# Volume and Feature Preservation in Surface Mesh Optimization

Xiangmin Jiao

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332

**Summary.** Mesh optimization is critical in numerical simulations involving complex or evolving geometry. Because of the geometric constraints, such as preservation of sharp features and conservation of volume, optimizing a surface mesh poses significant challenges, especially when a CAD model is unavailable. In this paper, we introduce a formulation of volume conservation in a local sense for surface meshes under smoothing or other types of mesh motion, and propose a simple and efficient technique to solve it. We also present a simple and robust feature detection technique to enhance the effectiveness of local volume conservation and mesh optimization. We present the theoretical foundation of our techniques and experimental study to demonstrate their effectiveness.

**Key words:** Mesh optimization; mesh smoothing; surface meshes; volume conservation; feature detection

## 1 Introduction

Mesh optimization is important in mesh generation for numerical simulations [8] and in many simulations with moving boundaries [3]. For its potential high efficiency and simplicity, a commonly used optimization strategy is mesh smoothing, which redistributes the vertices without changing the connectivity of a mesh. Although it has been widely used in optimizing 2-D and 3-D meshes [2, 4, 5, 12, 17], smoothing a surface mesh has some significant challenges due to additional geometric constraints. Two critical and nontrivial constraints are the preservation of sharp features and conservation of volume. Although some sophisticated techniques were developed and used in stand-alone meshing and remeshing tools, they are hard to implement and ill-suited for numerical simulation codes, especially on parallel computers. The lack of simple and effective surface mesh smoothing techniques significantly limits the capabilities or efficiencies of numerical simulations involving complex geometry under significant motion, such as in multiphase flows [25].

Conservation of volume (or mass) is a fundamental issue for accurate and stable numerical simulations, especially the simulations of dynamic systems over a long period of time [19]. It is well-known that a naive procedure, such as Laplacian smoothing, may shrink the volume of a domain substantially [24]. Volume conservation has attracted significant attention in recent years in mesh optimization and surface fairing [9, 19, 23, 27]. In the context of mesh optimization, most methods primarily focus on limiting volume errors during mesh generation or remeshing by projecting the vertices onto a continuous or discrete surface [6, 7, 9, 18]. These methods involve point location procedures, which are potentially expensive and difficult to implement especially on a parallel computer. In addition, if the mesh-optimization procedure must be called repeatedly as required in many numerical simulations, severe volume errors may still occur due to accumulation of errors. The method in [19] enforces volume conservation up to machine precision, but it may incur large local errors near singularities or sharp features.

Another issue intimately related to volume conservation is feature detection and preservation. Feature detection is a critical issue also in its own right in mesh generation, mesh optimization, and numerical simulations [1, 11, 16, 20, 26]. Without proper treatment of features or singularities, large errors may occur during a meshing or remeshing process, and significant undershoots or overshoots may occur near singularities in high-order approximations to a surface. Within numerical simulations, improper treatment of features may undermine the accuracy or stability. Although a number of feature detection techniques have been proposed in the literature, the most robust techniques are also difficult to implement and typically lack a consistent theoretical foundation. Although they may be sufficient for stand-alone meshing tools and interactive environments, simple and robust feature detection techniques are still needed for many other computational applications.

This paper aims at developing simple and efficient techniques for surface mesh optimization that can be easily integrated into numerical simulations based on sound mathematical foundations. The main contributions of the paper are twofold. First, we formulate volume conservation in a *local* sense for meshes under smoothing or other types of mesh motion, and propose a simple and efficient technique to achieve volume conservation. Second, we propose a robust feature detection technique that is relatively easy to implement and to integrate into meshing processes or numerical simulations. Both of these techniques are based on an eigenvalue analysis of a metric tensor at each vertex of the mesh, which combines the asymptotic analysis for fine discretizations of smooth surfaces and the singularity analysis for coarse meshes or sharp features in a fine mesh. This analysis establishes a new theoretical foundation for feature detection and volume conservation in mesh optimization and leads to simple and robust algorithms for them.

The remainder of the paper is organized as follows. Section 2 presents the basic framework for our surface mesh smoothing, referred to as *null-space smoothing*. Section 3 presents a formulation and numerical solution to enforce

volume conservation in a local sense. Section 4 describes a simple and robust feature detection technique that reuses the computational tools present in earlier sections and improves the effectiveness of mesh smoothing and local volume conservation. Section 5 demonstrates the effectiveness of our new technique in surface mesh smoothing. Section 6 concludes the paper with a discussion.

## 2 Null-Space Smoothing

### 2.1 Formulation

We review our basic framework for surface mesh smoothing, referred to as *null-space smoothing*, proposed previously in [15]. This technique smooths a surface mesh by moving each vertex within a *null space*, which in general is a plane, a line, or the empty set, tangential to the surface at the vertex. Let $\mathbf{T}$ denote a matrix whose column vectors are the bases of the null space, and let $\mathbf{c}$ denote the vector from $v$ to the centroid (or a weighted average) of its neighborhood. The null-space smoothing moves $v$ toward the centroid within the null space for a displacement $\mathbf{t}$, i.e.,

$$\mathbf{t} = \mathbf{T}\mathbf{T}^T\mathbf{c}. \tag{1}$$

We define the *null space* using an eigenvalue analysis of a metric tensor. At each vertex $v$, suppose $v$ is the origin of a local coordinate frame, and $m$ be the number of the faces incident on $v$. Let $\mathbf{N}$ be an $m \times 3$ matrix whose $i$th row vector is the unit outward normal to the $i$th incident face of $v$, and $\mathbf{W}$ be an $m \times m$ diagonal matrix with $W_{ii}$ equal to the weight (such as the face area) associated with the $i$th face. Let $\mathbf{A}$ denote $\mathbf{N}^T\mathbf{W}\mathbf{N}$, which we refer to as the *quadric metric tensor*, for its use in the well-known quadric error metric [14]. $\mathbf{A}$ is symmetric positive semi-definite (i.e., $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$), and its eigenvalue decomposition [10] is

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{2}$$

where its eigenvalues $\lambda_i = \Lambda_{ii}$ are all real and nonnegative, and its corresponding eigenvectors $\mathbf{e}_i$ are the column vectors of $\mathbf{V}$. We refer to the vector space spanned by the eigenvectors corresponding to relatively large eigenvalues of $\mathbf{A}$ as its *primary space* and the complementary space as its *null space*.

In general, the null space at a vertex is tangent to the surface. For sufficiently fine meshes, the null space has dimensions 2, 1, or 0 at a smooth, ridge, or corner vertex, respectively, so it is essentially the tangent space of the surface at a vertex. For relatively coarse meshes, however, it is in general a subspace of the tangent space (e.g., it may be a line along the direction of minimum curvature in the tangent plane). Null-space smoothing moves a vertex within the tangent plane or along the direction of minimum curvature,

and therefore it tends to preserve sharp features and areas with large curvatures. In addition, it stops changing a surface after the mesh has converged tangentially. Therefore, null-space smoothing is expected to work well and typically introduce negligible perturbation to a surface in practice.

### 2.2 Analysis of Null-Space Smoothing

We now analyze the error in null-space smoothing more formally in terms of volume change. Consider moving a vertex $v$ by a displacement $\mathbf{t}$. Let $a_i$ and $\mathbf{n}_i$ be the area and unit normal of the $i$th incident face of $v$, respectively, and assume $W_{ii} = a_i$. The volume change is $\delta V = \sum_{i=1}^{m} a_i \mathbf{n}_i^T \mathbf{t}/3 = \|\mathbf{WNt}\|_1/3$. From the definition of $\mathbf{A}$, we have the singular value decomposition [10]

$$\sqrt{\mathbf{W}}\mathbf{N} = \mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T, \tag{3}$$

where $\mathbf{U}$ is an $m \times 3$ matrix, and $\sqrt{\mathbf{W}}$ is the diagonal matrix whose $i$th diagonal entry is the square root of $W_{ii}$ (and similarly for $\sqrt{\mathbf{\Lambda}}$). Let $\mathbf{s}^T = \|\sqrt{\mathbf{W}}\mathbf{U}\|_1$, and then,

$$3\delta V = \|\mathbf{WNt}\|_1 = \|\sqrt{\mathbf{W}}\mathbf{U}\sqrt{\mathbf{\Lambda}}\mathbf{V}^T\mathbf{t}\|_1 = \sum_{i=1}^{3} s_i \sqrt{\lambda_i}(\mathbf{t}^T\mathbf{e}_i). \tag{4}$$

If the mesh is coarse, then $s_i$ (and similarly for $\mathbf{t}^T\mathbf{e}_i$) may have comparable sizes for different $i$, so null-space smoothing introduces a relatively small volume change proportional to the square roots of the smallest eigenvalues. Assume the mesh is relatively uniform and let $h$ be a measure of the average edge length. Following an analysis similar to that in [14], it can be shown that as $h$ approaches 0, the largest eigenvalues are $O(h^2)$ but the smallest eigenvalues corresponding to the null space of $\mathbf{A}$ are $O(h^4)$. In addition, $\mathbf{t}^T\mathbf{e}_i$ is $O(h^2)$, so $\delta V$ is $O(h^4)$ in null-space smoothing with a positional error of $O(h^2)$ even near singularities. If $\mathbf{t}$ contained a component corresponding to larger eigenvalues (such as in Laplacian smoothing), then $\delta V$ would be $O(h^3)$ with a positional perturbation of $O(h)$ near singularities. Therefore, null-space smoothing works significantly better than Laplacian smoothing.

## 3 Volume Conservation

Moving one vertex in null-space smoothing preserves the potential second-order accuracy of a surface triangulation. However, moving the vertices for many iterations may still degrade the order of accuracy. We propose an extension of null-space smoothing to reduce the volume error further. Our basic idea is to add a small component within the primary space at the vertex to correct the volume while preserving the singularities of the surface. If we move

each vertex one by one (i.e., updating in a Gauss-Seidel style), then the problem is relatively simple: we just need to determine a direction $\mathbf{d}$ in the primary space and then a length $\alpha$, such that $\alpha\|\mathbf{WNd}\|_1 = -\|\mathbf{WNt}\|_1$, because from (4) it is obvious that moving the vertex by $\alpha\mathbf{d}+\mathbf{t}$ leads to no volume change. If all vertices are moved concurrently (i.e., updating in a Jacobi style, which is advantageous for its preservation of symmetry and ease of parallelization), we may determine the direction for each vertex independently but the distances of their movement must be solved concurrently. In the following, we will first describe how to determine the directions that are applicable to both Gauss-Seidel and Jacobi styles, and then present how to compute the motion in the Jacobi style.

### 3.1 Estimation of Directions

Let us first consider the problem of estimating a direction at each vertex. The key requirement of this direction is that it must be in the primary space. Intuitively, one might determine a weighted average of face normals and project it onto the primary space, but a naive implementation may be sensitive to weights and hence prone to artifacts near singularities. It is desirable to find a direction that is well behaved near singularities so that it does not vary abruptly between two neighboring ridge vertices.

We compute the directions using a *mean normal* based on an extension of the preceding eigenvalues analysis. Suppose all the faces are offset outwards for a unit distance, and the intersection of the planes passing through the offset faces incident on a vertex $v$ is then the solution to an $m \times 3$ linear system

$$\mathbf{Nx} \approx \mathbf{1}. \tag{5}$$

Since $\mathbf{N}$ may be over- or under-determined, we reformulate it in a least squares sense and obtain a $3 \times 3$ linear system

$$\mathbf{Ax} = \mathbf{b}, \tag{6}$$

where $\mathbf{b} = \mathbf{N}^T\mathbf{W1}$. Let $k$ denote the dimension of the primary space, and assume $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Generally speaking, $k$ is one, two, and three at a smooth, ridge, and corner vertex, respectively, but may be smaller for extremely shape ridge or corner vertices where $\lambda_2$ or $\lambda_3$ is too small compared to $\lambda_1$ (e.g., $\leq \varepsilon\lambda_1$ for $\varepsilon \approx 0.003$). By restricting $\mathbf{x}$ to be within the primary space, the solution to (5) is then

$$\mathbf{x} \approx \sum_{i=1}^{k} \mathbf{e}_i^T\mathbf{b}\mathbf{e}_i/\lambda_i, \tag{7}$$

which is numerically more stable by avoiding division by very small numbers. If $k = 1$ (i.e., the surface is smooth at the vertex), then $\mathbf{x}$ reduces to the first eigenvector, which converges to the outward surface normal. If $k > 1$ (i.e., the surface is singular at the vertex), then $\mathbf{x}$ approximately points to the medial

axis of the surface and in turn approximately bisects the tangent planes at the singularity. For these reasons, $\mathbf{x}$ provides a good estimation of normals and is well behaved at singularities, and therefore we use $\mathbf{d} = \mathbf{x}/\|\mathbf{x}\|$ as the direction at each vertex for volume correction.

### 3.2 Concurrent Vertex Motion

After obtaining the direction at each vertex, we must then solve for the distance $\alpha$ that each vertex moves. In a Gauss-Seidel-style iteration, $\alpha$ is simply $-(\mathbf{b}^T\mathbf{t})/(\mathbf{b}^T\mathbf{d})$ at a vertex, which is similar to the single-node relaxation method in [19]. For Jacobi iterations, however, the problem is more difficult because the volume swept by each face is in general nonlinear in the displacements of the vertices. Although a naive technique such as rescaling the domain or making $\alpha$ uniform for all vertices may restore the total volume in a global sense, it has no physical meaning and may undermine accuracy. We propose a new approach that formulates volume conservation in a local sense to obtain a system of equations and then solves the equation efficiently using a simple iterative procedure. Although our focus is mesh smoothing, we present our formulation in a general form so that it can be easily adapted to other settings.

Suppose a volume flux (or mass flux) $\epsilon$ is given over a surface (where the flux may be due to the tangential motion in the null-space smoothing or other types of surface motion). We define a numerical flux $f$ at each face to be the gain or loss of volume per unit area, i.e.,

$$f = \left( \int_{e'} \mathbf{x}^T \mathbf{n}' \, \mathrm{d}\mathbf{x} - \int_{e} \mathbf{x}^T \mathbf{n} \, \mathrm{d}\mathbf{x} \right) \Big/ \int_{e} 1 \, \mathrm{d}\mathbf{x}, \tag{8}$$

where $e$ and $e'$ denote the face before and after adding the normal motion $\alpha\mathbf{d}$ at the vertices, and $\mathbf{n}$ and $\mathbf{n}'$ denote their unit normals, respectively. Our objective is to make the numerical flux $f$ as close to the prescribed flux $\epsilon$ as possible, i.e., $f \approx \epsilon$. Using a weighted-residual method, we obtain a weak form by requiring the error $f - \epsilon$ project orthogonally onto the function space of the basis functions of the mesh, i.e.,

$$\int (f - \epsilon)\omega_i \, \mathrm{d}\Gamma = 0 \tag{9}$$

for each shape function $\omega_i$ of the mesh. From the summation property $\sum_i \omega_i = 1$, this weak form of local volume conservation enforces global volume conservation strictly, i.e., $\int f \, \mathrm{d}\Gamma = \int \epsilon \, \mathrm{d}\Gamma$.

Given a triangle $e = \mathbf{p}_0\mathbf{p}_1\mathbf{p}_2$, let $\widetilde{\mathbf{d}}_i = \alpha_i \mathbf{d}_i$ and $\mathbf{q}_i = \mathbf{p}_i + \widetilde{\mathbf{d}}_i$. Let $\mathbf{t}_1 = \mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{t}_2 = \mathbf{p}_2 - \mathbf{p}_0$, and $\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2$. Given the displacements, we can evaluate the integral in (9) up to machine precision using numerical quadrature. To analyze $f$, we decompose the prism between $e$ and $e'$ into three tetrahedra, $\mathbf{p}_0\mathbf{p}_1\mathbf{p}_2\mathbf{q}_1$, $\mathbf{p}_0\mathbf{q}_1\mathbf{p}_2\mathbf{q}_2$, and $\mathbf{p}_0\mathbf{q}_1\mathbf{q}_2\mathbf{q}_0$, as illustrated in Fig. 1. If $\mathbf{d}_i$ has the same direction for all vertices, the swept volume $V$ of $e$ is then

$$
\begin{aligned}
V &= \frac{1}{6}\left(\mathbf{n}^T\widetilde{\mathbf{d}}_1 + (\mathbf{t}_1 + \widetilde{\mathbf{d}}_1) \times (\mathbf{t}_2 + \widetilde{\mathbf{d}}_2) \cdot (\widetilde{\mathbf{d}}_0 + \widetilde{\mathbf{d}}_2)\right) \\
&= \frac{1}{6}\left(\mathbf{n}^T(\widetilde{\mathbf{d}}_0 + \widetilde{\mathbf{d}}_1 + \widetilde{\mathbf{d}}_2) + (\widetilde{\mathbf{d}}_1 \times \mathbf{t}_2 + \mathbf{t}_1 \times \widetilde{\mathbf{d}}_2 + \widetilde{\mathbf{d}}_1 \times \widetilde{\mathbf{d}}_2)^T(\widetilde{\mathbf{d}}_0 + \widetilde{\mathbf{d}}_2)\right),
\end{aligned}
$$

which is a cubic function in $\alpha_i$. If $\mathbf{d}_i$ has different directions at different vertices, each quadrilateral of the prism would be bilinear, but it still holds that $V = \mathbf{n}^T(\widetilde{\mathbf{d}}_0 + \widetilde{\mathbf{d}}_1 + \widetilde{\mathbf{d}}_2)/6 + O(\alpha^2)$. Therefore, (9) leads to a nonlinear system of equations in $\alpha_i$.

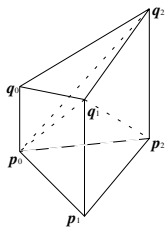



**Fig. 1.** Decomposition of prism swept by face $\mathbf{p}_0\mathbf{p}_1\mathbf{p}_2$ into three tetrahedra

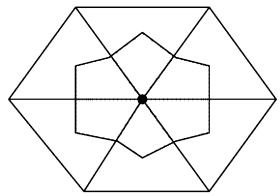


**Fig. 2.** Control volume for approximating numerical flux at vertex

To solve (9) efficiently, we use an efficient quasi-Newton method as follows. Let us define the control volume of each vertex to be composed of one third of each of its incident faces, as illustrated in Fig. 2. Let $g = \sum_i \alpha_i \mathbf{d}_i^T \mathbf{n} H_i$, where $\mathbf{n}$ denote the normal field over the surface, and $H_i$ is a step function, which is 1 in the control volume of the vertex and 0 elsewhere. Then $\int f\omega_i \,\mathrm{d}\Gamma = \int g\omega_i \,\mathrm{d}\Gamma + O(\alpha^2)$. Let $\mathbf{D}$ be the diagonal matrix, where

$$
D_{ij} = \frac{\partial \int g\omega_i \,\mathrm{d}\Gamma}{\partial \alpha_j} = \begin{cases} \frac{1}{3}\mathbf{d}_i^T \boldsymbol{\beta} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{10}
$$

where $\boldsymbol{\beta} = \sum_{k=1}^m a_k \mathbf{n}_k$ (similar to $\mathbf{b}$ in (2) but may be integrated over a different reference geometry). $\mathbf{D}$ approximates the derivative of $\int f\omega_i \mathrm{d}\Gamma$ with respect to $\alpha_j$. Using Broyden's method [13], $\boldsymbol{\alpha}$ can be solved iteratively using the secant equation

$$
\mathbf{D}\left(\boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)}\right) = \mathbf{r}^{(k)}, \tag{11}
$$

where $\mathbf{r}^{(k)}$ is the residual of the $k$th iteration, i.e.,

$$
r_i^{(k)} = -\int (f(\boldsymbol{\alpha}^{(k)}) - \epsilon)\omega_i \,\mathrm{d}\Gamma,
$$

with $r_i^{(1)} = \int \epsilon\omega_i \,\mathrm{d}\Gamma$. Because $\mathbf{D}$ is a diagonal matrix, this equation can be solved very conveniently. In general, the convergence rate of Broyden's method is superlinear and hence this method is very efficient.

We now plug in the above formulation into conservative mesh smoothing. Let $\Gamma$ denote the original surface mesh and $\widetilde{\Gamma}$ denote the surface mesh after tangential motion $\mathbf{t}$. We use $\widetilde{\Gamma}$ as the reference for the numerical integration. Let $V_\gamma(\mathbf{u})$ denote the swept volume of a surface $\gamma$ due to nodal displacements $\mathbf{u}$. The above nonlinear system would then enforce that $V_{\widetilde{\Gamma}}(\widetilde{\mathbf{d}}) = V_{\widetilde{\Gamma}}(-\mathbf{t}) = -V_\Gamma(\mathbf{t})$, and hence

$$V_\Gamma(\mathbf{t}+\widetilde{\mathbf{d}}) = V_\Gamma(\mathbf{t}) + V_{\widetilde{\Gamma}}(\widetilde{\mathbf{d}}) = V_\Gamma(\mathbf{t}) + V_{\widetilde{\Gamma}}(-\mathbf{t}) = V_\Gamma(\mathbf{t}) - V_\Gamma(\mathbf{t}) = 0. \quad (12)$$

In summary, this conservative algorithm proceeds as following:

1. obtain intermediate surface mesh by moving vertices by $\mathbf{t}$, and set $\widetilde{\mathbf{d}} = 0$;
2. for each vertex $v$, compute

$$\widetilde{\mathbf{d}}_v = \widetilde{\mathbf{d}}_v - \frac{\sum_e V_e(\widetilde{\mathbf{d}}) - \sum_e V_e(-\mathbf{t})}{\sum_e a_e \mathbf{n}_e^T \mathbf{d}_v}\mathbf{d}_v; \quad (13)$$

3. repeat step 2 until convergence.

On line 2, the volumes, areas, and normals are computed on the intermediate surface mesh $\widetilde{\Gamma}$, and the summations are over the incident faces of $v$. This quasi-Newton method in practice converges to nearly machine precision for only very few iterations. This method does not require the quadric metric tensor in (6) to be weighted by area, so alternative weighting schemes may be used. Note that (13) may be unstable if $\mathbf{n}_e^T \mathbf{d}_i \approx 0$. This case is unlikely to occur for our choice of $\mathbf{d}$ except at cusps, which are inherently unstable. So vertices on cusps should in general be skipped in this procedure for robustness.

## 4 Feature Detection

In our preceding analysis, it is obvious that detection of geometric features is critical in preserving high-order accuracy in mesh smoothing. In addition, it is important to identify regions with relatively large curvature to reduce errors for coarse meshes. Besides surface smoothing, feature detection also plays an important role in many other geometric and numerical computations involving surfaces, such as mesh generation and remeshing [1, 26], solution transfer across different meshes [16], etc..

A number of feature detection techniques have been proposed in the literature, but most of the robust ones are relatively difficult to implement or to integrate into application codes such as numerical simulations. We present a robust feature-detection technique based on a singularity analysis of the quadric metric tensor, which is an enhancement of our preliminary results in [15]. This method is also based on the eigenvalue analysis and hence is particularly well-suited in our setting. In addition, it can be implemented using only an element connectivity table and hence is easy to be integrated into application codes even on parallel machines.

### 4.1 Identifying Features

The relative sizes of the eigenvalues $\lambda_i$ of $\mathbf{A}$ in (6) are closely related to the local flatness at a vertex, as illustrated in Fig. 3, where the axes of the ellipsoids are aligned with the directions of the eigenvectors and the semiaxes are proportional to the eigenvalues. In general, $\mathbf{A}$ has three large eigenvalues at a corner, two large ones at a ridge, and one large one at a smooth point. It therefore seems natural to compare $\lambda_3/\lambda_1$ and $\lambda_2/\lambda_1$ against some thresholds to identify corners and ridges. Such a process has been used previously in processing image or meshes (such as "tensor voting" [21] and its variations [22]). However, because the metric tensor $\mathbf{A}$ is unsigned, feature detection based on eigenvalues alone cannot distinguish a near cusp (i.e., very acute features) from a flat surface and hence is unreliable for surface meshes with very sharp features.

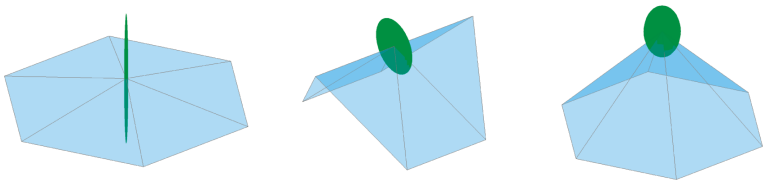


**Fig. 3.** Correlation of eigenvalues and local flatness

We safeguard sharp features as follows. An acute corner can be safeguarded by *angle defect*, denoted by $\theta_a$, which is the difference between $2\pi$ and the sum of the angles at the vertex in its incident faces. To safeguard acute ridges, because the first eigenvector points toward the dominant direction of normals, its projection onto the face normals would vary in signs. Let $\mathbf{e}_1$ point toward the positive side of $\mathbf{b}$ (i.e., $\mathbf{e}_1^T\mathbf{b} > 0$). We classify a vertex $v$ as follows:

1. if $\lambda_3/\lambda_1 \geq \chi_c$ or $|\theta_a| \geq \pi/2$, then $v$ is at corner;
2. if $\lambda_2/\lambda_1 \geq \chi_r$ or $\mathbf{e}_1\mathbf{n} \leq 0$ in incident face, then $v$ is on a ridge.

A tiny (close to zero) $\chi_c$ would classify all vertices as corners, and a large (close to one) $\chi_c$ would classify no corners; similarly for $\chi_r$ and the classification of ridges. Note that if the mesh has mesh folding or cusps, we can extend the second step to report a cusp if $\lambda_2/\lambda_1 \ll \chi_r$ and $\mathbf{e}_1\mathbf{n} \leq 0$ in any face.

To obtain meaningful and intuitive values for $\chi_c$ and $\chi_r$, we connect them with the *dihedral angle* and *open angle*. For a ridge with dihedral angle $\theta \leq \pi/2$ (i.e., the arc-cosine of the inner product of the face normals), assuming the weights in $\mathbf{W}$ are balanced along different sides of a singular point, the eigenvalues satisfy $\lambda_2/\lambda_1 \approx \tan^2(\theta/2)$ and $\lambda_3 \approx 0$. For a cone with an opening angle $\pi - \vartheta$ (i.e., the vertex angle in the cross section through the apex and center of the base), the eigenvalues satisfy $\lambda_3/\lambda_1 \approx \lambda_2/\lambda_1 \approx 2\tan^2(\vartheta/2)$. This

analysis provides a convenient way to choose the thresholds. Specifically, given a user-specified dihedral-angle threshold $\phi_r$ (on $\theta$) and open-angle threshold $\phi_c$ (on $\vartheta$), we then have $\chi_c = 2\tan^2(\phi_c/2)$ and $\chi_r = \tan^2(\phi_r/2)$. For example, if $\phi_r = 15°$ and $\phi_c = 45°$, then $\chi_r \approx 0.03$ and $\chi_c \approx 0.2$. Note that these thresholds assume that the weights in $\mathbf{A}$ are well balanced. For well-graded meshes, which are typically used in finite element analysis, the area weighting would suffice. For nonuniform meshes, as sometimes used in rapid prototyping, the angle weighting delivers more balanced weights (i.e., setting $W_{ii}$ in (2) to be the angle at vertex $v$ in its $i$th incident face).

### 4.2 Filtering Noise

The above feature-detection technique may be sensitive to noise, as it considers the eigenvalues at different vertices independently. It is therefore important to have additional rules to filter out false features and patch missing ones. We achieve this goal by identifying the ridge edges and using the connectivity among the ridge edges and feature vertices. We consider an edge as a *candidate ridge edge* if its dihedral angle is not too small (e.g., $> \phi_r$) and consider a ridge vertex to be *strong* if it is connected to another ridge or corner vertex by a candidate ridge edge. If the dihedral angle of a candidate edge is large (e.g., $> \phi_c$), we immediately accept it as a ridge edge. Otherwise, we accept a candidate edge if it is incident on a strong ridge vertex $v$ and its direction is the closest to the third eigenvector at $v$ (i.e., the inner product of the tangential direction of this edge with the eigenvector is either the positive maximum or the negative minimum among all the candidate edges incident on $v$).

After identifyig ridge edges, we then employ them to filter out false feature vertices, based on the observation that a ridge edge usually points toward a ridge or corner vertex. The filtration proceeds as follows:

1. for each vertex, count the number $k$ of incident ridge edges;
2. upgrade a vertex to a corner if $k > 2$;
3. upgrade a smooth vertex to a ridge if $k = 2$;
4. downgrade a ridge vertex to smooth vertex if $k < 2$ except for acute ridge vertices (i.e., $\mathbf{e}_1^T\mathbf{n} \leq 0$);
5. after reclassfying all vertices, downgrade a ridge edge if neither of its incident vertices is a ridge or corner vertex;

This procedure tends to identify the systematic patterns of ridge curves in the mesh and filter out isolated false features. If the surface is very noisy, we can reduce the sensitivity of the eigenvalues by computing the metric tensor $\mathbf{A}$ as the sum of the tensors of itself and its neighboring ridge vertices and then repeat the classification and filtration procedure. Note that this whole detection procedure can be implemented easily by iterating through the faces or vertices. It does not require advanced data structures (such as the half-edge data structure) other than the standard element connectivity tables, so it is

particularly convenient to be integrated into numerical simulations even on parallel computers.

## 5 Experimental Tests

In this section, we report some experimental tests of our feature detection and conservative mesh smoothing methods. Our formulations are not limited to specific schemes for computing the centroid at each vertex. For simplicity, in our tests we use Laplacian smoothing, which computes the centroid as the average of the neighboring vertices at each vertex, but more sophisticated schemes may be used instead. Figure 4 shows the results of smoothing a surface mesh with a relatively simple geometry with sharp features to study the convergence of our method. During the process, the sharp ridges and corners are automatically identified using our feature detection technique. Four meshes of different resolutions are used and smoothed for 1000 iterations each with two volume-correction steps at each iteration. The left image in Fig. 4 shows the meshes before and after conservative smoothing drawn on top of each other. It is obvious that many vertices moved for a nontrivial distance at smooth regions and along ridges, but the surface remained on top of each other. The right image in Fig. 4 shows the convergence of volume errors using null-space smoothing and conservative smoothing with one, two or three volume-correction steps, where the $x$-axis corresponds to the four different meshes. The convergence rate of null-space smoothing is roughly linear with respect to grid refinement, while conservative smoothing converges faster than fourth-order and its error decreases rapidly as the number of volume-correction steps increases. Figure 5 shows the results of another example using a triple-torus, in which the features are less salient. We show the input meshes and the close-up views of the meshes before and after smoothing near a joint between the tori. The relative volume error was $2.59 \times 10^{-8}$ after 500 steps of conservative smoothing with three iterations of correction steps. The geometry was clearly preserved after smoothing even with substantial tangential motion, owing to the weighted-residual formulation to minimize local errors.

To demonstrate the effectiveness of our feature detection technique, Fig. 6 shows the results of feature detection for three mechanical parts with fine features, obtained from http://www-c.inria.fr/Eric.Saltel/download/. The left images of Fig. 6 show the input meshes and the right images show in translucency the features detected by our method. In all the examples $\phi_r$ was chosen to be 15°. The first two examples (referred to as "thepart" and "fan1" in the mesh colletion) are fairly representative for the coarse meshes commonly used in rapid prototyping or stereolithography. Our method accurately identified all the salient features. The third example uses a typical finite-element mesh of the "fandisk," and remarkably some very fine features were identified by our method without any artifacts, while the commonly-used angle-based methods may have difficulties.

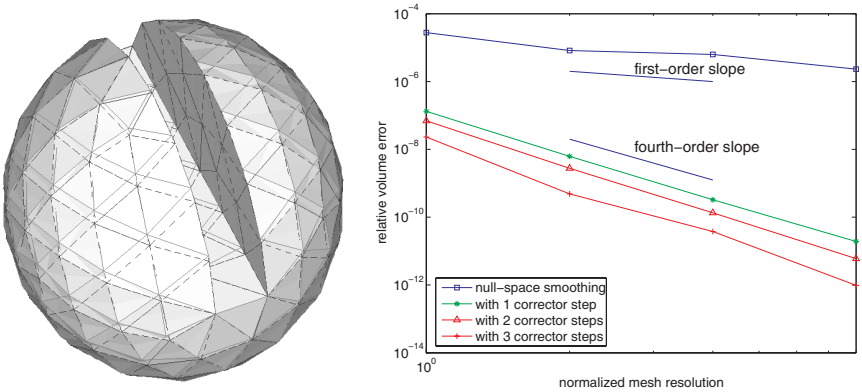
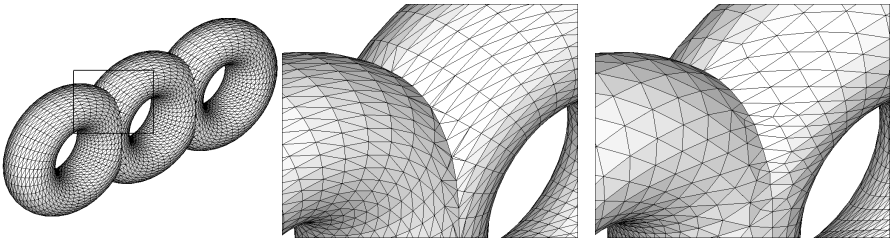



**Fig. 4.** Conservative smoothing of notched sphere and convergence study. In left image, dashed blue edges correspond to input mesh and solid green edges correspond to smoothed mesh



(a) Overall geometry. (b) Before smoothing. (c) After smoothing.

**Fig. 5.** Sample result of conservative smoothing of triple torus

## 6 Conclusion

In this paper, we presented novel techniques to conserve volume and preserve features in mesh optimization. Our techniques are based on an eigenvalue analysis of the quadric metric tensor. Due to their unified theoretical foundation, the analysis of our technique is relatively simple and coherent. More importantly, our techniques are easy to implement and does not required sophisticated data structures (such as the half-edge data structure) and expensive geometric algorithms (such as high-order surface reconstruction and point location), so they are particularly suitable to be integrated into numerical simulations. The proposed volume-conservation technique is also promising to generalize to other moving surfaces with a source term. In this paper, we only reported experimental results using Laplacian smoothing to compute the centroids around each vertex, but more sophisticated schemes can be used and are currently being investigated.

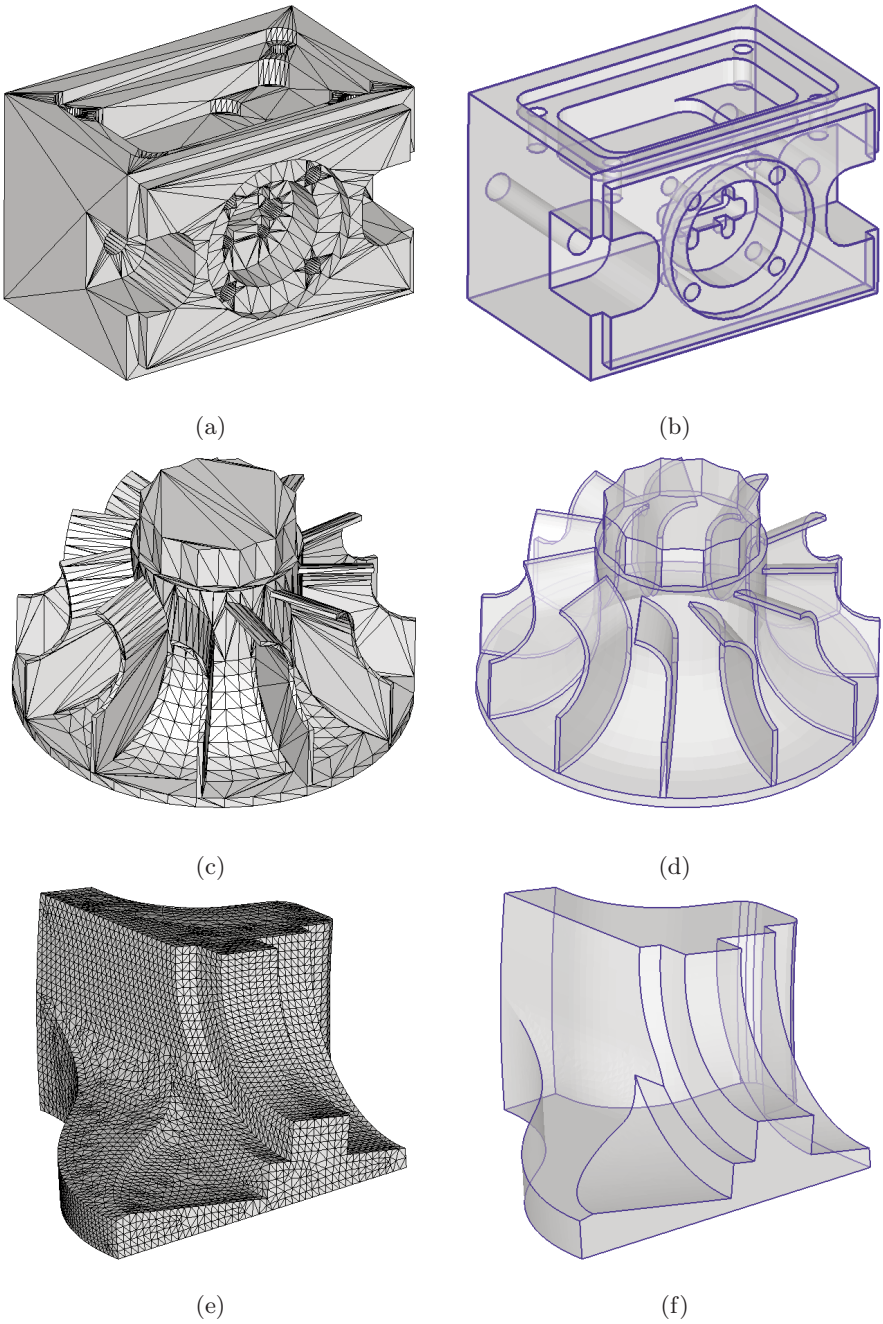



**Fig. 6.** Sample results of detecting features in mechanical parts

## Acknowledgement

This work was supported by the U.S. Department of Energy through the University of California under subcontract B523819 with the University of Illinois at Urbana-Champaign, and by NSF and DARPA under CARGO grant #0310446.

## References

[1] T. Baker. Identification and preservation of surface features. In *Proc. 13th Int. Meshing Roundtable*, pages 299–310, 2004.

[2] M. Brewer, L. F. Diachin, P. Knupp, T. Leurent, and D. Melander. The Mesquite mesh quality improvement toolkit. In *Proc. 12th Int. Meshing Roundtable*, pages 239–250, 2003.

[3] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodriguez-Ferran. Arbitrary Lagrangian-Eulerian methods. In E. Stein, R. de Borst, and T. J. Hughes, editors, *Encyclopedia of Computational Mechanics*, chapter 14. Wiley, 2004.

[4] L. A. Freitag and P. M. Knupp. Tetrahedral mesh improvement via optimization of the element condition number. *Int. J. Numer. Meth. Engr.*, 53:1377–1391, 2002.

[5] L. A. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *Int. J. Numer. Meth. Engr.*, 49:109–125, 2000.

[6] P. Frey and H. Borouchaki. Geometric surface mesh optimization. *Computing and Visualization in Science*, pages 113–121, 1998.

[7] P. J. Frey. About surface remeshing. In *Proc. 9th Int. Meshing Roundtable*, pages 123–136, 2000.

[8] P. J. Frey and P. George. *Mesh Generation: Application to finite elements.* Hermes, 2000.

[9] R. V. Garimella, M. J. Shashkov, and P. M. Knupp. Surface mesh quality optimization using local parametrization. In *Proc. 11th Int. Meshing Roundtable*, pages 41–52, 2002.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computation.* Johns Hopkins University Press, 3rd edition, 1996.

[11] S. Gumhold, X. Wang, and R. Macleod. Feature extraction from point clouds. In *Proc. 10th Int. Meshing Roundtable*, pages 293–305, 2001.

[12] G. A. Hansen, R. W. Douglass, and A. Zardecki. *Mesh Enhancement: Selected Elliptic Methods, Foundations And Applications.* Imperial College, 2005.

[13] M. T. Heath. *Scientific Computing: An Introductory Survey.* McGraw–Hill, New York, 2nd edition, 2002.

[14] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom.*, pages 49–65, 1999.

[15] X. Jiao and P. Alexander. Parallel feature-preserving mesh smoothing. In *Proc. Int. Conf. Comput. Sci. Appl.*, pages 1180–1189, 2005.

[16] X. Jiao and M. T. Heath. Overlaying surface meshes, part ii: Topology preservation and feature detection. *Int. J. Comput. Geom. Appl.*, 14:403–419, 2004.

[17] P. Knupp and S. Steinberg. *Fundamentals of Grid Generation.* CRC Press, 1994.

[18] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part i–a framework for surface mesh optimization. *Int. J. Numer. Meth. Engr.*, 48:401–420, 2000.

[19] A. Kuprat, A. Khamayseh, D. George, and L. Larkey. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. *J. Comput. Phys.*, 172:99–118, 2001.

[20] A. Lopez, F. Lumbreras, J. Serrat, and J. Villanueva. Evaluation of methods for ridge and valley detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:327–335, 1999.

[21] G. Medioni, M.-S. Lee, and C.-K. Tang. *A computational framework for segmentation and grouping*. Elsevier, 2000.

[22] D. L. Page, A. F. Koschan, Y. Sun, J. K. Paik, and M. A. Abidi. Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *Proc. Intl. Conf. on Computer Vision*, volume 1, pages 162–167, 2001.

[23] I. B. Semenova, V. V. Savchenko, and I. Hagiwara. Two techniques to improve mesh quality and preserve surface characteristics. In *Proc. 13th Int. Meshing Roundtable*, pages 277–288, 2004.

[24] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. of Int. Conf. on Computer Vision*, pages 902–907, 1995.

[25] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001.

[26] S. Yamakawa and K. Shimada. Polygon crawling: Feature-edge extraction from a general polygonal surface for mesh generation. In *Proc. 14th Int. Meshing Roundtable*, pages 257–274, 2005.

[27] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *Proc. 13th Int. Meshing Roundtable*, 2005.