Parallel Feature-Preserving Mesh Smoothing

Xiangmin Jiao and Phillip J. Alexander

Computational Science and Engineering,
University of Illinois, Urbana, IL 61801, USA
{jiao, palexand}@cse.uiuc.edu

Abstract. We present a parallel approach for optimizing surface meshes by redistributing vertices on a feature-aware higher-order reconstruction of a triangulated surface. Our method is based on a novel extension of the fundamental quadric, called the *medial quadric*. This quadric helps solve some basic geometric problems, including detection of ridges and corners, computation of one-sided normals along ridges, and construction of higher-order approximations of triangulated surfaces. Our new techniques are easy to parallelize and hence are particularly beneficial for large-scale applications.

Keywords: Computational geometry; feature detection; mesh smoothing; quadric.

1 Introduction

In this paper, we devise new techniques for estimating normals and identifying geometric features for triangulated surfaces, and apply them to redistribute vertices of surface meshes on parallel machines. Mesh smoothing is an important problem in many computational applications [4]. It is frequently used as a post-processing step in mesh generation, and is critical in numerical simulations with deforming geometry. Compared to two-dimensional meshes, surface meshes are particularly difficult to optimize, because curved shapes and sharp features of geometric models must be preserved, frequently without the availability of the underlying CAD models. Therefore, feature-aware higher-order approximations must be constructed by analyzing discrete surfaces. In large-scale scientific simulations, the problem is even more challenging, because meshes are partitioned and distributed across multiple processes on a parallel machine, making it difficult to apply some traditional analytic and reconstruction techniques.

To achieve our objectives, we first develop new techniques to estimate surface normals, especially one-sided normals along ridges, and devise a new vertex-based scheme for detecting ridges and corners of a triangulated surface. Our techniques are based on an extension of the well-known fundamental quadric [2, 9] and tensor voting [12]. These previous techniques provide insights into the local geometry of a discrete surface, but suffer from ambiguities such as undetermined signs of the estimated normals and indistinction between near cusps and smooth surfaces. Our extension, called the *medial quadric*, implicitly uses a local coordinate frame with origin approximately on the medial axis to resolve these ambiguities. Utilizing the results of the medial quadric, we then present a feature-aware higher-order reconstruction of a surface triangulation, and integrate them to deliver a parallel method for surface mesh smoothing.

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3483, pp. 1180–1189, 2005.

2 Estimation of Vertex Normals

Surface mesh smoothing, like many other geometric problems, requires accurate estimation of vertex normals. We present a novel concept called the *medial quadric*, which connects two seemingly unrelated classes of normal estimations (namely, *weighted averaging* [13] and *tensor voting* [12]) and subsequently develop a new estimation method.

Weighted Averaging. A commonly used approach for estimating vertex normals is to average the (potentially weighted) normals of the faces incident on a vertex. There is no consensus on the best choice of weights [15]. The simplest weighting scheme is to use unit weight for every face [5]. Other popular schemes include area-weighted average [13] and angle-weighted average [17]. Another scheme was recently derived to recover the exact normal if the mesh is a tessellation of a sphere [11]. Empirically, these weighting schemes produce nearly identical results for well-shaped smooth surfaces. For well-shaped surfaces with singularities, angle-weighted averaging tends to deliver balanced weights and hence better results, but it is sensitive to perturbation for surfaces with obtuse (especially nearly 180°) triangles. We therefore propose a *guarded angle-weighting scheme* to take the smaller of the edge angle at a vertex and its complement as the weight for each face, i.e., $w_i = \min\{\theta_i, \pi - \theta_i\}$, where θ_i is the edge angle in the ith incident face.

Quadric and Tensor Voting. Another class of estimation is obtained through eigen-decomposition of a quadric. Let γ be a plane containing a point $\boldsymbol{p} \in \mathbb{R}^3$ with unit normal vector $\hat{\boldsymbol{n}}$. The *offset* of γ from the origin is $\delta = -\boldsymbol{p}^T\hat{\boldsymbol{n}}$. The signed distance of γ from any point $\boldsymbol{x} \in \mathbb{R}^3$ is then

$$d(\boldsymbol{x}, \gamma) = (\boldsymbol{x} - \boldsymbol{p})^T\hat{\boldsymbol{n}} = \boldsymbol{x}^T\hat{\boldsymbol{n}} + \delta. \tag{1}$$

Given a collection of planes $\{\gamma_i\}$ (in particular, the tangent planes of the triangles incident on a vertex), let $\hat{\boldsymbol{n}}_i$ denote their unit outward normals, δ_i their offsets, and w_i their associated *positive* weights. The weighted sum of squared distances to γ_i from $\boldsymbol{x}$ is

$$Q(\boldsymbol{x}) = \sum_i w_i d^2(\boldsymbol{x}, \gamma_i) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + 2\boldsymbol{b}^T\boldsymbol{x} + c, \tag{2}$$

where $\boldsymbol{A} = \sum_i w_i\hat{\boldsymbol{n}}_i\hat{\boldsymbol{n}}_i^T$, $\boldsymbol{b} = \sum_i w_i\delta_i\hat{\boldsymbol{n}}_i$, and $c = \sum_i w_i\delta_i^2$. The metric Q is the well-known *fundamental quadric* [2, 9], which is minimized in $\mathbb{R}^3$ at the solution of the 3×3 linear system

$$\boldsymbol{A}\boldsymbol{x} = -\boldsymbol{b}. \tag{3}$$

In general, $\boldsymbol{A}$ is symmetric and positive semi-definite, and its eigenvalues are all real and nonnegative. For an introduction to eigenvalue problems, see textbooks such as [6].

Let λ_i be the eigenvalues of $\boldsymbol{A}$ such that $\lambda_1 \geq \lambda_2 \geq \lambda_3$, and $\hat{\boldsymbol{e}}_i$ their corresponding orthonormal eigenvectors. Based on the spectrum theorem, $\boldsymbol{A}$ can be decomposed into

$$\boldsymbol{A} = \sum_{i=1}^{3} \lambda_i\hat{\boldsymbol{e}}_i\hat{\boldsymbol{e}}_i^T. \tag{4}$$

If the neighborhood of a vertex v is smooth, then $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$ are approximately the principal directions at v, and $\hat{\boldsymbol{e}}_1$ approximates the normal direction [9, 12]. This approximation scheme is referred to as *tensor voting* [12] or *normal voting* [14] in the literature. However, it has the following limitations:

- the direction of $\hat{\boldsymbol{e}}_1$ is also sensitive to weights, similar to weighted averaging
- the sign of $\hat{\boldsymbol{e}}_1$ is undetermined and may point inward or outward
- if the vertex v is on a sharp ridge with dihedral angle $> \pi/2$, then $\hat{\boldsymbol{e}}_2$ instead of $\hat{\boldsymbol{e}}_1$ provides a more meaningful approximation to the normal
- if the ridge nearly forms a right angle, then none of the eigenvectors provides a meaningful approximation to the normal

Another popular approach, which is the dual of tensor voting, is to take the eigenvector corresponding to the smallest eigenvalue of the matrix $\sum_i w_i \hat{\boldsymbol{t}}_i \hat{\boldsymbol{t}}_i^T$, where $\hat{\boldsymbol{t}}_i$ is a tangent vector of the ith face [16]. This approach has limitations similar to tensor voting.

Medial Quadric. To overcome the above limitations, suppose there is a point $\boldsymbol{o}$ such that all faces have the same *negative* offset δ to $\boldsymbol{o}$. As the quadric is scale- and position-independent, without loss of generality, we take $\delta = -1$ and place the origin of the coordinate frame at $\boldsymbol{o}$. Because $\boldsymbol{o}$ is on the *medial axis* of the surface, we refer to this quadric as the *medial quadric*. This quadric is minimized by the solution of (3) with

$$\boldsymbol{b} = -\sum_i w_i \hat{\boldsymbol{n}}_i. \tag{5}$$

The unit vector of $-\boldsymbol{b}$ (i.e., the right-hand side of (3)) is the *weighted-average* outward normal. The solution $\boldsymbol{x}$ is the position vector from $\boldsymbol{o}$ to v, and its unit vector $\hat{\boldsymbol{x}}$ delivers another approximation to the outward normal at v. Unlike the weighted-average normals or eigenvectors, however, $\hat{\boldsymbol{x}}$ is independent of the weights given that the point $\boldsymbol{o}$ exists (because if it exists, $\boldsymbol{o}$ is uniquely defined regardless of the weights).

Another geometric interpretation of the medial quadric is that the origin $\boldsymbol{o}$ is at the intersection of the planes that are parallel to the γ_i with a normal distance -1. When such an intersection does not exist exactly, the origin would then be at the intersection in the least squares sense. When the planes γ_i are nearly parallel to each other, this intersection and in turn $\hat{\boldsymbol{x}}$ are sensitive to perturbation. Numerically, this sensitivity corresponds to a large condition number of $\boldsymbol{A}$. To solve for $\boldsymbol{x}$ robustly, we constrain it within the primary space of $\boldsymbol{A}$, i.e. the space spanned by the eigenvectors corresponding to relatively large eigenvalues. Let d be the dimension of the primary space, and $\boldsymbol{V}$ be a $3 \times d$ matrix, whose ith column is $\hat{\boldsymbol{e}}_i$. The solution of Eq. (3) then reduces to finding a vector $\boldsymbol{s} \in \mathbb{R}^d$, such that Q is minimized at $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{s}$. The vector $\boldsymbol{s}$ is the solution to

$$\left(\boldsymbol{V}^T \boldsymbol{A} \boldsymbol{V}\right) \boldsymbol{s} = -\boldsymbol{V}^T \boldsymbol{b}, \tag{6}$$

which is an $d \times d$ linear system. The condition number of Eq. (6) is then λ_1/λ_d. The solution to (6) is $s_i = -\hat{\boldsymbol{e}}_i^T \boldsymbol{b}/\lambda_i$, and hence

$$\boldsymbol{x} = \sum_{i=1}^{d} s_i \hat{\boldsymbol{e}}_i = \sum_{i=1}^{d} -\hat{\boldsymbol{e}}_i^T \boldsymbol{b} \hat{\boldsymbol{e}}_i/\lambda_i. \tag{7}$$

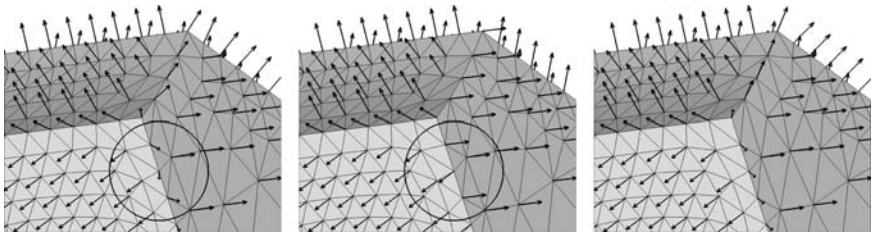

Fig. 1. Demonstration of effect of imbalance of weights. From left to right, arrows indicate estimated normals from averaging, tensor voting, and medial quadric, all weighted by area

In particular, when $d = 1$ (i.e., for smooth points), $\boldsymbol{x}$ is along the direction of $\hat{\boldsymbol{e}}_1$ where the sign is determined by the weighted average. For $d \geq 2$ (such as at a ridge or corner), $\boldsymbol{x}$ is then a linear combination of the eigenvectors and delivers an approximation *mean normal*, which is insensitive to weights. Fig. 1 compares the normal estimations from weighted-averaging, tensor voting, and medial quadric. Only the medial quadric delivers consistent weight-insensitive approximation along features.

One-Sided Normals. The medial quadric is particularly useful in estimating one-sided normals along ridges. In particular, given a face σ, let $\hat{\boldsymbol{n}}_\sigma$ be its face normal and $\boldsymbol{t}_\sigma$ its average tangent pointing from v to its opposite edge center. Given that the weights are balanced between the two sides of the ridge, the one-sided normal on the side of σ at v is

$$\hat{\boldsymbol{n}}_+ = \left(\sqrt{\lambda_I}\hat{\boldsymbol{x}} + \text{sign}\left(\hat{\boldsymbol{n}}_\sigma^T \hat{\boldsymbol{y}}\right)\sqrt{\lambda_J}\hat{\boldsymbol{y}}\right) / \sqrt{\lambda_1 + \lambda_2}, \tag{8}$$

where $I = \text{argmax}_i\{|s_i| \mid 1 \leq i \leq 2\}$, $J = 3 - I$, and $\hat{\boldsymbol{y}}$ is the binormal $\hat{\boldsymbol{x}} \times \hat{\boldsymbol{e}}_3$. To arrive at the coefficients, assuming the total weight $w \equiv \sum w_i$ is balanced between the two sides at a ridge vertex, then λ_1 and λ_2 with dihedral angle θ are $w \max\{\sin^2(\theta/2), \cos^2(\theta/2)\}$ and $w \min\{\sin^2(\theta/2), \cos^2(\theta/2)\}$, respectively. Therefore, the guarded angle-weighted scheme tends to produce reasonable estimates except next to obtuse triangles. For meshes with obtuse triangles, a more accurate estimation can be obtained at additional cost, by constructing and solving one-sided quadrics for each ridge vertex, i.e., $\boldsymbol{A}_+\boldsymbol{x}_+ = \boldsymbol{b}_+$, where $\boldsymbol{A}_+$ and $\boldsymbol{b}_+$ are constructed using the faces of which $\text{sign}(\hat{\boldsymbol{n}}_i^T \hat{\boldsymbol{y}}) > 0$.

3 Vertex-Based Geometric Features

Extracting features (or singularities) from a discretized surface is an important subject for geometric applications. Most feature detection schemes are *edge-based*, in that they first identify ridge edges and then classify vertices based on edge classification; see e.g. [1, 10]. To facilitate feature detection for a partitioned surface mesh on a parallel machine, we present a vertex-based detection scheme, which extends the approach of Medioni et al. [12] with inspirations from the medial quadric.

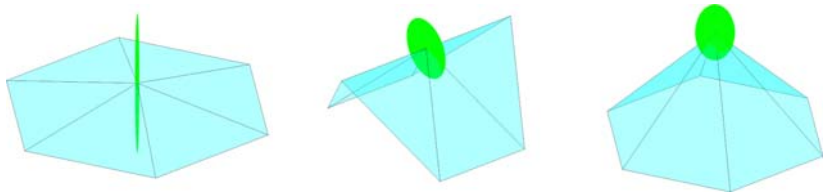

Fig. 2. Differing behavior of eigenvalues at smooth, ridge, and corner points. Eigenvalues and eigenvectors are depicted by ellipsoids, whose axes are aligned along eigenvectors, with semi-axes proportional to their corresponding eigenvalues

Vertex Classification. The relative sizes of eigenvalues of the matrix $\boldsymbol{A}$ of Eq. (3) are closely related to the local smoothness at a vertex v, as illustrated in Fig. 2. More formally, $\boldsymbol{A}$ can be expressed as the linear combination of

$$\boldsymbol{A} = (\lambda_1 - \lambda_2)\boldsymbol{E}_1 + (\lambda_2 - \lambda_3)\boldsymbol{E}_2 + \lambda_3\boldsymbol{E}_3, \tag{9}$$

where $\boldsymbol{E}_d \equiv \sum_{i=1}^{d} \hat{\boldsymbol{e}}_i\hat{\boldsymbol{e}}_i^T$ are the *saliency tensors* for surface, curve (ridge), and point (corner) for $d = 1$, 2, and 3, respectively [12]. The relative sizes of these components were used in [12] and [14] to classify features. Similar to such approaches, we define

$$r = \lambda_3 / \max\{\lambda_1 - \lambda_2, \lambda_2 - \lambda_3\}$$

as the *corner saliency* and consider a vertex as a corner if s is larger than a threshold β. A tiny (nearly zero) β would classify all vertices as corners and a huge (nearly infinity) β would classify no corners. Given a user-specified ridge-angle threshold ψ, as a rule of thumb, we take $\beta \approx \cot\psi$. When r is small, unlike previous methods, we consider λ_3 and its corresponding eigenvector as noise in the model, and hence classify a ridge by comparing λ_2/λ_1 against a threshold α. This approach leads to a more reliable classification for ridges in practice than previous methods. For a ridge with dihedral angle $\theta \leq \pi/2$, the eigenvalues satisfy $\lambda_2/\lambda_1 \approx \tan^2(\theta/2)$, and therefore we set $\alpha = \tan^2(\psi/2)$.

Because matrix $\boldsymbol{A}$ is independent of the signs of normals, the proceeding approach may falsely classify a sharp ridge (e.g., a near cusp) as a smooth surface and classify a sharp corner as a ridge. To resolve this issue, we observe that acute angles are accompanied by the reversal of the order of $\left|\boldsymbol{x}^T\hat{\boldsymbol{e}}_i\right| = \left|s_i\boldsymbol{b}^T\right|/\lambda_i$ (c.f. Eq. (7)), and this order is nearly independent of the weights. Therefore, we introduce a safeguard $g_i = \left|\hat{\boldsymbol{b}}^T\hat{\boldsymbol{e}}_i\right| / \min\{\varepsilon\lambda_1, \lambda_i\}$, where ε (say 10^{-7}) avoids potential division by zero. In summary, a vertex is classified as follows:

1. if $\text{argmax}_i\{g_i\} = 3$ or $\lambda_3 \geq \beta\max\{\lambda_1 - \lambda_2, \lambda_2 - \lambda_3\}$, then v is a corner
2. otherwise, if $\text{argmax}_i\{g_i\} = 2$ or $\lambda_2 \geq \alpha\lambda_1$, then v is on a ridge
3. otherwise, v is at a smooth point

To demonstrate the robustness of this new method, Fig. 3 highlights the ridge vertices detected by our approach with $\psi = 20^\circ$ and by the normal-voting scheme [14] with

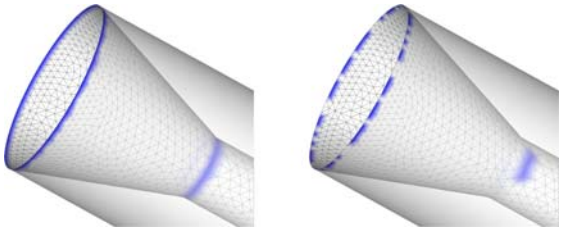

Fig. 3. Comparison of ridge detection with our new method (left) and normal voting (right)

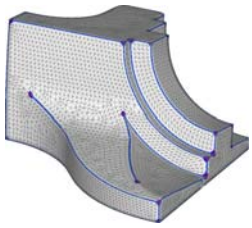

Fig. 4. Features on fandisk detected by our method

$(\lambda_1 - \lambda_2)/(\lambda_0 - \lambda_1) \geq \cos\psi/\sin^2(\psi/2)$ for $\psi = 19°$ (because all vertices would be classified to be smooth with $\psi \geq 20°$). Our scheme is clearly more reliable and less sensitive to perturbation.

Edge Classification. If vertex v is a ridge vertex, then the eigenvector $\hat{\boldsymbol{e}}_3$ is approximately tangential to the ridge, and its incident ridge edges are nearly parallel to $\hat{\boldsymbol{e}}_3$. In addition, the other vertex of either of its incident ridge edges is most likely also a ridge or corner vertex. Therefore, we identify ridge edges as follows. Let $\hat{\boldsymbol{t}}_\tau$ denote the unit tangent of an edge τ incident on v pointing away from v. For each ridge vertex, compute the largest (positive) and the smallest (negative) values of $s_\tau \equiv m_\tau \hat{\boldsymbol{e}}_3^T \hat{\boldsymbol{t}}_\tau$, where m_τ is the number of incident ridge or corner vertices of τ. An incident edge is on the ridge if its associated s_σ has either of the two extreme values and $|s_\tau| \geq 2\cos\psi$, i.e., τ is nearly parallel to $\hat{\boldsymbol{e}}_3$. After classifying all ridge edges, a ridge vertex is upgraded to a corner if it is incident on more than two ridge edges or $|s_\tau| < 2\cos\psi$ for either of its extreme values of s_τ. Fig. 4 shows the corners and ridges after these adjustments for the fandisk model, where even weak features were detected accurately by our method.

4 Feature-Preserving PN Triangles

Utilizing normal estimation and feature detection, we now develop a feature-aware higher-order approximation of a surface triangulation suitable for parallelization. In particular, we employ curved point-normal triangles, or *PN triangles* [18], which provide a simple and convenient way to construct a piecewise cubic approximation of a surface with triangular Bézier patches from vertex coordinates and normals. The resulting approximation is C^1 continuous at vertices and C^0 continuous everywhere else, and a separate quadratic approximation of normals recovers continuity of normals. PN triangles are constructed triangle-by-triangle, without using additional neighbor information, and therefore make a good candidate for distributed meshes.

Summary of PN Triangles. The key component of PN triangles is to determine the seven non-vertex control points for each triangle (two along each edge and one inside the face) from the vertices. The construction first linearly interpolates the control points, so that edge points uniformly subdivide the edges and the face point is at the centroid.

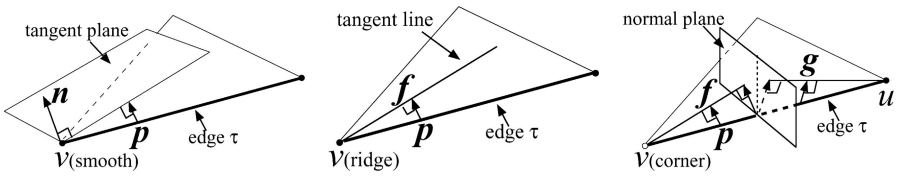

Fig. 5. Construction of control points along edges for feature-preserving PN triangles at smooth, ridge, and corner vertices, respectively

Each control point $\boldsymbol{p}$ on edge τ is then moved by a displacement $\boldsymbol{f}$, as we will describe shortly. The centroid is moved by a displacement equal to 1.5 times the average displacement of the six edge points, so that quadratic polynomial patches can be reconstructed exactly [3]. To construct a continuous normal field, the normal direction at the midpoint of an edge τ is set to the mirror image of the average normal of the vertices of τ against the normal plane. For details, readers are referred to [18]. This simple construction delivers good results for smooth surfaces, but some amendment is needed at the presence of sharp features.

Feature Treatments. To deliver a systematic treatment at sharp ridges and corners for the geometric construction of PN triangles, we leverage the results of our medial quadric. Given an edge τ, let $\boldsymbol{p}$ be a control point on τ, vertex v the end-point of τ closer to $\boldsymbol{p}$, and vector $\boldsymbol{v}$ its coordinates. Suppose τ and its end-points have been classified by feature detection. As illustrated in Fig. 5, we evaluate the displacement $\boldsymbol{f}$ at $\boldsymbol{p}$ as follows:

1. If v is at a smooth point, then project $\boldsymbol{p}$ onto the tangent plane at v, i.e., $\boldsymbol{f} = (\boldsymbol{v} - \boldsymbol{p})^T \hat{\boldsymbol{n}} \hat{\boldsymbol{n}}$ (c.f. Fig. 5(left)), where $\hat{\boldsymbol{n}} = \hat{\boldsymbol{e}}_1$.
2. If both v and τ are on a ridge (c.f. Fig. 5(middle)), then project $\boldsymbol{p}$ onto the tangent line of the ridge at v, i.e., $\boldsymbol{f} = (\boldsymbol{v} - \boldsymbol{p}) - (\boldsymbol{v} - \boldsymbol{p})^T \hat{\boldsymbol{e}}_3 \hat{\boldsymbol{e}}_3$.
3. If v is on a ridge but τ is not, then project $\boldsymbol{p}$ onto the one-sided tangent plane at v (c.f. Fig. 5(left), i.e., $\boldsymbol{f} = (\boldsymbol{v} - \boldsymbol{p})^T \hat{\boldsymbol{n}}_+ \hat{\boldsymbol{n}}_+$, where $\hat{\boldsymbol{n}}_+$ is the one-sided normal.
4. If both vertices of τ are corners, then consider the edge as straight and take $\boldsymbol{f} = \boldsymbol{0}$.
5. If v is at a corner and the other vertex u of τ is not (c.f. Fig. 5(right)), then compute $\boldsymbol{f}$ as the displacement $\boldsymbol{g}$ of u mirrored against the normal plane of τ, i.e., $\boldsymbol{f} = \boldsymbol{g} - 2\boldsymbol{g}^T \boldsymbol{t}\boldsymbol{t}$, where $\boldsymbol{t}$ is the tangent of τ.

The first two cases are equivalent to the projections in [18], but Cases 3 through 5 are introduced here to avoid large errors near features. In Case 5, the mirroring operation allows higher-order reconstruction at a corner.

5 Parallel Surface Mesh Smoothing

We now leverage the above techniques to address the problem of parallel surface mesh smoothing, i.e., to achieve better mesh quality by redistributing smooth or ridge vertices while preserving the shape and features of a surface.

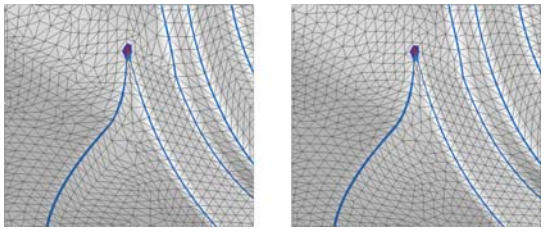

Fig. 6. Before and after smoothing of fandisk model

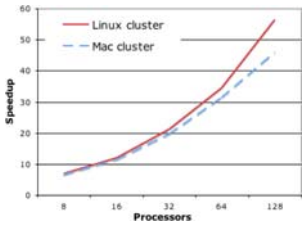

Fig. 7. Parallel performance

Smoothing Algorithm. Given a smooth or ridge vertex v, we project its incident faces (i.e., its *star*) onto its tangent space, i.e., the space spanned by $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$ if v is at a smooth point, or $\hat{\boldsymbol{e}}_3$ if v is on a ridge. Let $\boldsymbol{T}$ be a rectangular matrix, whose column vectors form the orthonormal vector base of the tangent space. We perform a coordinate transformation locally so that v becomes the origin. The projection of a neighbor vertex (denoted by $\boldsymbol{u}_i$) of v onto the tangent space is $\boldsymbol{T}^T(\boldsymbol{u}_i - \boldsymbol{v})$. We compute the *center* of the star of v in the tangent space, where the definition of center depends on the specific metric in use, but is typically a weighted sum of the vertices in its star [4], i.e.,

$$\boldsymbol{d} = \left(\sum_i w_i \boldsymbol{T}^T (\boldsymbol{u}_i - \boldsymbol{v}) \right) / \sum_i w_i, \tag{10}$$

where the w_i are metric-dependent weights. If v is at a smooth vertex, we set the weights to the sum of the edge angles at v in the incident faces of edge vu_i. If v is a ridge vertex, then we set the weights for its neighbor ridge vertices along the ridge to 1 and those for its neighbor smooth vertices to 0.

After obtaining $\boldsymbol{d}$, in general we move v for a fraction of $\boldsymbol{d}$, say $a\boldsymbol{d}$. To avoid fold-over of triangles, we choose a to be ≤ 0.5 and to be small enough so that the barycentric coordinates of $a\boldsymbol{d}$ corresponding to the other two vertices in each triangle are no greater than $c/3$ for some $c \in (0, 1)$ (say $c = 0.5$). To utilize the higher-order constructions, in particular the PN triangles, we locate the triangle σ that contains the new point $\boldsymbol{p} = \boldsymbol{v} + a\boldsymbol{d}$ and then map $\boldsymbol{p}$ onto the Bézier patch constructed by feature-preserving PN triangles. Because the new positions of smooth vertices may depend on those of ridge vertices but not vice versa, we first perform vertex redistribution within ridges and then redistribute smooth vertices using the new locations of ridge vertices. A simple Jacobi iteration is adopted within either redistribution stage. When performing smoothing for multiple iterations, we interpolate the normals using the quadric reconstruction for better efficiency. Fig. 6 shows the fandisk model (c.f. Fig. 4) near the junction before and after smoothing, where the dark curves indicate the detected features. The shapes of the triangles were improved noticeably without distorting the features.

Parallel Implementation. In large-scale computational applications, mesh smoothing, including feature detection and surface projection, must be performed on a mesh that is partitioned and distributed across multiple processors on a parallel machine. The techniques and algorithms presented above are all easily parallelized, as we have algorithmi-

cally localized their calculations, of which the most noteworthy are classification of feature vertices and calculation of one-sided normals. These operations are difficult to compute for vertices along partition boundaries using traditional methods, unless a process has access to the remote faces, vertices, and feature edges next to its boundary vertices.

Our algorithms do require a few communication steps, all of which are reduction operations on the vertices shared by more than one process along partition boundaries. These include the summation operations in the construction of $\boldsymbol{A}$ and $\boldsymbol{b}$ for the medial quadric and in the numerator and denominator in Eq. (10) for vertex redistribution. In addition, classification of ridge edges requires reduction to the maximum and minimum values of s_σ for shared vertices. To broadcast the displacements of each shared vertex in its containing PN triangle, we first zero out the displacements for shared vertices without a local containing triangle on each process, and then reduce to the values of the largest magnitude for each component.

Fig. 7 shows the scalability results of our straightforward parallel implementation for a fixed problem with a total number of 30768 vertices and 59236 triangles. The experiments were conducted on a Linux cluster with dual 1GHz Pentium III processors per node and Myrinet interconnection, and on a faster Mac cluster with dual 2GHz G5 processors per node and also Myrinet interconnection, both located at the University of Illinois. Our method delivers nearly linear scalability for this modest size problem up to 128 processors, and better scalability was achieved on the Linux cluster due to higher ratio of bandwidth to processing power. Better scalability is expected for larger problems and further optimization of the implementation.

6 Conclusion

We have developed a parallel method for surface mesh smoothing based on a new concept called the medial quadric. This quadric facilitates the solution of a number of geometric primitives, including a reliable vertex-based scheme for feature detection, which is easier to parallelize than edge-based schemes, and accurate one-sided normal estimation. These primitives are then used to construct feature-aware higher-order approximation for surface triangulations based on PN triangles. We presented some preliminary but promising experimental results of our surface mesh smoothing algorithm. We are currently conducting more tests, especially for distributed meshes on parallel machines, and integrating our methods into large-scale scientific simulations at the Center for Simulations of Advanced Rockets at University of Illinois [7, 8]. Future directions include more extensive experimental study of our algorithms, detailed comparison against other methods, systematic analysis of normal estimation and feature detection schemes, and extension to estimation of curvatures.

Acknowledgments

Supported by the U.S. Department of Energy through the University of California under subcontract B523819, and in part by NSF and DARPA under CARGO grant #0310446. We thank Prof. John Hart for helpful discussions and references on curved PN triangles,

Prof. Michael Garland for discussions on quadrics, and Prof. Herbert Edelsbrunner for suggestions on enhancing the rigorousness of the paper.

References

1. T. Baker. Identification and preservation of surface features. In *13th Int. Meshing Roundtable*, pages 299–310, 2004.
2. H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.
3. G. Farin. Smooth interpolation to scattered 3D data. In R. E. Barnhill and W. Boehm, editors, *Surfaces in Computer-Aided Geometric Design*, pages 43–63, 1983.
4. P. J. Frey. *Mesh Generation: Application to finite elements*. Hermes, 2000.
5. H. Gouraud. Continuous shading of curved surfaces. *IEEE Trans. Computers*, 20:623–629, 1971.
6. M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw–Hill, New York, 2nd edition, 2002.
7. M. T. Heath and W. A. Dick. Virtual prototyping of solid propellant rockets. *Comput. Sci. & Engr.*, 2:21–32, 2000.
8. M. T. Heath and X. Jiao. Parallel simulation of multicomponent systems. In *6th Int. Conf. on High Performance Computing for Computational Science*, Valencia, Spain, 2004.
9. P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom.*, pages 49–65, 1999.
10. X. Jiao and M. T. Heath. Feature detection for surface meshes. In *Proc. of 8th Int. Conf. on Numerical Grid Generation in Computational Field Simulations*, pages 705–714, 2002.
11. N. Max. Weights for computing vertex normals from facet normals. *J. Graphics Tools*, 4:1–6, 1999.
12. G. Medioni, M.-S. Lee, and C.-K. Tang. *A computational framework for segmentation and grouping*. Elsevier, 2000.
13. D. Meek and D. Walton. On surface normal and Gaussian curvature approximations of given data sampled from a smooth surface. *Comput. Aid. Geom. Des.*, 17:521–543, 2000.
14. 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.
15. S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Comput. Surv.*, 34:211–262, 2002.
16. 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.
17. G. Thürmer and C. A. Wüthrich. Computing vertex normals from polygonal facets. *J. Graphics Tools*, 3:43–46, 1998.
18. A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN triangles. In *Proc. of 2001 Symposium on Interactive 3D graphics*, pages 159–166, 2001.