EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

Curvature-Domain Shape Processing

Michael Eigensatz Robert W. Sumner Mark Pauly

Applied Geometry Group
ETH Zurich

Abstract

We propose a framework for 3D geometry processing that provides direct access to surface curvature to facilitate advanced shape editing, filtering, and synthesis algorithms. The central idea is to map a given surface to the curvature domain by evaluating its principle curvatures, apply filtering and editing operations to the curvature distribution, and reconstruct the resulting surface using an optimization approach. Our system allows the user to prescribe arbitrary principle curvature values anywhere on the surface. The optimization solves a nonlinear least-squares problem to find the surface that best matches the desired target curvatures while preserving important properties of the original shape. We demonstrate the effectiveness of this processing metaphor with several applications, including anisotropic smoothing, feature enhancement, and multi-scale curvature editing.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Curvature is an essential concept in geometry and plays a crucial role in surface optimization, geometric modeling, and shape classification. Many geometry processing operations strive to optimize the curvature distribution of a surface based on energy functionals that measure surface fairness [BPK*07]. Shape classification, feature extraction, and segmentation algorithms [Sha06, AKM*06] heavily rely on curvature information to identify meaningful geometric structures, such as ridges, valleys, and corners, that are mapped to features or used as segmentation boundaries. Similarly, non-photorealistic rendering approaches often make use of surface curvature to determine the position and style of rendered strokes [MHIL02]. Typically, these methods use curvature either as a tool for analysis, or indirectly in the optimization of fairness energies that are defined as curvature integrals over the entire surface.

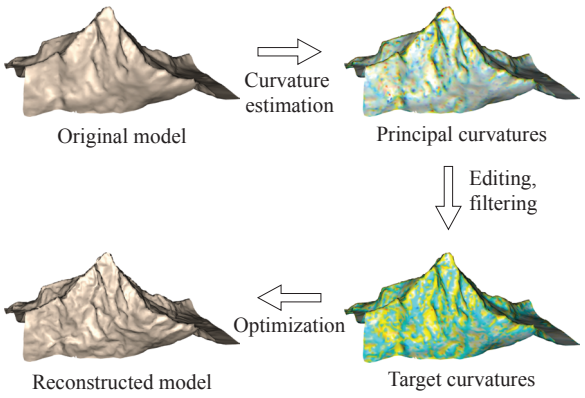

Figure 1: *The central idea of our approach is to process the geometry of a model by altering its curvature distribution.*

Our goal is to provide direct access to curvature as a geometry processing tool to facilitate advanced shape editing, filtering, and synthesis algorithms. We propose an optimization framework that allows editing and filtering of surface curvatures in order to alter the shape of an object, as illustrated in Figure 1. Our system first computes a mapping from the spatial domain to the curvature domain by evaluating principal curvatures for each point on the surface of a given object. In this domain, important geometric features and properties are more directly accessible and can be manipulated by setting specific curvatures to desired target values or by applying filtering operations on the curvature distribution. The mapping to the curvature domain is then inverted to reconstruct the modified object geometry. As

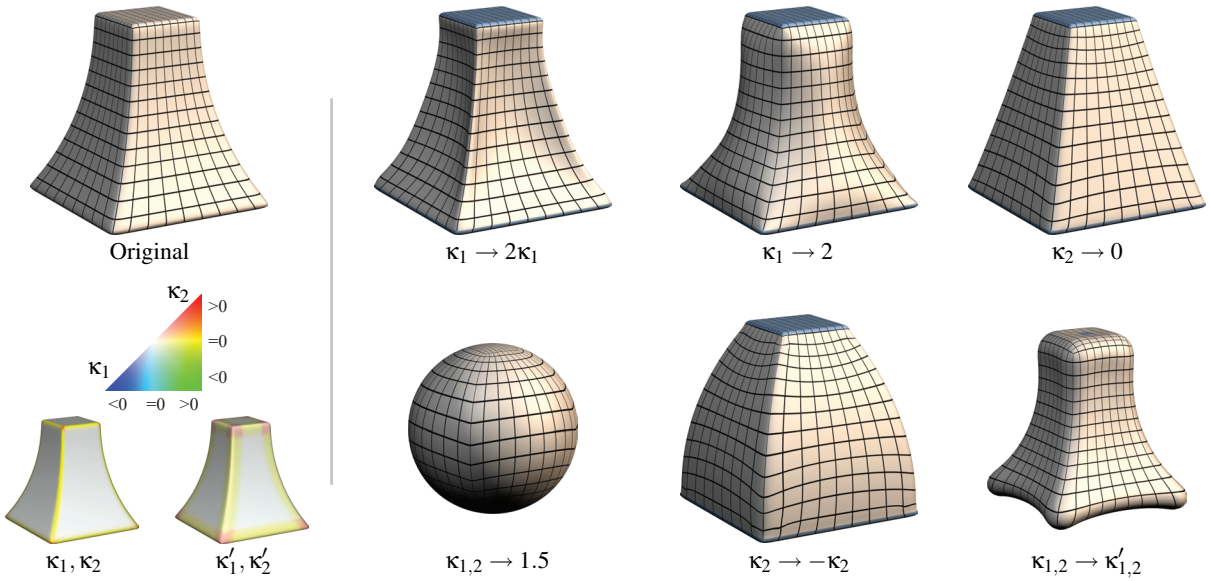

Figure 2: *Geometry processing in the curvature domain. The models on the right (blue regions indicate constrained vertices) have been reconstructed by modifying the signed maximum and minimum curvatures, κ_1 and κ_2, respectively, of the model shown in the upper left. For example, setting the minimum curvature κ_2 to zero while keeping κ_1 fixed straightens out the curved regions, as shown in the top right. In the bottom right, the curvatures of a smaller scale are set to the curvatures estimated at a larger scale, as visualized in the curvature plots in the bottom left.*

shown in Figure 2 and in Section 5, this approach facilitates a variety of geometry processing operations that are difficult to achieve by manipulating spatial 3D coordinates but trivial to formulate in the curvature domain.

The main challenge lies in the formulation and implementation of the inverse mapping, i.e., the reconstruction of the surface geometry from the modified curvature values. In general, this mapping is not well-defined and not every set of curvature values is realizable. For example, there is no 3D embedding of a genus-1 surface with constant principal curvatures everywhere.

We address this issue by formulating the reconstruction as an optimization process that computes a deformation of the input surface that best approximates the prescribed curvatures in a least-squares sense. Additional terms in the objective function aim at preserving metric properties of the original surface and ensure that the solution is well-defined.

Due to the nonlinear relation between spatial coordinates and corresponding curvature values, the optimization employs an iterative Levenberg-Marquardt method to solve the nonlinear least-squares problem. This procedure requires partial derivatives for which we derive analytical expressions. We show various applications of our framework that demonstrate how complex geometry processing operations can be formulated as simple operations on surface curvatures.

Related Work. The most common methods in surface optimization are based on energy minimizing flows, where a given surface is progressively evolved to decrease an energy functional that quantifies the desired surface properties. Taubin [Tau95] proposed an iterative Laplacian scheme to implement surface diffusion for low-pass filtering of discrete surfaces. Desbrun and colleagues [DMSB99] perform mean curvature flow to remove geometric noise on a surface and propose an implicit scheme to stabilize the computation. Ohtake and co-workers [OBS02] apply diffusion to the mesh normals and reconstruct the smoothed surface using a fitting approach. Bobenko and Schröder [BS05] introduced a version of discrete Willmore flow that preserves important symmetries of the continuous setting. A variety of different energy functionals, including Willmore and minimum variation of curvature energies, are studied by Pushkar and Sequin [PS07] in the context of fair surface design. Similarly, Pinkall and Polthier [PP93] construct discrete minimal surfaces based on an area minimizing flow.

Various researchers have extended this class of shape optimization methods by adding more direct control of the flow evolution. Hildebrandt and Polthier [HP04] present a method for feature-preserving noise removal on surface meshes based on an anisotropic mean curvature flow. Their method allows mean curvatures to be prescribed as targets for the flow evolution. Eckstein and co-workers [EPT*07] generalize geometric surface flows by tailoring the inner

product of the underlying vector field to the requirements of specific applications. This extension provides a design tool for controlling the flow, which has been successfully applied for surface fairing and deformable shape matching. Tosun and co-workers propose a system for shape optimization using reflection lines that allows modifying surfaces by specifying a target reflection function gradient [TGRZ07]. An optimization then solves for the surface that best matches the prescribed reflection field. In the context of surface reconstruction from differential quantities, Lipman and colleagues [LSLCO05] introduced a rigid motion invariant surface representation based on discrete forms.

Building on these ideas, we develop a comprehensive system for curvature-domain shape processing. Our work complements existing surface optimization methods by supporting direct editing and filtering of surface curvatures, thus broadening the toolset for manipulating discrete surfaces.

2. Continuous Problem Formulation

Our goal is to develop a geometry processing tool that enables the user to modify a given surface $\mathcal{S} \subset \mathbb{R}^3$ by altering its curvature profile to obtain a new surface $\mathcal{S}'$. This procedure requires the computation of curvatures on the input model $\mathcal{S}$, suitable operators to modify these curvatures, and a method to reconstruct the corresponding surface $\mathcal{S}'$. We formulate the reconstruction as an optimization process that finds the unknown surface $\mathcal{S}'$ by minimizing an appropriate energy function. We first derive this energy in the continuous setting, then propose a suitable discretization in Section 3, and discuss the resulting optimization in Section 4.

Let $\mathcal{S}$ and $\mathcal{S}'$ be defined over a domain $\Omega \subset \mathbb{R}^2$ by mappings $\mathbf{p}, \mathbf{p}' : \Omega \to \mathbb{R}^3$, respectively. We denote with $\kappa_1(u,v)$ the maximum and with $\kappa_2(u,v)$ the minimum curvature at a point $\mathbf{p}(u,v) \in \mathcal{S}$, but omit the (u,v) parameters for notational brevity. Quantities of the unknown surface $\mathcal{S}'$ are defined analogously and denoted by a prime. In principle, we want to be able to prescribe arbitrary target curvature values $\hat{\kappa}_1$ and $\hat{\kappa}_2$ for the modified surface $\mathcal{S}'$ and preserve important properties of the original surface $\mathcal{S}$. We define an energy that quantifies this goal by measuring the deviation of principal curvatures κ_1', κ_2' of the surface $\mathcal{S}'$ from the prescribed target values $\hat{\kappa}_1$, $\hat{\kappa}_2$ as

$$E_c = \int_\Omega (\hat{\kappa}_1 - \kappa_1')^2 + (\hat{\kappa}_2 - \kappa_2')^2 \mathrm{d}A, \tag{1}$$

where $\mathrm{d}A = \sqrt{\det \mathbf{I}}\, \mathrm{d}u\mathrm{d}v$ and $\mathbf{I}$ is the first fundamental form. In addition, we wish $\mathcal{S}'$ to have a similar metric as $\mathcal{S}$ and define the energy

$$E_m = \int_\Omega \left\| \mathbf{I} - \mathbf{I}' \right\|_F^2 \mathrm{d}A, \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The surface $\mathcal{S}'$ can then be found as the minimizer of the combined energy

$$E = k_c E_c + k_m E_m, \tag{3}$$

with constants k_c and k_m that allow a trade-off between curvature fit and metric distortion.

3. Discretization

Minimizing the energy defined in Equation 3 requires a suitable discretization of the surface, its principal curvatures, and the metric term. We approximate a surface $\mathcal{S}$ by a triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V} = \{v_i\}$ denotes the set of vertices, $\mathcal{E} = \{e_{ij}\}$ the edge set, and $\mathcal{F} = \{f_{ijk}\}$ the face set with $1 \le i,j,k \le n = |\mathcal{V}|$. The position of vertex v_i is given by $\mathbf{v}_i \in \mathbb{R}^3$ and the edge vector corresponding to e_{ij} is $\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i$. To simplify notation, we omit the indices when the meaning is clear from the context.

3.1. Discrete Curvature

A variety of techniques have been proposed to estimate curvatures on piece-wise linear surfaces (cf., [MDSB02, CSM03, CP05, GGRZ06, KSNS07]). In our implementation, we use the curvature tensor proposed by Cohen-Steiner and Morvan [CSM03] as it provides a robust and theoretically well-founded estimation of the principle curvatures at multiple scales, which we exploit in our framework to implement multi-scale processing operations. However, our method is not dependent on this specific curvature discretization and other techniques could be used instead.

We adopt the notation used by Alliez and colleagues [ACSD*03] and summarize the curvature computation here. The curvature tensor $\mathcal{K}(v)$ for vertex v is found by averaging an edge-based tensor for all edges that fall within a region B surrounding the vertex. The curvature tensor is represented by the 3×3 matrix

$$\mathcal{K}(v) = \frac{1}{|B|} \sum_{e \in B} \beta(e)\, |e \cap B|\, \bar{\mathbf{e}}\bar{\mathbf{e}}^\top, \tag{4}$$

where $|B|$ is the surface area of the region B, $\beta(e)$ represents the signed angle between the normals of the triangles incident to e and is positive for a convex crease and negative for a concave one, $|e \cap B|$ is the length of the portion of the edge within the region B, and $\bar{\mathbf{e}}$ is a unit-length vector aligned with $\mathbf{e}$. The principle curvatures are found by computing the eigenvalues of the curvature tensor $\mathcal{K}(v)$. The eigenvalue closest to zero is discarded, and the remaining two form the signed maximum and minimum curvatures: the larger signed eigenvalue is κ_1 and the smaller κ_2. The

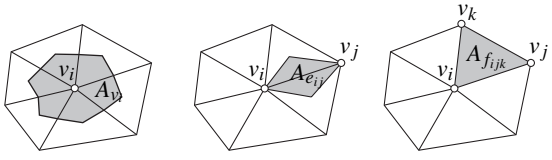

Figure 3: *Vertex, edge, and face areas.*

size of B determines the scale of the curvature estimation. As the smallest scale we use the barycentric area A_{v_i} of a vertex v_i [MDSB02] (Figure 3). We use barycentric areas rather than generalized Voronoi regions to simplify derivative computations. For larger scales, we compute the union of the barycentric areas of all vertices within a certain distance to v_i.

3.2. Discrete Energy

We discretize the integral of Equation 1 as a sum of discrete curvatures, leading to

$$E_c = \sum_{v_i \in \mathcal{V}} A_{v_i} \left[(\hat{\kappa}_{1,i} - \kappa'_{1,i})^2 + (\hat{\kappa}_{2,i} - \kappa'_{2,i})^2 \right], \qquad (5)$$

where the subscript i indicates that the corresponding quantities are computed with respect to vertex v_i.

The metric term E_m measures the deviation from isometry, i.e. change of lengths. A discrete version can thus be formulated as

$$E_m = \sum_{e \in \mathcal{E}} A_e (1 - \frac{\|\mathbf{e}'\|}{\|\mathbf{e}\|})^2, \qquad (6)$$

where A_e is the barycentric area associated with edge e (Figure 3). We found that trying to preserve isometry can be too restrictive in certain applications, in particular when large changes in shape are desired. We thus propose a conformal energy E_α that only penalizes deviation of inner angles:

$$E_\alpha = \sum_{f_{ijk} \in \mathcal{F}} A_{f_{ijk}} \left[(\alpha_i - \alpha'_i)^2 + (\alpha_j - \alpha'_j)^2 + (\alpha_k - \alpha'_k)^2 \right], \qquad (7)$$

where $A_{f_{ijk}}$ denotes the area of triangle f_{ijk}, and α_i, α_j, α_k are the inner angles of f_{ijk} at vertices v_i, v_j, and v_k, respectively (Figure 3). Note that similar energies are also used in surface parameterization [SdS01].

The isometric energy E_m aims at preserving lengths, i.e. angles and area, whereas the conformal term E_α tries to preserve inner angles only, i.e. does not place a penalty on uniform scaling. Figure 4 illustrates the difference between the two terms.

For the results in this paper, we use the conformal formulation E_α and focus on this energy for the remainder of the discussion. Additionally, we add a third term to the optimization that measures the size of the deformation field:

$$E_d = \sum_{v_i \in \mathcal{V}} \|\mathbf{v}_i - \mathbf{v}'_i\|^2. \qquad (8)$$

This term has a twofold purpose. First, it resolves ambiguity in the objective function that exists because E_c, E_m, and E_α are invariant under rigid transformations. Second, it addresses the problem of overfitting by penalizing large deformations that might otherwise be selected by the optimizer

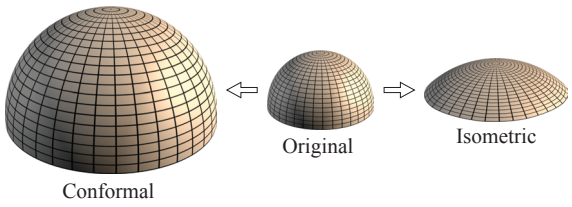

Figure 4: *Scaling curvatures by a factor of 0.5 for a hemisphere (center). The conformal energy (left) only preserves inner angles and achieves the target curvature by scaling the model. The isometric energy (right) additionally tries to preserve area at the cost of a slight deviation in inner angles.*

in an attempt to drive the total energy to its lowest possible value. Thus, the energy we use is defined by

$$E = \frac{1}{2}(k_c E_c + k_\alpha E_\alpha + k_d E_d), \qquad (9)$$

where the parameters k_c, k_α, and k_d allow the user to adapt the optimization to a particular application domain. The factor $\frac{1}{2}$ is introduced for convenience in subsequent equations.

4. Optimization

We minimize the energy defined in Equation 9 in order to find the vertex positions $\mathbf{v}'_1 \dots \mathbf{v}'_n$ of the deformed shape that best matches the target curvatures:

$$\mathbf{v}'_1 \dots \mathbf{v}'_n = \underset{\mathbf{v}'_1 \dots \mathbf{v}'_n}{\operatorname{argmin}} E. \qquad (10)$$

This optimization is a nonlinear least-squares problem, due to the nonlinear relationship between the terms of the objective function (principle curvature values, triangle angles) and the vertex positions. Specifically, it falls into the category of composite non-smooth optimization [WF86] due to discontinuities in the derivatives of the principle curvatures at umbilic points. As detailed in the Appendix, we are able to derive suitable approximations for the derivatives at these points so that efficient methods designed for smooth functions can be applied. Although we have written the problem as an unconstrained optimization, positional vertex constraints can be trivially enforced by treating the constrained vertices as constants rather than free variables.

We employ the Levenberg-Marquardt algorithm [MNT04], which is a damped version of the Gauss-Newton method that first linearizes the nonlinear problem with Taylor expansion around $\mathbf{x}$:

$$\mathbf{f}(\mathbf{x} + \delta) = \mathbf{f}(\mathbf{x}) + \mathbf{J}\delta \qquad (11)$$

The vector $\mathbf{f}(\mathbf{x})$ stacks the equations that define the objective function so that $\mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) = 2E$, the vector $\mathbf{x}$ stacks the unknown vertex positions, and $\mathbf{J}$ is the Jacobian matrix of $\mathbf{f}(\mathbf{x})$. The details of the derivative computations to build $\mathbf{J}$ are given in the Appendix.

Each iteration solves a linearized problem to improve $\mathbf{x}_k$, the current estimate of the unknown vertex positions:

$$\begin{aligned} \delta_k &= \underset{\delta}{\operatorname{argmin}} \quad \|\mathbf{f}(\mathbf{x}_k) + \mathbf{J}\delta\|_2^2 \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \delta_k. \end{aligned} \tag{12}$$

In solving this problem, the Levenberg-Marquardt method adds a damping factor to the normal equations, yielding:

$$(\mathbf{J}^\top \mathbf{J} + \mu \mathbf{I})\delta = \mathbf{J}^\top \mathbf{f}(\mathbf{x}_k), \tag{13}$$

with $\mu > 0$. The initial value of μ is chosen to be 10^{-6} times the largest entry on the diagonal of $\mathbf{J}^\top \mathbf{J}$ evaluated at $\mathbf{x}_0$. In subsequent iterations, μ is updated according to the schedule suggested by Nielsen [Nie99]. The algorithm is not sensitive to the initial value of μ as it is continually adapted by the update procedure.

The damping factor serves two main purposes. First, it improves numeric robustness by guaranteeing that the coefficient matrix is positive definite and that δ_k is in the decent direction. Second, the damping parameter influences both the step direction and the step size, obviating the need for a specific line search and leading to faster convergence [MNT04].

Since the system matrix in Equation 13 is sparse, we solve the normal equations in each iteration using a direct solver that employs sparse Cholesky factorization [SG04]. We detect convergence by monitoring the change in the objective function $F_k = E(\mathbf{x}_k)$, the gradient of the objective function, and the magnitude of the update vector δ_k [GMW89]:

$$\begin{aligned} |F_k - F_{k-1}| &< \varepsilon(1 + F_k) \\ \|\nabla F_k\|_\infty &< \sqrt[3]{\varepsilon}(1 + F_k) \\ \|\delta_k\|_\infty &< \sqrt[2]{\varepsilon}(1 + \|\delta_k\|_\infty). \end{aligned} \tag{14}$$

In our experiments, the optimization converges after about five iterations with $\varepsilon = 10^{-6}$.

For some applications, there may be a conflict between the desired curvatures and the original shape metric, since the requested curvatures necessitate a deviation in triangle shape. For this reason, we adapt the optimization procedure to allow the metric to evolve in an iterative fashion. Initially, the target angles and vertex positions for E_α and E_d are derived from the original shape and the optimization is allowed to converge. Then, the angles and positions are updated with values taken from the converged shape and the process is repeated. Conceptually, this iterative scheme evolves the metric to better match the curvatures, while avoiding drastic changes in triangle shape that may cause numeric instabilities.

5. Results

This section shows various geometry processing applications supported by our system. Following the pipeline depicted in Figure 1, we first compute discrete principal curvatures on the input mesh as described in Section 3.1. These curvature values are then modified by applying filtering operations or direct edits, and the resulting surface is reconstructed using the optimization introduced in Section 4.

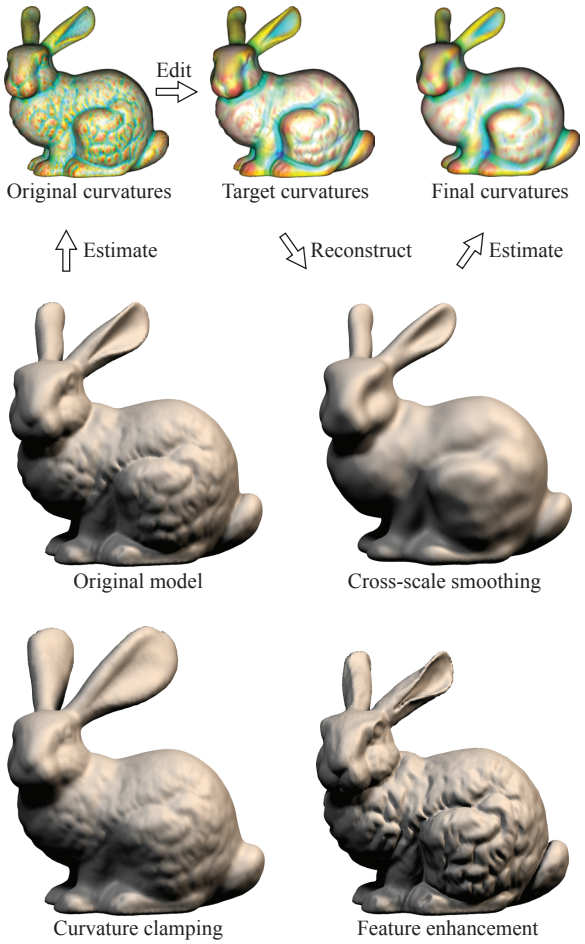

Figure 5: *Curvature-domain processing on the Stanford bunny. The top row shows the curvature plots of the original model, the prescribed target values, and the actual achieved curvatures on the final reconstructed model. Vertices on the base of the bunny are constrained to remain fixed in place in the optimization.*

Figure 5 shows a number of curvature-domain edits on a shape with a complex curvature distribution. For cross-scale smoothing, we set the target principal curvatures of the smallest scale (barycentric vertex area) to the values computed at a scale of four times the average one-ring radius. As the curvature plots indicate, the target curvature values are well approximated in the final model. The curvature plots visualize both principle curvatures according to the colormap in Figure 2. Note that the resulting smoothing avoids local shrinkage artifacts often observed with diffusion-based ap-

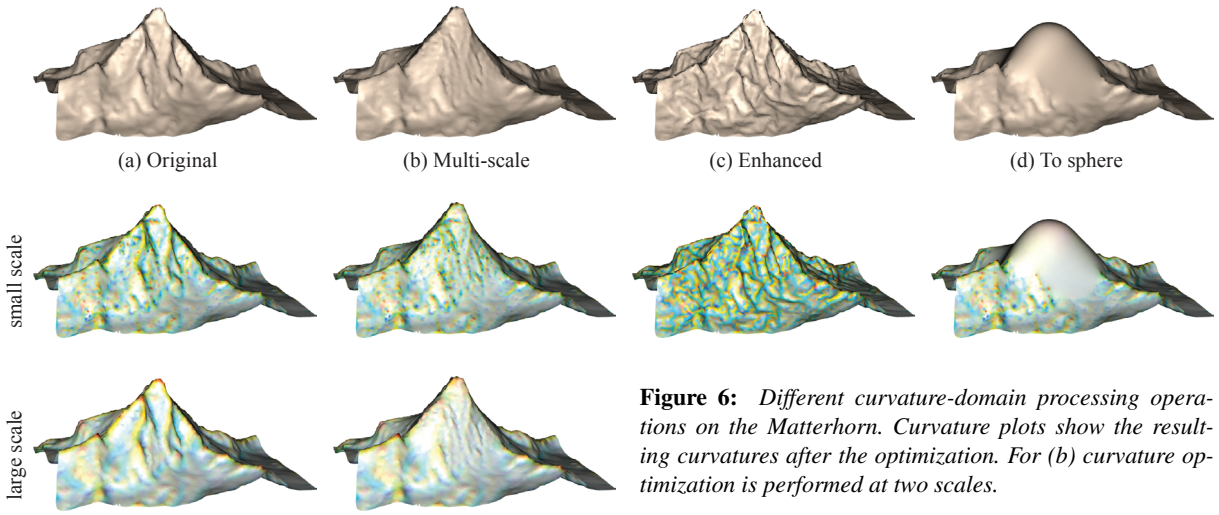

Figure 6: *Different curvature-domain processing operations on the Matterhorn. Curvature plots show the resulting curvatures after the optimization. For (b) curvature optimization is performed at two scales.*

proaches that can increase mean curvature and eventually lead to pinch-off singularities.

Curvature clamping restricts both the minimum and maximum curvature to lie within a user-defined interval, thus removing the extreme curvatures from the surface. As a result, the bunny's ears inflate in order to avoid high curvatures. Note how surface detail that is characterized by curvatures within the clamping interval is preserved, while strong creases are smoothed and rounded.

Figure 7 shows one-sided curvature clamping on a machine part, where curvatures are restricted to lie in the interval $[-5,\infty]$. As a result, concave corners evolve into smooth fillets to meet the new curvature requirements. Note that normal discontinuities across feature edges are introduced purely for rendering purposes. Curvature processing is oblivious to these tagged edges, i.e. the entire mesh is considered a discrete approximation of a smooth surface.

Feature enhancement in Figure 5 is achieved by increasing the largest absolute principal curvature by an amount proportional to the difference of the absolute principal curvature values. This change enhances convex ridges as well as concave valleys. The same enhancement filter is applied in Figure 6 (c) on a digital elevation model. This figure also shows the effect of multi-scale editing (b), where target curvatures have been specified on two different scales: the curvatures of the original model are prescribed on the smallest scale, while on a coarser scale target curvatures are set to a constant in a circular region around the peak of the mountain. This "flattens out" the mountain flanks, while preserving the fine-scale structure. Multi-scale editing is easily incorporated into the optimization by including target curvatures at different scales in the energy E_c. Image (d) shows the result of a single-scale edit, where both principal curvatures for the center region have been set to the same constant, thus pushing the shape towards a spherical configuration.

Figure 8 shows a curvature-domain bilateral filter applied to a noisy range scan. Target curvature values $\hat{\kappa}_{1,i}$ for vertex v_i (and analogously for $\hat{\kappa}_{2,i}$) are computed as local weighted averages that combine domain and range filtering

$$\hat{\kappa}_{1,i} = \frac{\sum_{v_j \in N_i} \phi_c(\|\mathbf{v}_i - \mathbf{v}_j\|)\phi_s(|\kappa_{1,i} - \kappa_{1,j}|)\kappa_{1,i}}{\sum_{v_j \in N_i} \phi_c(\|\mathbf{v}_i - \mathbf{v}_j\|)\phi_s(|\kappa_{1,i} - \kappa_{1,j}|)}, \qquad (15)$$

where N_i is the local averaging region around vertex v_i, and ϕ_c and ϕ_s are two Gaussians measuring spatial closeness and curvature similarity, respectively (see [TM98] for details). Compared to isotropic Laplacian smoothing, the bilateral filter better preserves ridges and corners, while leading to an overall smoother curvature distribution. The curvature plots reveal the anisotropic behavior of the filter, i.e., illustrate

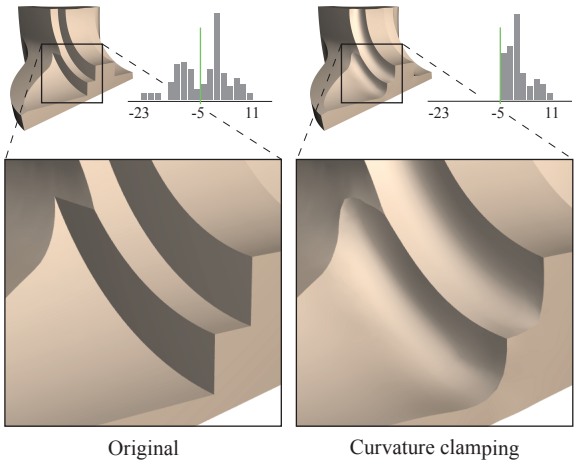

Figure 7: *Curvature clamping of a machine part. The histograms show the distribution of κ_2 on a logarithmic scale before and after the optimization.*

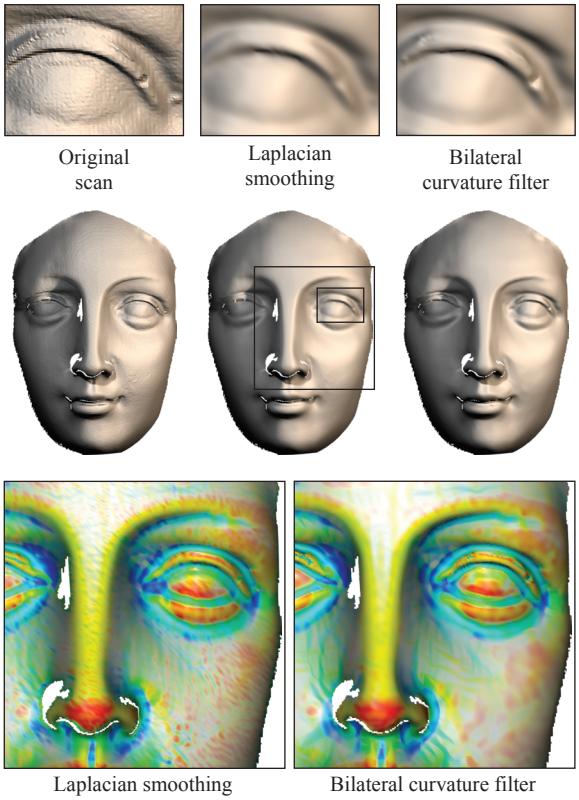

Figure 8: *Comparison of isotropic Laplacian smoothing with bilateral filtering of curvatures on a noisy range scan.*

how curvature variation is reduced along feature lines without blurring the surface across the features. For comparison we apply ten smoothing steps in both examples. Boundaries are handled without special treatment. Our formulation operates directly on discrete curvatures, i.e., scalar attributes defined on each mesh vertex, so that we avoid local height-field parameterizations of vertex positions, as for example used in [FDCO03], that can lead to distortions for larger neighborhoods or high curvature regions.

To evaluate the results of our algorithm we introduce a score function σ that measures the degree to which the prescribed curvatures are achieved by the optimization procedure. This function is designed to be a relative measure independent of both scale and sampling and is defined as

$$\sigma = 1 - \frac{\sum_{v_i \in \mathcal{V}} A_{v_i} \left[(\hat{\kappa}_{1,i} - \kappa'_{1,i})^2 + (\hat{\kappa}_{2,i} - \kappa'_{2,i})^2 \right]}{\sum_{v_i \in \mathcal{V}} A_{v_i} \left[(\hat{\kappa}_{1,i} - \kappa_{1,i})^2 + (\hat{\kappa}_{2,i} - \kappa_{2,i})^2 \right]}, \quad (16)$$

where $\kappa_{j,i}$ denotes the principal curvatures of the original model.

Higher values indicate that the optimization procedure was better able to match the desired curvatures. Table 1 lists the values of σ for the models shown in Figures 2 and 5. The differences in the achieved scores stem from the fact that the metric regularization prevents drastic changes of the shape and that target curvatures can be incompatible.

Fig. 2	σ	Bunny	σ
$\kappa_1 \rightarrow 2\kappa_1$	.950	Clamp	.821
$\kappa_1 \rightarrow 2$	.944	Cross-Scale	.963
$\kappa_2 \rightarrow 0$	.858	Enhance	.721
$\kappa_{1,2} \rightarrow 1.5$	.999		
$\kappa_2 \rightarrow -\kappa_2$	.988		
$\kappa_{1,2} \rightarrow \kappa'_{1,2}$	.994		

Table 1: *The value $\sigma \in [0,1]$ indicates to what extent the target curvatures are achieved by the final model.*

6. Discussion

The results shown in the previous section demonstrate the flexibility of curvature-domain shape processing. Although specialized techniques based only on mean curvature exist for some of the applications presented, our approach offers a different perspective since the specific geometric operations employed by our method are often quite different than the ones already known. Investigating these similarities and differences for specific applications may lead to new insights about shape processing and suggest avenues of future work in this area.

An interesting connection of our approach to continuum mechanics becomes apparent when comparing the energy function of Equation 3 with the elastic thin-shell energy

$$E = \int_\Omega k_s \left\| \mathbf{I} - \mathbf{I}' \right\|_F^2 + k_b \left\| \mathbf{II} - \mathbf{II}' \right\|_F^2 \mathrm{d}u\mathrm{d}v, \quad (17)$$

where $\mathbf{II}$ denotes the second fundamental form, and k_s and k_b are stiffness parameters that determine the stretching and bending resistance of the material [TPBF87]. In our formulation we replace the bending term by principal curvature differences, thus avoiding the directional component of the second fundamental form. The key difference, however, is that in Equation 17 the second fundamental form $\mathbf{II}$ is computed from a given, initial rest state, whereas we prescribe arbitrary target curvatures values. Effectively, minimizing the objective function 3 pulls the surface to a "fictitious" rest state defined by the target curvatures.

Rusinkiewicz observed [Rus04] that the curvature tensor proposed by Cohen-Steiner and Morvan can yield a poor approximation of the true curvatures for small local neighborhoods at low-valence vertices. In our optimization this can lead to local distortions that diminish the quality of the final result. In general, we observed that the performance degrades with poor input mesh quality, since the estimation of curvatures and corresponding derivatives becomes numerically unstable. Our experiments indicate that these artifacts

can be reduced by applying appropriate local remeshing operations, e.g., [SG03], to improve the stability of the curvature estimation.

In terms of performance, the dominating factor is the nonlinear optimization used to reconstruct the final model from the given target curvature values. The initial curvature estimation constitutes less then 5 percent of the total computation time, and the overhead for filtering/editing of curvatures is negligible. While performance was not our main concern, the optimization for e.g. the bunny model with 35,111 vertices, i.e. more than 100k unknowns, runs in less than 2.5 minutes on a 2.4 GHz Intel Core 2 Duo with 2GB of memory. Our current prototype implementation supports moderately sized meshes up to 60,000 vertices. Since memory constraints restrict the use of larger meshes with our in-core solver, we intend to improve the scalability of our system by investigating alternative solvers, in particular multi-level solvers that operate on a hierarchy of surface approximations of different levels of detail.

The example of Figure 8 demonstrates how a standard image processing filter can be applied in curvature-domain to implement an advanced geometry processing tool. We believe that our curvature-based processing metaphor offers a versatile framework for exploring a variety of such filtering methods. In the future we intend to follow this avenue and evaluate different filters for their potential use in curvature-based geometry processing.

7. Conclusion

We introduced curvature-domain shape processing, an optimization framework that allows arbitrary curvature values to be prescribed on a given surface. We show how advanced geometry processing techniques can be implemented as simple operations in the curvature domain. Through the use of nonlinear optimization, we push the complexity of geometry processing tasks toward computation, thus reducing the burden on the user, who can apply simple and intuitive modifications and filtering operations in the curvature domain to edit and design geometric models. Our framework offers a new perspective on shape optimization and provides a platform for further development in 3D geometry processing.

Acknowledgments

We would like to thank Pierre Alliez, Mario Botsch, David Cohen-Steiner, Joachim Giesen, Eitan Grinspun, Martin Kilian, Niloy Mitra, Helmut Pottmann, Filip Sadlo and Johannes Wallner for fruitful discussions about this research.

Appendix

Building the Jacobian matrix for the optimization problem outlined in Section 4 requires the computation of partial derivatives of the per-vertex principal curvatures with respect to the unknown vertex positions. We present the derivative formulas here. In the following equations, $\mathbf{v}_{i,c}$ refers to the component c of vertex i, $c \in \{x, y, z\}$.

Curvature Derivatives

The principle curvatures κ_1 and κ_2 are eigenvalues of the curvature tensor $\mathcal{K}$ defined in Equation 4. Umbilic points occur whenever the curvature tensor field is isotropic, indicating that the underlying surface locally approximates a sphere or plane. At such points, the principal curvatures are equal and their derivatives are undefined. However, this problem falls into the class of composite nonsmooth optimization [WF86] since, although the derivatives are discontinuous at umbilic points, they can be approximated from information available at the discontinuities. Following Kim and colleagues [KCH02], we average the derivative computation at nonsmooth points to achieve a viable approximation. With these observations in place, we compute the curvature derivatives according to three cases.

Case 1: The most common case occurs when all three eigenvalues are distinct. In this situation, we have

$$\frac{\partial}{\partial \mathbf{v}_{i,c}} \lambda_j = \mathbf{u}_j^\top \left(\frac{\partial}{\partial \mathbf{v}_{i,c}} \mathcal{K} \right) \mathbf{u}_j, \tag{18}$$

where λ_j and $\mathbf{u}_j$ are the j^{th} eigenvalue and normalized eigenvector, respectively, of the curvature tensor $\mathcal{K}(v_i)$ and $\lambda_1 < \lambda_2 < \lambda_3$ [HUY95]. The tensor derivative is given below in Equation 21.

Case 2: A cylindric point occurs when the two smallest eigenvalues, in an absolute sense, are nearly equal and distinct from the remaining eigenvalue. The derivative of this remaining eigenvalue, which corresponds to either κ_1 or κ_2, can be computed as described above. The derivative of the other principle curvature can be computed using the mean curvature $H = \frac{1}{2}(\kappa_1 + \kappa_2)$. Since $H = \frac{1}{2}\text{trace}(\mathcal{K})$, we have

$$\frac{\partial}{\partial \mathbf{v}_{i,c}} H = \frac{1}{2} \left(\frac{\partial}{\partial \mathbf{v}_{i,c}} t_{11} + \frac{\partial}{\partial \mathbf{v}_{i,c}} t_{22} + \frac{\partial}{\partial \mathbf{v}_{i,c}} t_{33} \right), \tag{19}$$

where t_{jj}, $j \in 1 \ldots 3$, are the diagonal entries of $\mathcal{K}$. The formula for their partial derivatives is presented in Equation 21.

Case 3: When $\kappa_1 = \kappa_2$, we have an umbilic point and the derivative is not defined. However, a viable approximation to the derivative at this point is given by averaging the derivatives that meet at the discontinuity [KCH02], which leads to

$$\frac{\partial}{\partial \mathbf{v}_{i,c}} \kappa_1 = \frac{\partial}{\partial \mathbf{v}_{i,c}} \kappa_2 \approx \frac{\partial}{\partial \mathbf{v}_{i,c}} H, \tag{20}$$

where the rightmost term is defined in Equation 19.

As a side remark, the principle curvatures and their derivatives can also be estimated in terms of the mean and Gaussian curvatures. Under the assumption that the third eigenvalue is zero, this approach avoids eigenvalue decomposition and leads to simpler derivative expressions. However, we found this method to be less accurate near umbilic points.

Tensor Derivatives

Equation 18 requires the partial derivative of the curvature tensor $\mathcal{K}$ with respect to the unknown vertices:

$$\frac{\partial}{\partial \mathbf{v}_{i,c}}\mathcal{K} = \frac{1}{|B|}\frac{\partial}{\partial \mathbf{v}_{i,c}}\hat{\mathcal{K}} - \frac{\hat{\mathcal{K}}}{|B|^2}\frac{\partial}{\partial \mathbf{v}_{i,c}}|B| \qquad \text{where } \hat{\mathcal{K}} = \sum_{e\in B} \beta(e)\,|e\cap B|\,\bar{\mathbf{e}}\bar{\mathbf{e}}^\top. \tag{21}$$

The derivative of $\hat{\mathcal{K}}$ is given by

$$\frac{\partial}{\partial \mathbf{v}_{i,c}}\hat{\mathcal{K}} = \sum_{e\in B}\Big[\Big(\frac{\partial}{\partial \mathbf{v}_{i,c}}\beta(e)\Big)|e\cap B|\bar{\mathbf{e}}\bar{\mathbf{e}}^\top + \beta(e)\Big(\frac{\partial}{\partial \mathbf{v}_{i,c}}|e\cap B|\Big)\bar{\mathbf{e}}\bar{\mathbf{e}}^\top + \beta(e)|e\cap B|\Big(\frac{\partial}{\partial \mathbf{v}_{i,c}}\bar{\mathbf{e}}\bar{\mathbf{e}}^\top\Big)\Big] \tag{22}$$

where

$$\frac{\partial}{\partial \mathbf{v}_{i,c}}\bar{\mathbf{e}}\bar{\mathbf{e}}^\top = \frac{\partial}{\partial \mathbf{v}_{i,c}}\frac{\mathbf{e}\mathbf{e}^\top}{\|\mathbf{e}\|^2} = \Big(\frac{\partial}{\partial \mathbf{v}_{i,c}}\mathbf{e}\mathbf{e}^\top\Big)\frac{1}{\|\mathbf{e}\|^2} - \frac{\bar{\mathbf{e}}\bar{\mathbf{e}}^\top}{\|\mathbf{e}\|^2}\frac{\partial}{\partial \mathbf{v}_{i,c}}\|\mathbf{e}\|^2. \tag{23}$$

The outer product derivative is given by

$$\frac{\partial}{\partial \mathbf{v}_{i,c}}\mathbf{e}\mathbf{e}^\top = \frac{\partial}{\partial \mathbf{v}_{i,c}}\begin{bmatrix}\mathbf{e}_x\mathbf{e}_x & \mathbf{e}_x\mathbf{e}_y & \mathbf{e}_x\mathbf{e}_z\\ \mathbf{e}_y\mathbf{e}_x & \mathbf{e}_y\mathbf{e}_y & \mathbf{e}_y\mathbf{e}_z\\ \mathbf{e}_z\mathbf{e}_x & \mathbf{e}_z\mathbf{e}_y & \mathbf{e}_z\mathbf{e}_z\end{bmatrix}, \tag{24}$$

which is straightforward to evaluate entrywise.

The remaining terms in Equations 21, 22, and 23 are more easily interpreted geometrically as derivatives with respect to vertex $\mathbf{v}_i$, rather than with respect to its individual components.

The area gradient for Equation 21 is given by

$$\frac{\partial}{\partial \mathbf{v}_i}|B| = \sum_{f\in B} B_f \frac{\partial}{\partial \mathbf{v}_i}A_f \tag{25}$$

where A_f is the face area and B_f is the fraction of A_f which is inside B. Since B is a union of barycentric regions, B_f is always equal to $\frac{1}{3}$, $\frac{2}{3}$, or 1. The derivative of the face area is

$$\frac{\partial}{\partial \mathbf{v}_1}A_{f_{123}} = \frac{1}{2}(\bar{\mathbf{n}}\times\mathbf{e}_{23}), \tag{26}$$

where $\bar{\mathbf{n}}$ is the normalized face normal and $\mathbf{e}_{23}$ is the edge vector opposite v_1 (Figure 9).

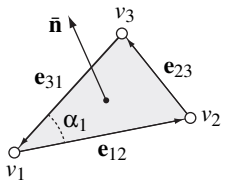

Figure 9: *Notation used in the derivation of derivatives.*

The dihedral angle gradients from Equation 22 can be computed according to the formulas presented by Bridson, Marino, and Fedkiw [BMF03].

The edge length derivatives from Equations 22 and 23 are

$$\frac{\partial}{\partial \mathbf{v}_1}|e_{12}\cap B| = -B_{e_{12}}\bar{\mathbf{e}}_{12} \qquad \frac{\partial}{\partial \mathbf{v}_2}|e_{12}\cap B| = B_{e_{12}}\bar{\mathbf{e}}_{12}$$
$$\frac{\partial}{\partial \mathbf{v}_1}\|\mathbf{e}_{12}\|^2 = -2\mathbf{e}_{12} \qquad \frac{\partial}{\partial \mathbf{v}_2}\|\mathbf{e}_{12}\|^2 = 2\mathbf{e}_{12}, \tag{27}$$

where $B_{e_{12}}$ is the fraction of the edge which is inside B. Again because of the barycentric regions, $B_{e_{12}}$ is either $\frac{1}{2}$ or 1.

Metric Derivatives

As discussed in Section 3.2, we have experimented with two forms of metric regularization: isometric and conformal. The isometric regularization measures the change in edge length, and the appropriate derivatives are already given in Equation 27. The conformal regularization measures the change in each triangle's inner angles. The necessary derivatives are

$$\frac{\partial}{\partial \mathbf{v}_2}\alpha_1 = \frac{\mathbf{e}_{12}\times\bar{\mathbf{n}}}{\|\mathbf{e}_{12}\|^2} \tag{28}$$

$$\frac{\partial}{\partial \mathbf{v}_3}\alpha_1 = \frac{\mathbf{e}_{31}\times\bar{\mathbf{n}}}{\|\mathbf{e}_{31}\|^2} \tag{29}$$

$$\frac{\partial}{\partial \mathbf{v}_1}\alpha_1 = -\frac{\mathbf{e}_{12}\times\bar{\mathbf{n}}}{\|\mathbf{e}_{12}\|^2} - \frac{\mathbf{e}_{31}\times\bar{\mathbf{n}}}{\|\mathbf{e}_{31}\|^2} \tag{30}$$

for face f_{123} and the inner angle α_1 at v_1 (Figure 9).

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Transactions on Graphics 22*, 3 (July 2003), 485–493.

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANE G., SPAGNUOLO M., TAL A.: Mesh segmentation – a comparative study. In *IEEE International Conference on Shape Modeling and Applications* (2006).

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Symposium on Computer Animation* (2003), pp. 28–36.

[BPK*07] BOTSCH M., PAULY M., KOBBELT L., ALLIEZ P., LEVY B.: Geometric modeling based on polygonal meshes. *ACM SIGGRAPH Course Notes* (2007).

[BS05] BOBENKO A. I., SCHRÖDER P.: Discrete willmore flow. In *Symposium on Geometry Processing* (2005).

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aided Geom. Des. 22*, 2 (2005), 121–146.

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *Symposium on Computational geometry* (2003), pp. 312–321.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH* (1999), pp. 317–324.

[EPT*07] ECKSTEIN I., PONS J.-P., TONG Y., KUO C.-C. J., DESBRUN M.: Generalized surface flows for mesh processing. In *Symposium on Geometry Processing* (2007), pp. 183–192.

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003), 950–953.

[GGRZ06] GRINSPUN E., GINGOLD Y., REISMAN J., ZORIN D.: Computing discrete shape operators on general meshes. *Computer Graphics Forum 25*, 3 (2006).

[GMW89] GILL P. E., MURRAY W., WRIGHT M. H.: *Practical Optimization*. Academic Press, London, 1989.

[HP04] HILDEBRANDT K., POLTHIER K.: Anisotropic filtering of non-linear surface features. *Computer Graphics Forum 23*, 3 (2004).

[HUY95] HIRIART-URRUTY J.-B., YE D.: Sensitivity analysis of all eigenvalues of a symmetric matrix. *Numer. Math. 70*, 1 (1995), 45–72.

[KCH02] KIM M. S., CHOI D. H., HWANG Y.: Composite nonsmooth optimization using approximate generalized gradient vectors. *J. Optim. Theory Appl. 112*, 1 (2002), 145–165.

[KSNS07] KALOGERAKIS E., SIMARI P., NOWROUZEZAHRAI D., SINGH K.: Robust statistical estimation of curvature on discretized surfaces. In *Symposium on Geometry Processing* (2007), pp. 13–22.

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005* (2005), ACM Press, pp. 479–487.

[MDSB02] MEYER M., DESBRUN M., SCHROEDER P., BARR A.: Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath.* (2002).

[MHIL02] MA K.-L., HERTZMANN A., INTERRANTE V., LUM E. B.: Recent advances in non-photorealistic rendering for art and visualization. *ACM SIGGRAPH Course Notes* (2002).

[MNT04] MADSEN K., NIELSEN H., TINGLEFF O.: *Methods for Non-Linear Least Squares Problems*. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[Nie99] NIELSEN H.: *Damping Parameter in Marquardt's Method*. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark, 1999.

[OBS02] OTHAKE Y., BELYAEV A., SEIDEL H.-P.: Mesh smoothing by adaptive and anisotropic gaussian filter. *Vision, Modeling, and Visualization* (2002).

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993).

[PS07] PUSHKAR J., SEQUIN C.: Energy minimizers for curvature-based surface functionals. *Computer-Aided Design and Applications* (2007).

[Rus04] RUSINKIEWICZ S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission* (2004).

[SdS01] SHEFFER A., DE STURLER E.: Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers 17* (2001), 326–337.

[SG03] SURAZHSJY V., GOTSMAN C.: Explicit surface remeshing. In *Symposium on Geometry Processing* (2003), pp. 20–30.

[SG04] SCHENK O., GÄRTNER K.: Solving unsymmetric sparse systems of linear equations with pardiso. *Future Gener. Comput. Syst. 20*, 3 (2004), 475–487.

[Sha06] SHAMIR A.: Segmentation and shape extraction of 3d boundary meshes. *Eurographics State-of-the-Art Report* (2006).

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH* (1995), pp. 351–358.

[TGRZ07] TOSUN E., GINGOLD Y. I., REISMAN J., ZORIN D.: Shape optimization using reflection lines. In *Symposium on Geometry Processing* (2007).

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of ICCV* (1998), p. 839.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of SIGGRAPH* (1987), pp. 205–214.

[WF86] WOMERSLEY R. S., FLETCHER R.: An algorithm for composite nonsmooth optimization problems. *J. Optim. Theory Appl. 48*, 3 (1986), 493–523.