journal of Computational Mathematics 22 (2) (2004) 299–308.

[30] L. Chen, M. J. Holst, Efficient mesh optimization schemes based on optimal Delaunay triangulations, Computer Methods in Applied Mechanics and Engineering 200 (2011) 967–984.

[31] P. Alliez, D. Cohen-Steiner, M. Yvinec, M. Desbrun, Variational tetrahedral meshing, Proc. of 2005 ACM SIGGRAPH 24 (2005) 617–625.

[32] J. Tournois, C. Wormser, P. Alliez, M. Desbrun, Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation, ACM Transactions on Graphics 28 (2009) 75:1–75:9.

[33] D. L. Page, Y. Sun, A. F. Koschan, J. Paik, M. A. Abidi, Normal vector voting: Crease detection and curvature estimation on large, noisy meshes, Graphical Models 64 (3-4) (2002) 199–229.

[34] T. R. Jones, F. Durand, M. Desbrun, Non-iterative, feature-preserving mesh smoothing, ACM Transactions on Graphics 22 (2003) 943–949.

[35] X. Sun, P. Rosin, R. Martin, F. Langbein, Fast and effective feature-preserving mesh denoising, Transactions on Visualization and Computer Graphics 13 (5) (2007) 925–938.

[36] B. Vallet, B. Lévy, Spectral geometry processing with manifold harmonics, in: Computer Graphics Forum (Proceedings Eurographics), Vol. 27, 2008, pp. 251–260.

[37] X. Sun, P. L. Rosin, R. R. Martin, F. C. Langbein, Random walks for feature-preserving mesh denoising, Computer Aided Geometric Design 25 (7) (2008) 437–456.

[38] Z. Li, L. Ma, X. Jin, Z. Zheng, A new feature-preserving mesh-smoothing algorithm, The Visual Computer 25 (2009) 139–148.

[39] S. J. Ahn, Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space, Springer, 2005.

[40] N. Aspert, D. Santa-Cruz, T. Ebrahimi, Mesh: measuring errors between surfaces using the hausdorff distance, in: Multimedia and Expo,

2002. ICME '02. Proceedings. 2002 IEEE International Conference on, Vol. 1, 2002, pp. 705–708.

[41] D. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1) (1989) 503–528.

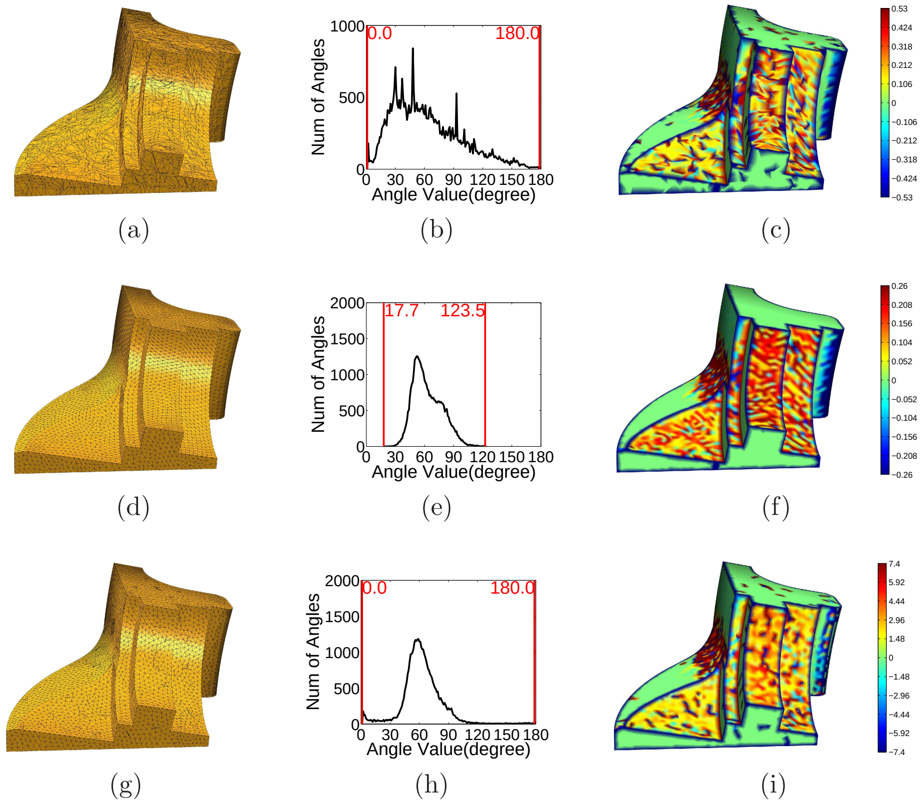


Figure 7: Illustration of the feature-preserving S-ODT method (Algorithm 2) and comparison with remeshing method in [18]. (a-c) show the original noise-free fandisk model containing sharp edges and corners, its angle histogram, and the corresponding distribution map of mean curvatures. (d-f) show the smoothed mesh with significantly improved angle quality and regularized curvatures. (g-i) show the remeshing results using the method in [18]. The model is provided by the AIM@SHAPE Shape Repository.

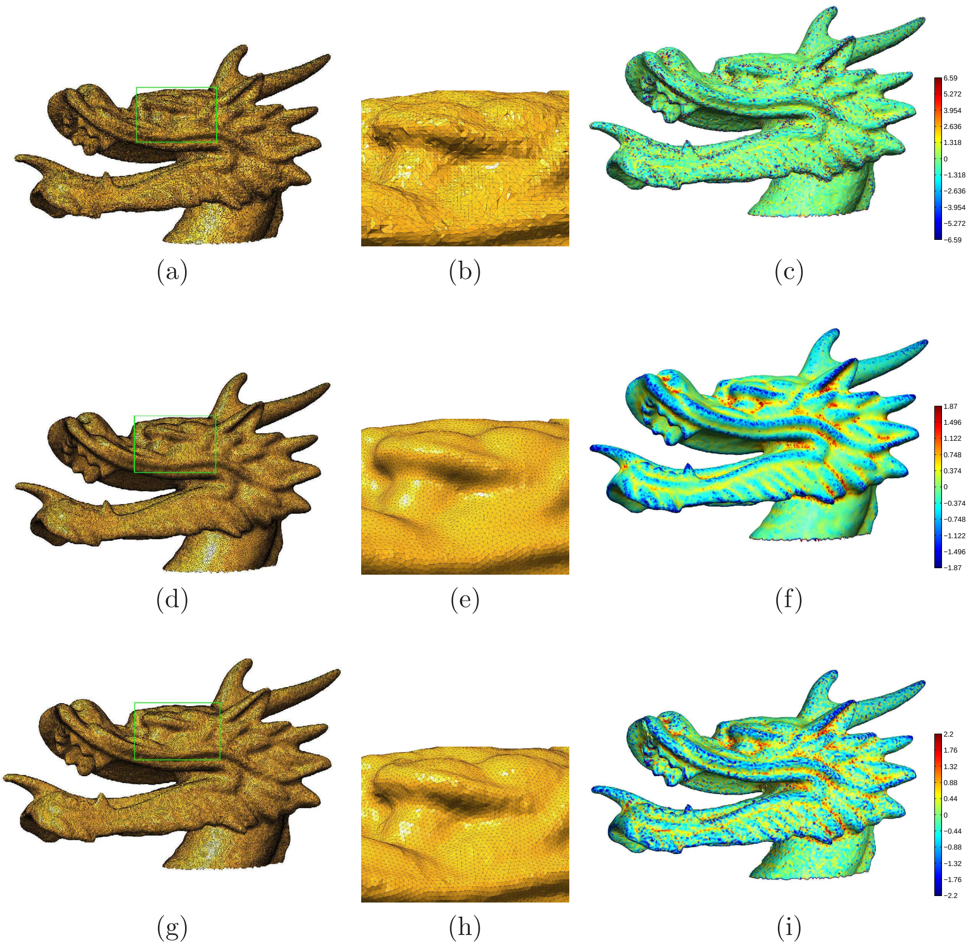


Figure 8: Mesh smoothing of the dragon head model. (a-c) Original mesh with noise and extremely low quality (mesh courtesy of Stanford University - 3D scanning repository). (d-f) Smoothed mesh with significantly improved quality. (g-i) Remeshed results using [17].

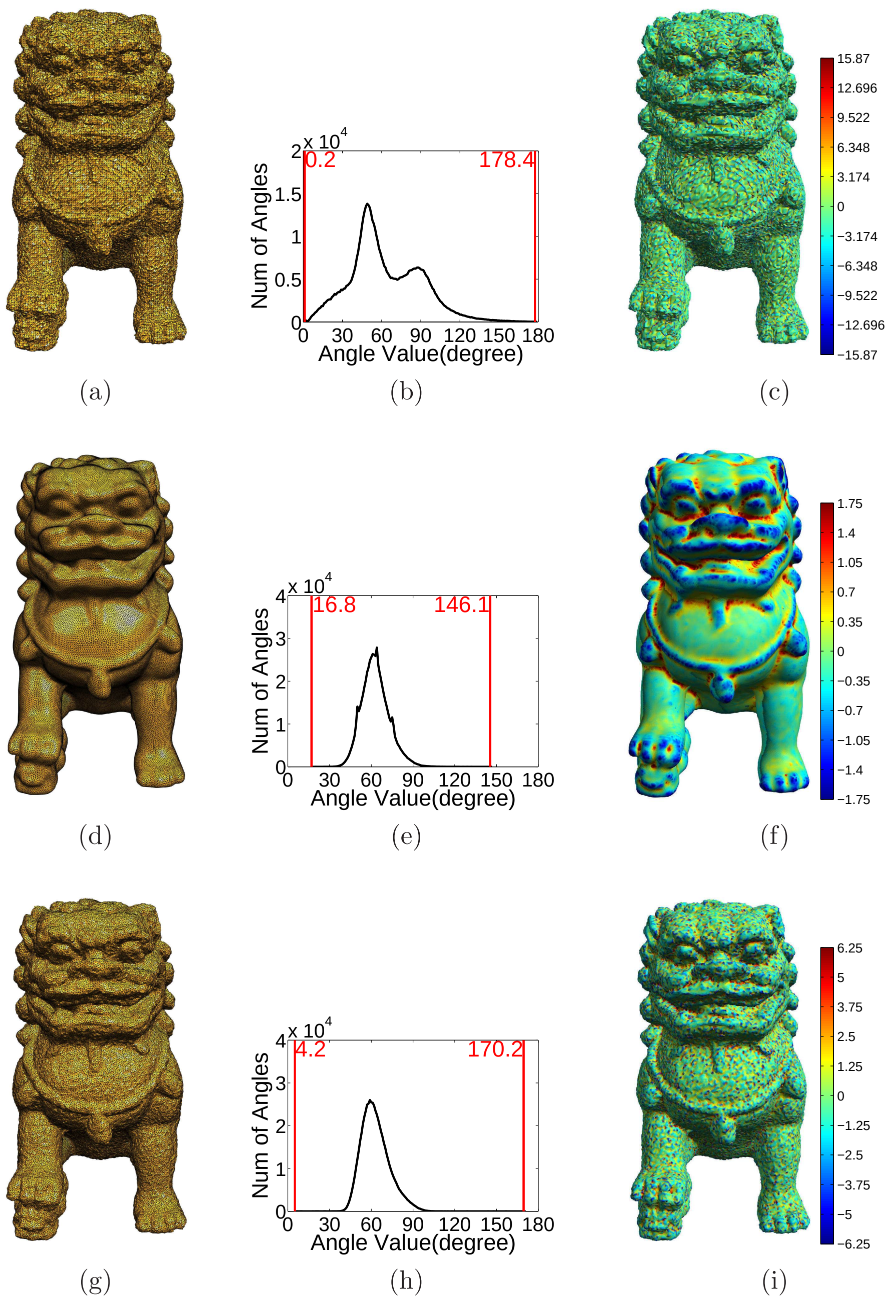


Figure 9: Illustration of the feature-preserving and noise-removing S-ODT method (Algorithm 3) and comparison with remeshing method in [18]. The original and smoothed meshes of the Chinese lion are shown on the top and middle respectively. The remeshed mesh is shown on the bottom. The model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository.

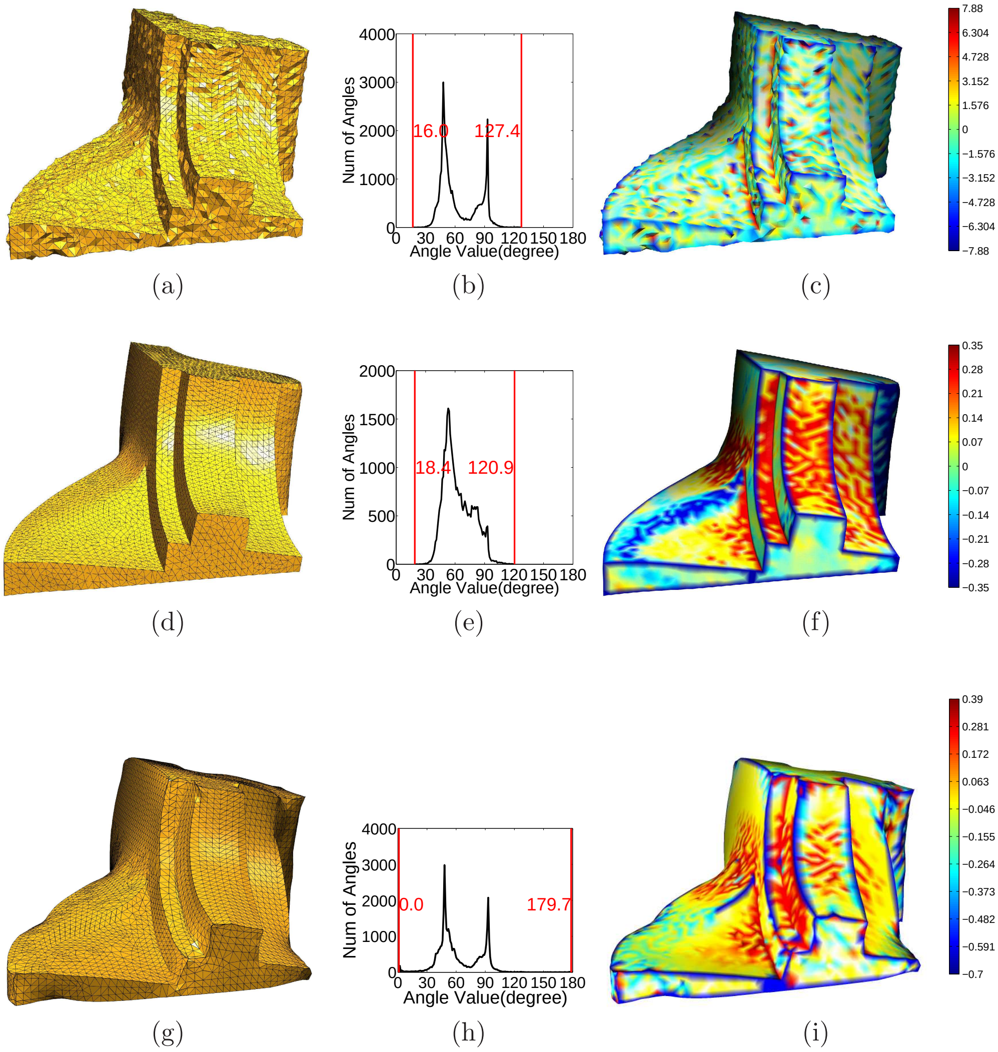


Figure 10: Illustration of the feature-preserving and noise-removing S-ODT method (Algorithm 3). The original and smoothed meshes of the fandisk and their corresponding histograms and curvature maps are shown on the top and middle rows respectively. And the bilateral filtering [10] result is shown in the bottom row.

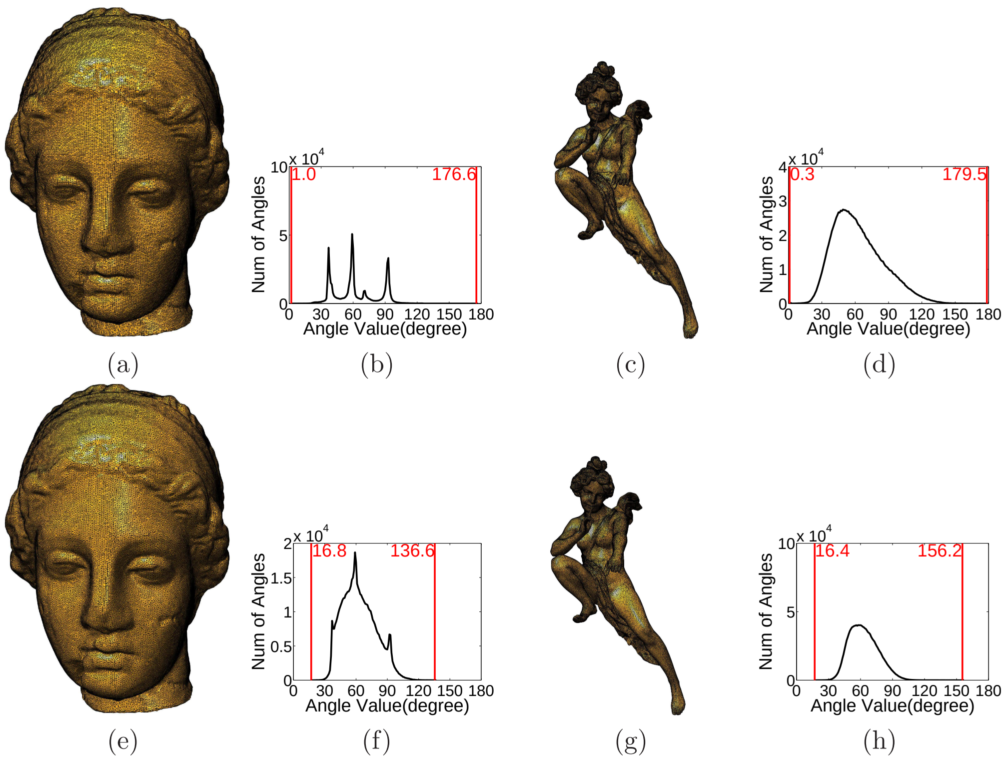


Figure 11: Illustration of the mesh quality improvement of S-ODT method (Algorithm 3) on the Venus and Angel models.

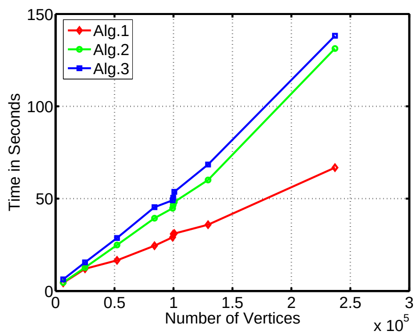


Figure 12: The running time of the three variants of our algorithm, measured on the models with mesh sizes shown in Table II.

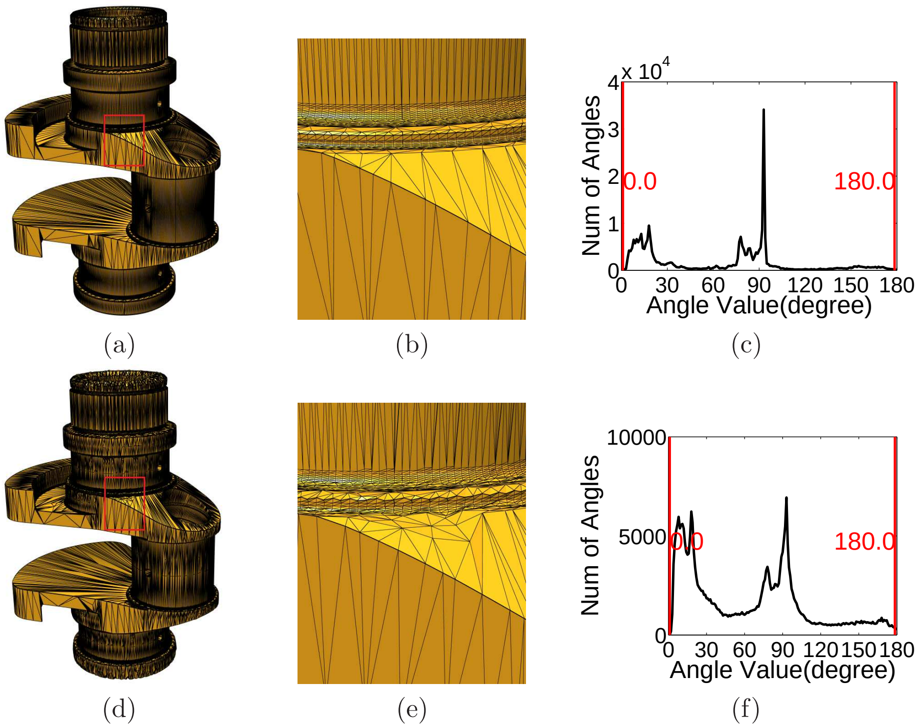


Figure 13: An example for which our algorithm fails in improving the quality. (a) The original crank model, where poorly-shaped triangles are formed by vertices mostly lying on sharp edges. Note that all non-manifold vertices in the mesh have been removed by using the MeshLab tool (http://meshlab.sourceforge.net/) prior to applying our algorithm. (b-c) A closer view of the selected region and the angle histogram of the original mesh. (d-f) The processed mesh using our method (Algorithm 2) and the angle histogram. Overall, the sharp features are preserved and the histogram becomes more uniform after mesh smoothing. But the minimal and maximal angles are not improved. The original mesh is provided courtesy of INRIA by the AIM@SHAPE Shape Repository.