

Smoothing of Noisy Laser Scanner Generated Meshes Using Polynomial Fitting and Neighborhood Erosion

Miguel Vieira

Kenji Shimada

e-mail: shimada@cmu.edu

Tomotake Furuhashi

The Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213

This paper presents a method for removing geometric noise from triangulated meshes while preserving edges and other intended geometric features. The method iteratively updates the position of each mesh vertex with a position lying on a locally fitted bivariate polynomial. The user selects the width of the vertex neighborhood, the order of the fitted polynomial, and a threshold angle to control the effects of the smoothing operation. To avoid smoothing over discontinuities, the neighborhood can be eroded by removing vertices with normals that deviate beyond a threshold from the estimated median normal of the neighborhood. The method is particularly suitable for use on laser scanner generated meshes of automobile outer body panels. Smoothing methods used on these meshes must allow C^2 continuous equilibrium surfaces and must minimize shrinkage. Despite the abundance of existing smoothing schemes, none addresses both of these specific problems. This paper demonstrates the effectiveness of our method with both synthetic and real world examples. [DOI: 10.1115/1.1737381]

1 Introduction

This work is motivated by the errors associated with collecting digital data from a real world object with laser scanners. In particular, industry is increasingly using laser scanners to convert physical prototypes made by designers into computer models for design, analysis, and manufacturing. These digital representations take the form of dense point clouds that are often converted to triangular meshes. This is both a boon and a bane. On one hand, the data set is a large and detailed digital model of the object being investigated. On the other hand, it can be so large it strains computational ability and it is often riddled with both geometric and topological noise. The laser-scanned data often requires significant preprocessing before it can be used in the design process.

This paper addresses the issue of geometric noise in meshes created from laser-scanned data. Geometric noise is defined as small deviations in the positions of scanned data points from their actual positions and has several potential sources [1]. Noise can arise from the physical features, such as shape, texture, and spectral properties, of the object being scanned. It can also arise from internal properties of the scanner such as lens quality, CCD array capabilities, and calibration. External variables including environmental light can affect scan quality. Finally, the process of assembling multiple scans, called registration, can cause noise.

Although eliminating all the sources of noise is impossible, one can often put bounds on the accuracy and precision of the scanning equipment. The goal of this work was to develop a method for removing noise from, or smoothing, meshes created from laser-scanner data while preserving salient surface features. The industrial design nature of the problem prohibits any methods that may severely deform the data. It is therefore necessary that any smoothed version of the noisy mesh deviate as little as possible from the scanner output. This is a significant restriction because many existing methods have a strong tendency to shrink the mesh, as discussed in detail in the related work section.

The principal of simplest shape plays an important role in automobile body design. It essentially states that beautiful objects

are free of inessential features and are simple in design [2]. With regard to removing noise from scanned automobile panels, the principal of simplest shape implies that smoothing artifacts such as waviness or lumpiness should be eliminated to the greatest possible extent. Such artifacts are clearly visible by plotting mean curvature or reflection lines on the surface, although such plots are rarely shown in the smoothing literature. Figure 1 demonstrates the difference between reflection line simulations and mean curvature plots on noisy and smoothed meshes. The smoothed result is after ten iterations of our operator, varying the neighborhood width for best results.

More so than in most engineering design, smooth reflection lines are especially important for automobile panels. At any point on a surface, the reflection lines are C^{k-1} continuous if the surface is C^k continuous [1]. Therefore, since visually smooth, or C^1 , reflection lines are a necessity on an automobile body, the body itself must be at least C^2 almost everywhere. A mesh of course, being piecewise linear, is not truly differentiable. However, one can insist that the vertices of a mesh lie on a surface which is differentiable. In this work, a mesh is said to be, say, C^2 at a point if the positions of the vertices near that point are on a C^2 surface.

That an automobile body should be at least C^2 almost everywhere is a significant restriction because most existing smoothing methods do not have C^2 equilibrium surfaces. When smoothing with a given operator, if the vertices of different meshes always move toward positions lying on surfaces with certain properties, it is said that such surfaces are the equilibrium surfaces of the smoothing operator. Some examples of equilibrium surfaces are surfaces minimizing area or bending energy. The equilibrium surfaces of a smoothing operator are also the types of surfaces on which the operator will have no effect.

A distinction must be made here between the terms smoothing and fairing. A surface is said to be smooth if it is continuously differentiable and fair if it is aesthetically pleasing. Smoothness is necessary but not sufficient for fairness. For example, a surface can be differentiable everywhere and thus smooth, but still have arbitrarily many saddle points that make it lumpy, violating the principal of simplest shape. Fairness is typically quantified by integrals of some intrinsic geometric property over the surface [2].

Contributed by the Design Automation Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received May 2003; revised October 2003. Associate Editor:

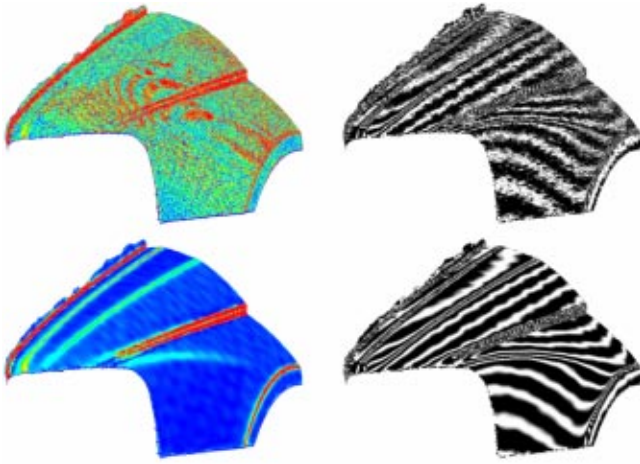


Fig. 1 Front side panel of an automobile. The top row shows the original, noisy data and the bottom row shows the smoothed data. The left column shows a plot of the mean curvature and the right column shows the simulated reflection lines.

All the schemes discussed are smoothing schemes, but it is assumed that, in the limit, they fair the surfaces they are operating on.

This work considers automobile bodies to be continuously differentiable almost everywhere. That is, there are very small areas on automobile bodies with discontinuous derivatives. Automobile panels often consist of many large, simple surfaces which can locally be described by polynomials. These different surfaces are sometimes connected continuously, but can also be connected along sharp edges. These sharp edges are called character lines and are frequently exploited by designers to give automobiles distinctive appearances, or character. Unfortunately, existing mesh smoothing methods either round out sharp edges or preserve them at the expense of less powerful smoothing elsewhere. By distinguishing between differentiable and non-differentiable points on a surface, powerful smoothing can be applied where it is appropriate, leaving edges and character lines sharp.

Summarizing, several requirements arise in the smoothing of automobile panels that demand special treatment:

- Displacement of data points from their original positions must be minimal.
- Reflection lines and curvature plots must be smooth.
- The equilibrium surfaces of the smoothing operator must have C^2 continuity.
- Edges and character lines must be preserved.

As explained in the Related Works section, diffusion and curvature flow based operators intrinsically shrink surfaces, and either different frameworks or computational duct taping are required to preserve mesh size. The framework presented here allows a far larger set of equilibrium surfaces than existing methods, preventing shrinkage and preserving edges in several cases. In addition, because this method approximates the surface with local polynomials, smooth reflections and discrete C^k differentiability can be enforced almost everywhere.

This paper is organized as follows. Section 2 discusses related work and examines the definitions and properties of various existing smoothing operators. Section 3 develops the framework for the polynomial fitting approach by explaining the necessary ideas and mathematics of differential geometry. Section 4 describes the application of the differential geometry to the smoothing of noisy surfaces. Section 5 discretizes the previous concepts and presents the method. Results are presented in Section 6.

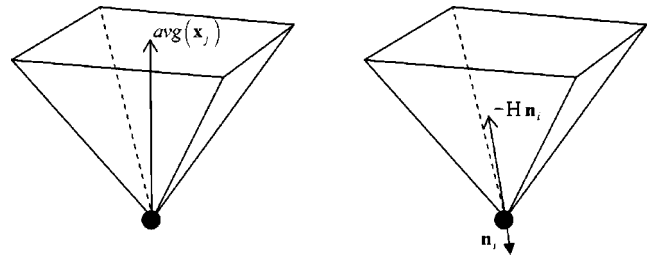


Fig. 2 Laplacian and curvature flow operators: the Laplacian operator (left) moves each vertex to the spatial average of its neighbors; the curvature flow operator (right) moves each vertex a distance equal to its estimated mean curvature in the direction opposite its normal.

2 Related Work

Standard smoothing operators locally satisfy some minimizing functional that promotes the principal of simplest shape. Local discrete operators are in favor for several reasons. First, as described in the Introduction, the large numbers of vertices created by laser scanning make global minimization algorithms prohibitive. Furthermore, local discrete operators are easy to implement, are efficient, and sometimes yield readily to mathematical formulation and analysis. Most methods are based on either diffusion or curvature flow, although Vieira [3] introduced the concept of finding new vertex positions using least squares fit polynomials. Diagrams giving qualitative interpretations of these operators are shown in Figs. 2 and 3. One typically implements them by iteratively updating the position of each vertex using a simple rule that is a discretization of the minimizing functional. The iterative process can take the form of a Jacobi method, where all the vertex positions are updated simultaneously, or a Gauss-Seidel method, where the newest vertex positions are used for updating each vertex position. Multigrid methods, which can speed up smoothing by creating a mesh hierarchy and smoothing at coarse levels to obtain initial conditions for smoothing at finer levels also enjoy popularity [4,5]. Boundary conditions can be implemented as constraints on vertex positions, their normal vectors, or even curvatures [6]. The topology of the mesh is assumed to be given and generally does not change during smoothing.

Diffusion flow (Fig. 2) iteratively moves each vertex \mathbf{x}_i to a weighted average of the positions of its immediate neighbors,

$$\mathbf{x}_{i,new} = \mathbf{x}_i + t \left(\sum_{j \in N_1(i)} w_j \mathbf{x}_j - \mathbf{x}_i \right), \quad (1)$$

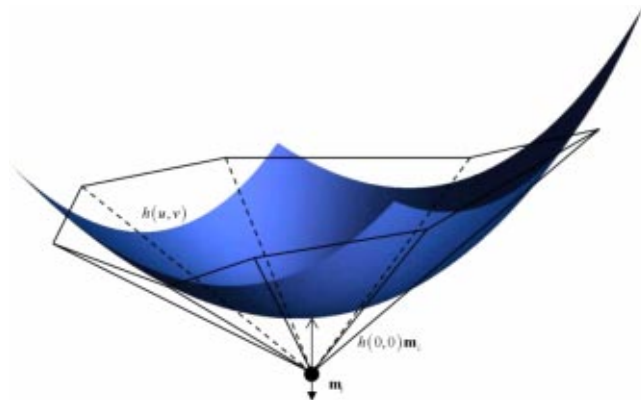


Fig. 3 Polynomial smoothing operator: move each vertex in its median normal direction to a locally fitted polynomial $h(u, v)$.

where $N_1(i)$ is the 1-ring neighborhood—the set of the indices of all the vertices topologically adjacent to vertex i and w_j are the weights, which must sum to 1, associated with each adjacent vertex. The time step is chosen such that $0 < t < 1$ to ensure stability. The well known Laplacian operator is a type of diffusion flow that is a linear approximation of the Laplace equation $\nabla^2 f = 0$. This involves setting the $w_j = 1/|N_1(i)|$ and $t = 1$ so that each vertex is moved to the barycenter of its neighbors, locally satisfying Laplace's equation. As one would expect, various other weighting schemes exist. For example, in [7] the weights enable parameterization independent smoothing with no tangential drift of vertices, in [5] the weights are linear analogues of the dihedral angles between incident triangles, and in [8] the weights are inversely proportional to incident edge lengths.

Mean curvature flow (Fig. 2) moves each vertex in its normal direction a distance equal to the negative of its mean curvature [7]:

$$\mathbf{x}_{i,new} = \mathbf{x}_i - H\mathbf{n}_i. \quad (2)$$

There are various ways to estimate the mean curvature H on a mesh. Some of these are discussed in the Results section. Desbrun, et al., do not calculate the mean curvature and surface normal separately, but instead compute the quantity $H\mathbf{n}_i$ as a function of the positions of the vertices adjacent to \mathbf{x}_i [7]. Because noise can be seen as high-curvature points on the mesh, this technique rapidly eliminates it. For example, in flat areas where there is zero mean curvature, the vertices do not move, but in areas where either principal curvature is higher, the vertices move a greater distance along their normal direction, decreasing the curvature. Furthermore, assuming that the mean curvature and surface normal calculation are robust, this operator has the advantage of being mesh parameterization independent.

Although they possess desirable properties, both diffusion and curvature flow operators have an intrinsic bias toward planar geometry. Let us consider diffusion flow. By moving each vertex to its neighborhood barycenter, it forces $\nabla^2 \mathbf{x}_i = 0$ to be discretely approximated locally. This means each vertex is moved so that it is as coplanar as possible with its neighbors. In fact, the equilibrium surface of the Laplacian operator is the minimal area surface satisfying given boundary conditions. Likewise, mean curvature flow, which is also known as the natural surface area decreasing flow [9], converges toward a discrete minimal surface satisfying $H = 0$ [1]. This condition is satisfied exactly only at saddle points and on planes. Unfortunately, this means it shrinks the mesh and smoothes out geometric features like edges and corners, so the user must decide when to stop the iteration, getting a compromise between smoothing and shrinking.

Because diffusion and curvature flow are shrinking operators with smoothing tendencies, most work related to them focuses on shape and size preservation. Taubin combines the Laplacian operator and ideas from signal processing to extend the application of various well known filters to meshes and develops fast, non-shrinking methods [10,11]. Desbrun rescales the mesh to preserve volume [7]. Ohtake avoids shrinkage by not moving vertices when their displacement would be below a certain threshold, enforcing convergence [12]. Clarenz detects edges and then enhances them by applying an anisotropic diffusion tensor where they occur [9]. Vollmer moves each vertex back toward its previous location to preserve size [13].

The equations for diffusion and curvature given above are first order schemes. Higher order methods with larger classes of equilibrium surfaces have been developed, but they can be unwieldy. The natural extension of the Laplacian operator given above, called the bilaplacian operator, generates a discrete surface minimizing the equation $\nabla^2 f = 0$. One implementation results from giving both coefficients equal absolute value in Taubin's $\lambda|\mu$ algorithm [11,4]. Chopp and Sethian offer a method for simulating the evolution of a surface moving under the intrinsic Laplacian of curvature by tracking a level set representing a higher dimensional

embedding of the surface [14]. Welch and Witkin approximate the total curvature $\int_A \kappa_1^2 + \kappa_2^2 dA$ at each vertex and minimize it by appropriate vertex displacements [15]. Schneider and Kobbelt minimize the partial differential equation $\Delta_B H = 0$, the Laplace-Beltrami Operator acting on the mean curvature, which has equilibrium surfaces of constant mean curvature such as spheres and cylinders [16].

Some recent work in mesh noise removal focuses on smoothing the face normal field of the mesh and then reconstructing the mesh to best fit the modified normal field. This approach makes it possible to extend the established and well-developed tools available for signal and image processing to more general mesh processing. Traditional image processing operators such as Gaussian filters [17], mean and median filters [18], and unsharp masking [19,20] have been effectively adapted to meshes. There are various implementations of the basic approach. Tasdizen, et al., [19] offer a parameterization-independent method using level sets to modify the surface. Taubin [21] presents a least-squares approach to reconstructing mesh vertex positions from a given field of face normals. Belyaev and Ohtake give an informative summary of the approach used in their work in [22]. Finally, Yamada et al. [23] and Karbacher and Häusler [24] present early methods that foreshadow the later work. Mesh processing by first modifying the normal field and then reconstructing the surface to fit the new normal field is useful and interesting theoretically, but such approaches have difficulty obtaining smooth reflection lines on meshes or smooth mean curvature plots.

Because first order operators are not adequate to the task of smoothing automobile outer body panel meshes, this work presents a higher order scheme as well. The implementation, as will be shown, is straightforward. Each vertex is moved in its median normal direction, to be defined,

$$\mathbf{x}_{i,new} = \mathbf{x}_i + h(0,0)\mathbf{m}_i \quad (3)$$

onto a bivariate polynomial $h(u,v)$ locally fitted to the vertex neighborhood. This process is illustrated by the diagram in Fig. 3. The user selects the order of the polynomial, the width of the neighborhood, and a threshold for estimating the median normal. The mesh is then made to conform to a polynomial of the chosen order at each vertex. This permits convergence of the mesh vertices toward positions lying on surfaces with non-zero partial derivatives and also minimizes shrinkage. Indeed, the vertices will locally converge toward equilibrium surfaces that have derivatives of the same order as the fitted polynomial.

Selecting the polynomial order, neighborhood width, and threshold angle gives the user considerable control over the smoothing process. Selecting the polynomial order controls which kind of equilibrium surfaces are allowed. For example, a first order bivariate polynomial, a plane, would have an effect similar to the Laplacian operator, causing the mesh to converge toward planar geometry. A quadratic polynomial would move vertices so they locally lie on quadratic surfaces, making the mesh C^1 in a discrete sense. For meshes with details on the order of a few edge lengths, a small neighborhood width should be used. For very fine meshes, on the other hand, a larger neighborhood width is more appropriate. Finally, the threshold angle lets the user control what kinds of edges will be smoothed over and what kinds will be preserved.

3 Preliminaries: Differential Geometry

The framework presented in this paper rests on the assumption that the mesh represents a sampling of a piecewise differentiable manifold S_0 in \mathbb{R}^3 , here called the original surface, with noise added to each point. Once this assumption is made, the machinery of differential geometry can be invoked to formulate an operator for removing the noise. What follows is a discussion of those aspects that are relevant to the method.

A subset $S \subset \mathbb{R}^3$ is called a regular parametric surface of class C^m , $m \geq 1$, if, for every point $\mathbf{p} \in S$, there exists a neighborhood N of \mathbf{p} in \mathbb{R}^3 and a mapping $\mathbf{x}: U \subset \mathbb{R}^2 \rightarrow N \cap S$ expressed by

$$\mathbf{x} = \mathbf{x}(\mathbf{u}) = [x(u,v) \quad y(u,v) \quad z(u,v)]^T \quad (4)$$

of an open set $U \subset \mathbb{R}^2$ onto $N \cap S$ such that:

- (i) all partial derivatives of x, y, z of order m or less are continuous in U and
- (ii) $\mathbf{x}_u \times \mathbf{x}_v \neq [0, 0, 0]^T$ for all $(u, v) \in U$.

The mapping \mathbf{x} is called a parameterization of S at \mathbf{p} . Essentially, a regular surface is defined as a two-dimensional subset of three-dimensional space such that every point on the surface has a neighborhood that can be mapped into two-dimensional Euclidean space. The condition (ii) implies the existence of a tangent plane for every point on the surface. It can be shown that the transition from one parameterization to another is a diffeomorphism, a differentiable mapping between manifolds with a differentiable inverse. This intuitively means that a regular surface is a union of open sets of \mathbb{R}^2 organized so that the change between two neighboring sets that intersect each other is smooth [25].

At any differentiable point on the surface S_0 , then, a plane can be defined which is tangent to the surface at that point. The tangent plane $T_{\mathbf{p}}(S)$ at a point $\mathbf{p} = \mathbf{x}(\mathbf{u})$ on a regular parametric surface is the span of the two vectors $\mathbf{x}_u, \mathbf{x}_v$:

$$T_{\mathbf{p}}(S) = \mathbf{p} + a\mathbf{x}_u(\mathbf{u}) + b\mathbf{x}_v(\mathbf{u}), \quad a, b \in \mathbb{R}. \quad (5)$$

The surface normal $\mathbf{n} = \mathbf{n}(\mathbf{u})$ is a unit vector orthogonal to $T_{\mathbf{p}}(S)$. At point \mathbf{p} it is given by

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}. \quad (6)$$

To study the properties of the surface at differentiable point \mathbf{p} , choose a local Euclidean coordinate system with its origin at \mathbf{p} , \mathbf{n} as the w axis, and $T_{\mathbf{p}}(S)$ as the uv plane. A Monge patch can now be defined at each point of the surface. That is, for a point \mathbf{p} a small neighborhood can be represented around it by $\mathbf{y}(u, v) = (u, v, h(u, v))$ where $h(u, v)$ is some bivariate function and $\mathbf{p} = \mathbf{y}(0, 0) = (0, 0, 0)$. One can show that $h(u, v)$ is at least C^k for any regular parametric surface $\mathbf{x}(\mathbf{u})$ of class $m \geq k$ [26]. More specifically, considering terms only up to degree $n \leq m$, h has a Taylor expansion in two variables given by

$$h(u, v) = \sum_{k=2}^n \frac{1}{k!} \left(u \frac{\partial}{\partial u} + v \frac{\partial}{\partial v} \right)^k h(0, 0), \quad (7)$$

a bivariate polynomial of order n . Because of the coordinate system position and orientation, there are no 0th or 1st order terms. Specifically, the 0th order term is absent because the origin of the local coordinate system is at \mathbf{p} and the 1st order terms are both nil because directions of the w axis and the surface normal \mathbf{n} are the same. It is possible to orient the uv axes to simplify some calculations, but that restriction will not be made here.

When a manifold consists of differentiable surfaces joined along their boundaries, it is possible that not all first partial derivatives exist along a boundary. The potential exists for tangent plane discontinuities at the boundaries. Consider a point \mathbf{p} lying on such a discontinuity between two surfaces. The infinitesimal neighborhood around this point contains two areas that belong to smooth surfaces and a curve along which these areas are connected. All points on either side of this curve have Taylor expansions and surface normals, but the points on it do not, since the manifold is not differentiable along the curve. Therefore, there is no single polynomial surface that can describe the surfaces adjacent to \mathbf{p} .

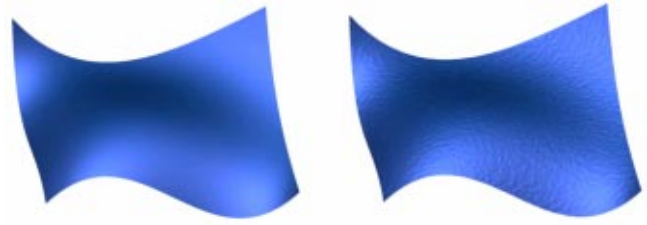


Fig. 4 The ideal mesh M_0 , left, whose vertices lie precisely on a regular parametric surface S_0 , and the noisy mesh M , right.

4 Smoothing Using Locally Fitted Polynomials

This section argues that, given a noisy surface known to be generated from a piecewise polynomial surface, the best approximation for a point on the smooth surface is given by least squares fitting a bivariate polynomial to a small neighborhood of that point.

Assume there is some ideal mesh, whose exact properties are unknown, with vertex positions corresponding perfectly to the real world object and a noisy mesh which has been imperfectly generated. An example of two such meshes is given in Fig. 4. The goal is to modify the positions of the noisy mesh vertices so they approximate the ideal mesh. Let M represent the noisy mesh created from the scanned data and M_0 represent the ideal mesh with identical topology whose points lie exactly on the scanned real world object. The vertex positions of M_0 cannot be known and therefore must be approximated. The vertices of M_0 are assumed to lie on a piecewise regular parametric surface. It is known that the neighborhood of any differentiable point on S_0 can be locally approximated by a Taylor series of given order and that the transition from such a point to a neighboring point is smooth. Therefore, the differential geometry implies that the vertices in the small neighborhood around each vertex of M_0 will lie on a locally oriented bivariate polynomial.

When formulating an update rule for each vertex, it should be moved toward its equivalent position in M_0 because the noisy mesh M is the ideal mesh M_0 with noise added to every vertex. Every point in M deviates slightly from S_0 . The best approximation of the updated position is given by estimating the locally oriented bivariate polynomial for the neighborhood. Therefore, each vertex is moved in its normal direction to a point that is the best approximation of a smooth surface. One could argue that the shortest distance to the polynomial could be used, but this is computationally expensive and there is no compelling reason to do so. The proposed method works well in practice and simplifies implementation and analysis.

As described above, automobile body panels consist of many polynomial surfaces that can be connected with or without tangent plane continuity. They are thus piecewise differentiable manifolds. A scheme that effectively smoothes polynomial surfaces can be used safely on any single surface, but care must be taken near lines where surfaces are connected. Smoothing over such lines will decrease the quality of the smoothing on both surfaces and eliminate the distinctiveness of the lines. Therefore, when updating the position of a point, the bivariate polynomial used to estimate its true position should not include points on the discontinuity or points on the other surface. Such points can reliably be found by comparing the surface normal directions of the target point and the points in the neighborhood.

The rationale behind this approach should be clear. It is assumed that the laser-generated mesh is a noisy, discrete approximation of a piecewise regular parametric surface. Based on this assumption, a smoothing scheme to restore the properties of the ideal surface can be developed. By finding the best estimate of the position of each vertex and moving the vertex to this position, the noisy mesh is forced to conform to a smooth approximation at

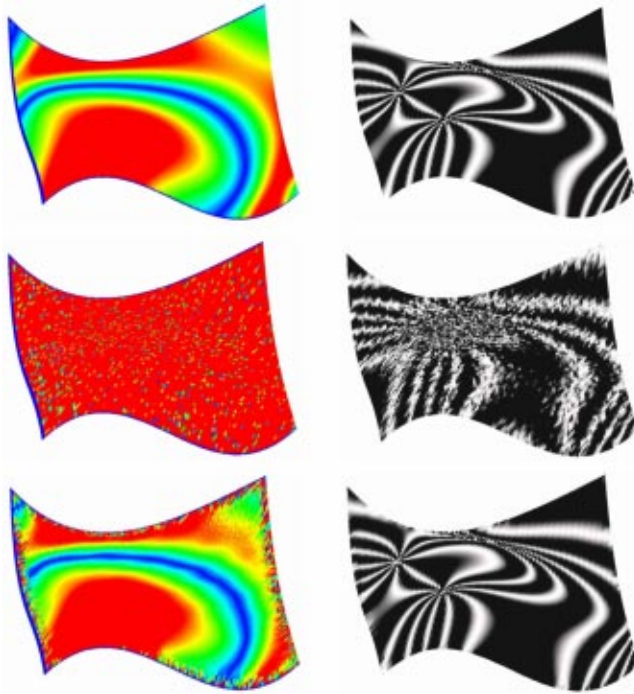


Fig. 5 The rows from top to bottom show, respectively, the original ideal mesh, the noisy mesh, and the smoothed mesh using the operator described in the paper. The roughness near the boundary is due to not moving boundary vertices. The mean curvature plot is shown on the left and the reflection line simulation is shown on the right.

each point. By choosing the order n of the fitted polynomial, partial derivatives of order higher than n are eliminated on the mesh in that each local neighborhood on the mesh is made to conform to a polynomial of degree n . At a given vertex, over successive iterations, its neighborhood will converge to points on a bivariate polynomial of order n and higher order terms will go to zero. Furthermore, the use of neighborhood widths other than unity ensures that there will be some overlap between data used for finding the respective polynomials, thus increasing the smoothing effect. The effectiveness of the method is demonstrated in Fig. 5. A comparison with the effects of different operators on the same mesh is given in the Results section.

5 Smoothing Method and Implementation

This section describes the discretization and implementation of the ideas discussed in the previous sections. The process pipeline requires finding the vertices in the user-defined neighborhood width, determining the median normal, removing the vertices in the neighborhood whose surface normals deviate from the median normal beyond the specified threshold (eroding the neighborhood), and then performing a least-squares fit on the eroded neighborhood using the median normal to define a coordinate frame. Once this is done, the fitted polynomial can be used for smoothing.

The notation is as follows. The mesh M is a set of vertices v_i , each with a position $\mathbf{x}_i = (x_i, y_i, z_i)$ defining the mesh geometry, and edges, defining the mesh topology. All the mesh properties are derived from this set. The mesh is assumed to be triangular. If this is not the case, the polygonal model can be subdivided into triangles. Methods will be presented for calculating the l -neighborhood $N_l(i)$, the median normal \mathbf{m}_i , the tangent plane $T_i(M)$, the surface normal \mathbf{n}_i , and the fitted polynomial h for each vertex i . Within the local coordinate system, vertex positions are given by $\mathbf{u}_i = (u_i, v_i, w_i)$.

The first step in the method is determining the user-defined neighborhood N of the target vertex v_i . This can be done by finding all the vertices within a given radius of the target vertex, estimating the geodesic distance between vertices using Dijkstra's algorithm and edge lengths [27]. However, because laser scanned data tends to be regular, the edge lengths of the mesh are quite similar and the problem can be simplified and computation eliminated by assuming vertex adjacency corresponds to distance. So, the l ring neighborhood of a vertex i is defined as the set of indices of all vertices connected to v_i by l or fewer edges and is denoted by $N_l(i)$. For example, $N_3(i)$ consists of all the indices for vertices connected to vertex i by three or fewer edges. Finding N is fast with standard mesh data structures. Furthermore, for a given width, the neighborhood need only be calculated once and then stored for use in succeeding iterations.

The surface normal \mathbf{n}_i must be estimated for every vertex. Let $F(i)$ be the set of triangular faces incident to vertex v_i . Then

$$\mathbf{n}_i = \frac{\sum_{j \in F(i)} \theta_j \mathbf{n}(F_j)}{\left\| \sum_{j \in F(i)} \theta_j \mathbf{n}(F_j) \right\|}, \quad (8)$$

where $\mathbf{n}(F_j)$ is the normal of face F_j , calculated by normalizing the cross product of two of its edges, and θ_j is the angle formed by the edges of F_j incident to vertex i . There are various ways of estimating the surface normal. This is a local formulation sensitive to the angles made by incident faces at the vertex.

Next, the median normal is calculated. The method for calculating the median normal given here is different from those in the literature. One method is to calculate the median normal as the direction of the Laplacian operator [12]. Another involves calculating the median normal for a face: the normals of the adjacent faces are averaged to find an average normal and then the median normal is chosen as the adjacent face normal that makes a median angle with the average normal [18]. In the method given here, the general idea is to first find an estimate of the surface normal at a vertex and then compare the angles it makes with nearby normals, be they of faces or vertices. The normal that makes the median angle is chosen as the median normal. The best results were obtained by using the surface normal of the target vertex as calculated above and the face normals of the incident triangles. The median normal \mathbf{m}_i for the target vertex is the incident face normal that makes the median angle with the target vertex normal.

When the neighborhood of the target vertex contains an edge, the median normal can be used to differentiate between vertices belonging to different surfaces. As described below, the median normal is used to remove vertices in the neighborhood that do not belong to the same surface as the target vertex when the neighborhood contains an edge. For example, consider a vertex lying next to an edge connecting two planar surfaces. The median normal at the vertex is the normal of the planar surface on which the vertex lies. It is assumed that this is true for surfaces more general than planes, and the smoothing results support this assumption.

Now that the surface and median normals have been computed, the tangent plane can be calculated for the target vertex. The tangent plane $T_i(M)$ of the target vertex is found by taking cross products to find two vectors orthonormal to the median normal. The median normal $\mathbf{m}_i = \mathbf{w}_i$ will be the w axis of the local coordinate system and will thus be orthogonal to the tangent plane, but two orthonormal vectors to define the local coordinate system are needed. The orientation of this pair of vectors is irrelevant. One can see that, by applying an arbitrary rotation to the uv axes in the equation for $h(u, v)$, the degree of the polynomial remains the same and only the coefficients change. By taking the normalized cross product of the median normal with any adjacent edge

$$\mathbf{u}_i = \mathbf{m}_i \times (\mathbf{x}_j - \mathbf{x}_i) / \|\mathbf{m}_i \times (\mathbf{x}_j - \mathbf{x}_i)\| \quad (9)$$

and then taking the cross product of the median normal and the new unit vector

$$\mathbf{v}_i = \mathbf{m}_i \times \mathbf{u}_i, \quad (10)$$

the unit vectors defining the uv axes are calculated.

It must be noted that the mesh induces an ordering on the adjacent vertices and that there may be instances when the projection onto the tangent plane will not preserve this ordering. In such a case, one can make use of the exponential map [15]. This operator preserves the adjacent edge lengths and the angles between them. However, it can only be used for a 1 ring neighborhood and is more expensive to calculate than the standard normal. If the quality of the triangular elements becomes very bad while using the polynomial smoothing described here, an application of the tangential component of the Laplacian operator will improve the vertex distribution with a small effect on the surface geometry.

The vertices in $N_l(i)$ are chosen by merely considering their connectivity with the target vertex. Thus, vertices beyond a tangent discontinuity in the mesh may have been included in the neighborhood. To avoid fitting a polynomial to vertices belonging to a different surface, the angle between the median normal and the surface normal at each vertex is found. If this angle is above the user threshold, the vertex is removed from the neighborhood, eroding it.

The positions \mathbf{u}_j of the neighboring vertices in the local coordinate system are calculated with

$$\mathbf{Q} = \begin{bmatrix} \mathbf{u}_i^T \\ \mathbf{v}_i^T \\ \mathbf{w}_i^T \end{bmatrix}, \quad \mathbf{T} = -\mathbf{x}_i, \quad (11)$$

in

$$\mathbf{u}_j = \mathbf{Q}\mathbf{x}_j + \mathbf{T}, \quad (12)$$

where \mathbf{Q} and \mathbf{T} , a rotation matrix and translation, respectively, are determined by the target vertex i and its unit normal vector, as shown above. These new positions are used to solve the overdetermined least squares problem of finding the coefficients of a bivariate polynomial given N data points in $N_l(i) \cup i$, the vertex neighborhood with index i added. Adding the index of the target vertex increases stability when the one ring neighborhood is used or when the operator is applied on mesh boundaries. The basis functions are the combinations of u and v , up to and including the order M of the polynomial being fit:

$$h(u,v) = \sum_{i=j=0, i+j \leq M} a_{ij} u^i v^j = a_{00} + a_{10}u + a_{01}v + a_{20}u^2 + a_{11}uv + a_{02}v^2 + \dots + a_{0M}v^M. \quad (13)$$

The design matrix for this least squares problem is

$$\mathbf{A} = \begin{bmatrix} 1 & u_1 & v_1 & u_1^2 & \dots & v_1^M \\ 1 & u_2 & v_2 & u_2^2 & & v_2^M \\ \vdots & & & & \ddots & \\ 1 & u_N & v_N & u_N^2 & & v_N^M \end{bmatrix} \quad (13)$$

and the equation

$$\mathbf{a} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (14)$$

must be solved, where

$$\mathbf{a} = [a_{00} \ a_{10} \ a_{01} \ a_{20} \ \dots \ a_{0M}]^T \quad (15)$$

and

$$\mathbf{b} = [w_0 \ w_1 \ \dots \ w_N]^T. \quad (16)$$

If there are not enough data points to determine the least squares problem, calculate $N_{l+1}(i) \cup i$ and then erode the neighborhood again. Solving the least squares problem can be done quickly with Cholesky factorization [28].

To smooth the mesh at the target vertex, displace it along the median normal direction onto the fitted polynomial. Because \mathbf{m}_i corresponds to \mathbf{w}_i , the new position is given in the local coordinate system by $h(0,0) = a_{00}$. This position in the global coordi-

nate system is found by inverting the rotation and translation above. Because the rotation matrix is orthogonal, $\mathbf{Q}^{-1} = \mathbf{Q}^T$, giving us

$$\mathbf{x}_{i,new} = \mathbf{Q}^T [0 \ 0 \ a_{00}]^T - \mathbf{T}. \quad (17)$$

The reader may note the similarity of this method to Savitzky-Golay smoothing filters known in spectrometric analysis and described in [28]. Indeed, the smoothing portion of this work can be viewed as an extension of the idea behind Savitzky-Golay filters to discrete regular parametric surfaces. In Savitzky-Golay filtering, a polynomial of prescribed order is least squares fit to all the points within a moving window and then the target point is moved vertically onto that polynomial. The user selects polynomial order and filter width.

In the one-dimensional setting and for regularly sampled points, Savitzky-Golay filters have the remarkable property that the coefficients of the smoothing polynomial turn out to be the same for every point and thus need only be computed once. The smoothed value of each data point is a mere weighted average of its neighbors. One may ask if the same is true on a regular parametric surface. Such an insight would make smoothing easier to implement and faster to execute. Making the assumption that the vertices in each l ring lie on a circle centered on the target vertex one can define and precompute stencils for calculating new vertex positions as a weighted average of the heights of neighboring vertices in the local coordinate system. Unfortunately, this assumption rarely ever holds and experiments have shown that the effects of its failure are quite noticeable. Furthermore, most of the computational time in the polynomial smoothing process comes from memory management, so there is no observable speed-up over actually solving the normal equations for each vertex.

6 Results

This section presents a comparison of the polynomial smoothing operator with the well-established Laplacian and curvature flow operators. The number of iterations varies for each capture so that the best-looking result is shown for each operator. For the polynomial fitting, the neighborhood width was slowly decreased from six to two to achieve smoothing over various length scales. The threshold angle was five degrees and all examples used cubic polynomials.

The mean curvature plots in the figures were calculated using the operator described in [29]. This is a local approximation in which the absolute mean curvature for each vertex is computed using only the positions of the vertices in its 1 ring neighborhood. There are various methods for calculating the mean curvature on a mesh, usually based on 1 ring approximations [30] or quadric surface fitting [31]. The method used for the results in this work was chosen because it is fast and easy to calculate, robust, and sensitive to noise.

In Figs. 6 and 8, mean curvature plots are shown on the left and reflection line plots are shown on the right. The top row shows the noisy original data. It is followed by the best results for Laplacian, curvature flow, and polynomial smoothing, in that order. The curvature plots give information on the overall smoothness of the surface, with blue areas signifying low mean curvature and red areas signifying high mean curvature. Speckles indicate waviness of the surface. The polynomial smoothing leaves less waviness on the surface than the other two methods. In addition, the high curvature areas, in particular the character lines, are left sharper than with the other two methods. The reflection lines reveal remaining waviness as well. They become smoothest with polynomial smoothing.

Figures 7 and 9 show plots of the total vertex displacement for the three smoothing methods along the top. Red indicates high displacement. The naive Laplacian smoothing generates significant displacements. The displacements resulting from curvature flow are more moderate, but the over-smoothing of character lines and edges can clearly be seen. The polynomial smoothing displays

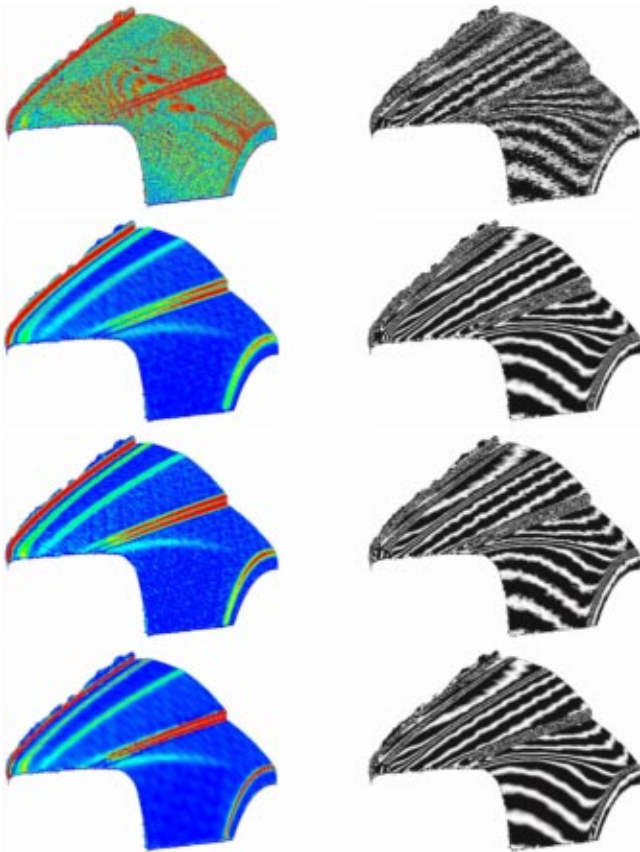


Fig. 6 Driver-side front panel of an automobile. The left column shows plots of the mean curvature and the right column shows simulated reflection lines. From top to bottom are shown: the noisy mesh followed by the results of Laplacian, curvature flow, and polynomial smoothing. The best looking results are shown for each operator.

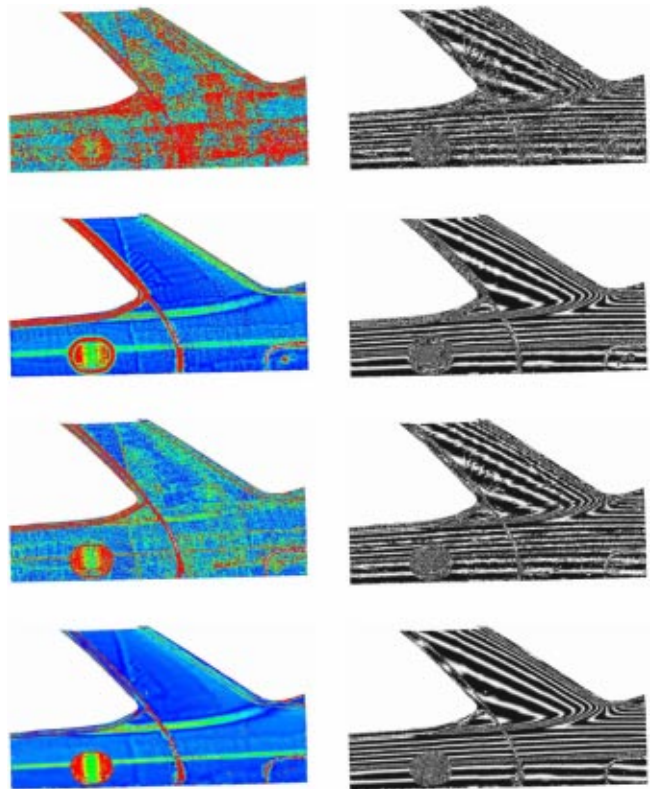


Fig. 8 Rear part of an automobile showing rear window, door handle, and gas tank. The left column shows plots of the mean curvature and the right column shows simulated reflection lines. From top to bottom are shown: the noisy mesh followed by the results of Laplacian, curvature flow, and polynomial smoothing. The best looking results are shown for each operator.

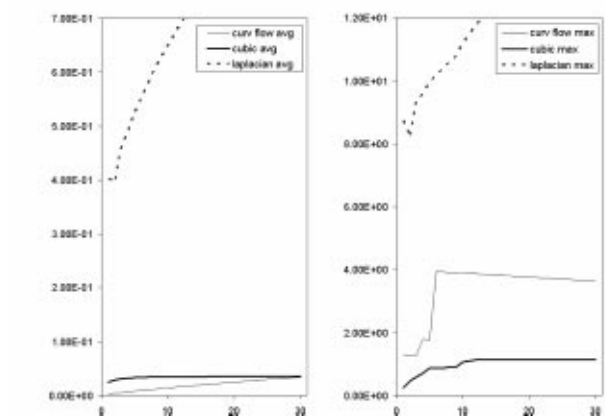
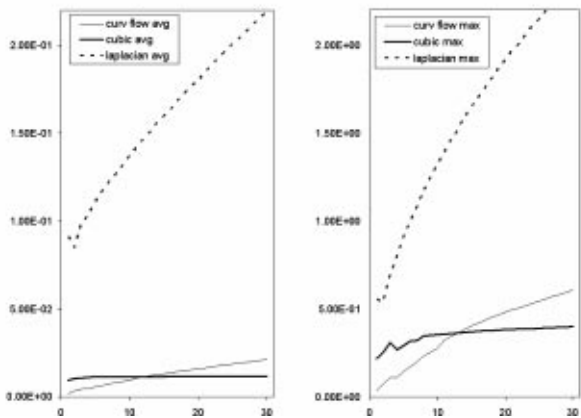
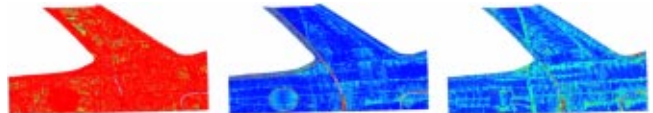
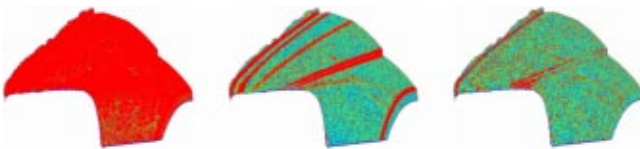


Fig. 7 The average and maximum displacement of the mesh vertices from their original positions. From left to right, the plots show Laplacian, curvature flow, and polynomial smoothing. These are for the smoothing operations shown in Fig. 8. The bottom graphs show the average and maximum displacement as a function of iterations for the different operators.

Fig. 9 The average and maximum displacement of the mesh vertices from their original positions. From left to right, the plots show Laplacian, curvature flow, and polynomial smoothing. These are for the smoothing operations shown in Fig. 10. The bottom graphs show the average and maximum displacement as a function of iterations for the different operators.

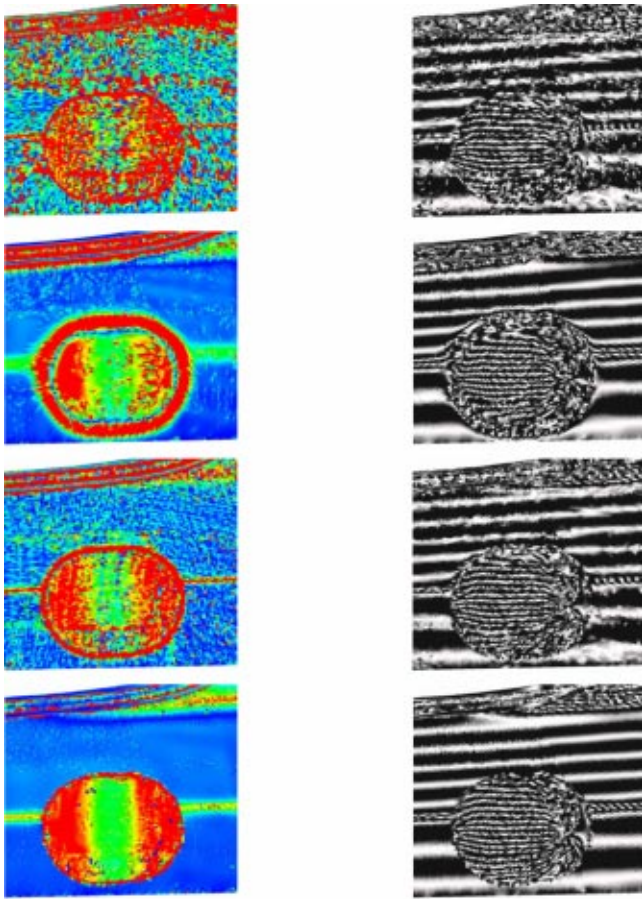


Fig. 10 Detail view of mesh shown in Fig. 10. The left column shows plots of the mean curvature and the right column shows simulated reflection lines. The original data is shown in the top row. From top to bottom are shown: the noisy mesh followed by the results of Laplacian, curvature flow, and polynomial smoothing. The best looking results are shown for each operator.

the smallest overall displacements. This is especially significant considering these smaller displacements still result in the smoothest mesh. The plots of average and maximum displacements at the bottom of the figures show how quickly the vertices drift with Laplacian smoothing. The displacements resulting from curvature flow and polynomial smoothing are comparable, but the polynomial smoothing stabilizes very quickly whereas the curvature flow deforms the mesh continuously. This demonstrates the convergent properties of the operator on piecewise differentiable manifolds.

Figures 10 and 11 show the effects of the median normal and neighborhood erosion. These are detail views of the meshes shown in Figs. 8 and 9. The most important feature of Fig. 10 is the ring surrounding the door handle. The Laplacian operator has

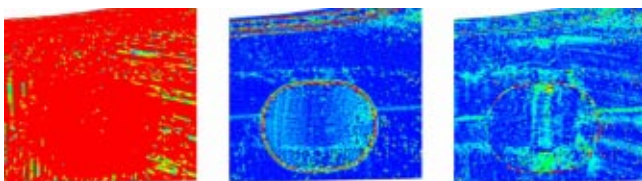


Fig. 11 The average and maximum displacement of the mesh vertices from their original positions. From left to right, the plots show Laplacian, curvature flow, and polynomial smoothing. These are for the smoothing operations shown in Fig. 10.

diffused the originally sharp feature, resulting in a wide, high-curvature ring around the handle. A similar, but less noticeable effect is shown in the curvature flow result. Polynomial smoothing, however, creates a sharp edge around the door handle and preserves the character line intersecting it. Note also the smoothness of the surrounding low curvature surfaces. Figure 11, of the total vertex displacements for polynomial smoothing, shows almost no significant displacements on the edge surrounding the handle.

7 Conclusions

This work expands the foundations, breadth, and usefulness of the smoothing technique first described in [3]. By demonstrating the basis of the polynomial fitting operator in differential geometry and then clarifying the assumptions made concerning mesh geometry, modifications presented themselves that greatly increased the effectiveness of the technique. Examples of this were shown in various figures.

CAD models used for automobile design are typically assembled from NURBS patches. Therefore, it is a reasonable assumption that the mesh of an automobile panel should be piecewise polynomial in a discrete sense. The scheme presented here is effective because it makes the mesh conform to a smooth polynomial at each vertex. The examples show that, when knowledge of the underlying geometry makes it appropriate, our method generally outperforms others.

Acknowledgments

This material is based in part on work supported under a NSF CAREER Award (No. 9985288).

References

- [1] Kobbelt, L., Bischoff, S., Botsch, M., Kahler, K., Schneider, C., and Vorsatz, J., 2000, "Geometric Modeling Based on Polygonal Meshes (tutorial)," *EUROGRAPHICS 2000*.
- [2] Sapidis, N. S., 1994, *Designing Fair Curves and Surfaces*, SIAM, USA.
- [3] Vieira, M., Shimada, K., and Furuhashi, T., 2002, "Local Least Squares Fitting for Surface Mesh Fairing in Automobile Panel Design," *ASME DETC 2002/DAC*, Montreal, Canada.
- [4] Kobbelt, L., Campagna, S., Vorsatz, J., and Seidel, H., 1998, "Interactive Multiresolution Modeling on Arbitrary Meshes," *SIGGRAPH 98*, pp. 105–114.
- [5] Guskov, I., Sweldens, W., and Schröder, P., 1999, "Multiresolution Signal Processing for Meshes," *SIGGRAPH 99*, pp. 325–334.
- [6] Hou, K.-H., 2002, "A Computational Method for Mesh-Based Free-Form Functional Surface Design," Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.
- [7] Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H., 1999, "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow," *SIGGRAPH 99*, pp. 317–324.
- [8] Fujiwara, K., 1995, "Eigenvalues of Laplacians on a Closed Riemannian Manifold and Its Nets," *Proceedings of AMS 123*, pp. 2585–2594.
- [9] Clarenz, U., Diewald, U., and Rumpf, M., 2000, "Anisotropic Geometric Diffusion in Surface Processing," *IEEE Visualization 2000*.
- [10] Taubin, G., Zhang, T., and Golub, G., 1996, "Optimal Surface Smoothing as Filter Design," IBM T.J. Watson Research, Tech. Rep. 90237.
- [11] Taubin, G., 1995, "A Signal Processing Approach to Fair Surface Design," *SIGGRAPH 95*, pp. 351–358.
- [12] Ohtake, Y., Belyaev, A. G., and Bogaevski, I. A., 2000, "Polyhedral Surface Smoothing with Simultaneous Mesh Regularization," *Geometric Modeling and Processing 2000 Proceedings*, pp. 229–237.
- [13] Vollmer, J., Mencl, R., and Muller, H., 1999, "Improved Laplacian Smoothing of Noisy Surface Meshes," *Computer Graphics Forum (Proceedings Eurographics 1999)*, 18(3), pp. 131–138.
- [14] Chopp, D. L., and Sethian, J. A., 1999, "Motion by Intrinsic Laplacian of Curvature," *Interfaces and Free Boundaries*, 1, pp. 1–18.
- [15] Welch, W., and Witkin, A., 1994, "Free-Form Shape Design Using Triangulated Surface," *SIGGRAPH 94*, pp. 247–256.
- [16] Schneider, R., and Kobbelt, L., "Geometric Fairing of Irregular Meshes for Free-Form Surface Design," *Computer Aided Geometric Design* (to appear).
- [17] Ohtake, Y., Belyaev, A., and Seidel, H., 2002, "Mesh Smoothing by Adaptive and Anisotropic Gaussian Filter Applied to Mesh Normals," *Vision, Modeling, and Visualization*, pp. 203–210.
- [18] Yagou, H., Ohtake, Y., and Belyaev, A., 2002, "Mesh Smoothing via Mean and Median Filtering Applied to Face Normals," *Geometric Modeling and Processing 2002 Proceedings*.
- [19] Tasdizen, T., Whitaker, R., Burchard, P., and Osher, S., 2002, "Geometric

- Surface Smoothing via Anisotropic Diffusion of Normals,” *IEEE Visualization 2002*.
- [20] Yagou, H., Belyaev, A., and Wei, D., 2002, “Shape Deblurring with Unsharp Masking Applied to Mesh Normals,” *Journal of Three Dimensional Images*, **16**, pp. 79–84.
- [21] Taubin, G., 2001, “Linear Anisotropic Mesh Filtering,” IBM T.J. Watson Research, Tech. Rep. RC22213 (W0110-051).
- [22] Belyaev, A., and Ohtake, Y., 2003, “A Comparison of Mesh Smoothing Methods,” *Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pp. 83–87.
- [23] Yamada, A., Furuhashi, T., Shimada, K., and Hou, K., 1998, “A Discrete Spring Model for Generating Fair Curves and Surfaces,” *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications*, pp. 270–279.
- [24] Karbacher, S., and Häusler, G., 1998, “A New Approach for Modeling and Smoothing of Scattered 3D Data,” *Three-Dimensional Image Capture and Applications, Proc. SPIE 3313*, pp. 115–125.
- [25] do Carmo, M., 1992, *Riemannian Geometry*, Birkhauser Boston, USA.
- [26] Hamann, B., 1991, “Visualization and Modeling of Contours of Trivariate Functions,” Arizona State University, Ph.D. thesis, Arizona State University, Tempe, AZ.
- [27] Cormen, T., Leiserson, C., and Rivest, R., 1990, *Introduction to Algorithms*, MIT Press, Cambridge, MA, USA.
- [28] Golub, G. H., and Van Loan, C. F., 1996, *Matrix Computations, Third Edition*, Johns Hopkins University Press, Baltimore, MD, USA.
- [29] Meyer, M., Desbrun, M., Schröder, P., and Barr, A., 2002, “Discrete Differential-Geometry Operators for Triangulated 2-Manifolds,” *VisMath 2002*, Berlin, Germany.
- [30] Lesage, D., Leon, J.-C., and Véron, P., 2001, “Discrete Curvature Approximations for the Segmentation of Polyhedral Surfaces,” *ASME DETC 2001*, Pittsburgh, PA.
- [31] McIvor, A., and Valkenburg, R., 1997, “A Comparison of Local Surface Geometry Estimation Methods,” *Machine Vision and Applications*, **10**, pp. 17–26.