Approximation and Visualization of Discrete Curvature on Triangulated Surfaces

Christian Rössl, Leif Kobbelt

Max-Planck-Institut für Informatik, Computer Graphics Group
Im Stadtwald, 66123 Saarbrücken, Germany
Email: roessl@mpi-sb.mpg.de

Abstract

Triangle meshes are a facile and effective representation for many kinds of surfaces. In order to rate the quality of a surface, the calculation of geometric curvatures as there are defined for smooth surfaces is useful and necessary for a variety of applications. We investigate an approach to locally approximate the first and second fundamental forms at every (inner) vertex of a triangle mesh. We use locally isometric divided difference operators, where we compare two variants of parameterizations (tangent plane and exponential map) by testing on elementary analytic surfaces.

We further describe a technique for visualizing the resulting curvature data. A simple median filter is used to effectively filter noise from the input data. According to application dependent requirements a global or a per-vertex local color coding can be provided. The user may interactively modify the color transfer function, enabling him or her to visually evaluate the quality of triangulated surfaces.

1 Introduction

Triangle meshes are becoming a more and more popular representation for surfaces of arbitrary shape and topology. For many applications the approximation of geometric entities such as normals and principle curvatures is very useful. Typical applications include (visual) evaluation of the surface quality, fairing [5, 9] and issues in reverse engineering, like surface segmentation and reconstruction [6].

A triangle mesh is a piecewise linear rather than a smooth surface, so it is not clear how to calculate any derivatives on such a mesh. There are several approaches: In numerical analysis divided differences operators [8] are a common way of estimating discrete derivatives. They often have no immediate geometric meaning since they are defined for a global parameterization. Considering the concepts from differential geometry of smooth surfaces on the other hand directly leads to a variety of methods where geometric primitives are locally fitted to the triangle mesh. The differential parameters can then be obtained from differenciating those well known primitives. In [4] e.g. circular arcs are fitted to the edges of a triangle mesh, approximating normal curvature and eventually principle curvatures. Instead of using curves, one can locally fit an analytic surface to a vertex and its neighbors. Often second order surfaces (quadrics) are used, in [7] e.g. a paraboloid. The polynomal coefficients of this paraboloid are obtained by solving a linear system.

In the first part of this paper we present a method for calculating discrete curvature on triangle meshes. Our approach uses locally isometric divided difference operators which are derived by fitting a second order Taylor polynomal to a vertex and its neighbors. Therefore it is some kind of compromise between the techniques illustrated above. A locally isometric parameterization allows us to utilize such a linear operator since derivatives

with respect to such parameterization do have a geometric interpretation. Two different parameterizations are tested on simple analytic surfaces.

The second part shows a technique of how to visualize the calculated data e.g. the discrete curvature on a triangle mesh. According to the intention of the user a global or a per-vertex local color table is appropriate for color coding. Applications that require absolute measuring of curvature on the mesh e.g. for identifying regions with a given curvature use a global table. If one is mainly interested in local changes of curvature, a local color table should be used. The user may interactively choose and adapt such a color table. In addition, noise is reduced from the input data.

2 Approximation of fundamental forms

We locally estimate the first and second fundamental form of the surface $F(u, v)$ in every vertex of its triangulation. Deriving surface curvatures like Gaussian or mean curvature from the fundamental forms is straightforward. An introduction to the basic concepts of differential geometry can be found e.g. in [3].

In this section V denotes the vertex for which the fundamental forms are to be approximated, $V_i (1 \leq i \leq n$, and for convenience $V_{n+1} := V_1)$ are its neighbors. Q and Q_i denote the positions of V and V_i in 3D space. Without loss of generality the origin is shifted such that $Q := (0, 0, 0)$. For now, we do not handle vertices on the boundary of the mesh.

We want to estimate geometric curvature. The linear divided difference operator supplies derivatives that enable us to get the fundamental forms, if the underlying parameterization is isometric.

2.1 Parameterization

As we are interested in curvatures, it is enough to estimate partial derivatives up to second order. In order to approximate the derivatives F_u, F_v, F_{uu}, F_{uv} and F_{vv} in a specific vertex V we need a locally isometric parameterization $F(u_i, v_i) = Q_i$ of its neighborhood with $F(0, 0) := (0, 0, 0) = Q$. A parameterization is *isometric* if $\|F_u\| \approx 1$, $\|F_v\| \approx 1$ and $F_u F_v \approx 0$.

The coefficients of the fundamental forms are completely defined by those derivatives. We tested two different approaches:

Projection into tangent plane

The first way of getting a parameterization is to project the neighborhood of the vertex into a tangent plane at this vertex. The projection plane P is given by averaging the triangle normals around V resulting in the normal vector N_P. By transforming the projected points into an orthonormal basis $\{U_P, V_P, N_P\}$ we get a parameterization $F(u_{P,i}, v_{P,i}) = Q_i$. This projection method suffers from the fact that the ordering of neighbors around V is not necessarily preserved. The ordering can be destroyed if the triangle mesh is not sufficiently flat [9].

Exponential map

The second parameterization considers the lengths and the angles between adjacent edges of the triangulated surface. The ordering of neighbors is preserved when using the exponential map [2]

$$\exp(Q_i) \mapsto \|Q_i\| \left(\cos(\sum_{j=1}^{i-1} \tilde{\alpha}_j), \sin(\sum_{j=1}^{i-1} \tilde{\alpha}_j) \right)$$

where $\tilde{\alpha}_i = \textbf{flat}(\angle(Q_i, Q_{i+1}))$ with $\sum_i \tilde{\alpha}_i = 2\pi$. Therefore **flat** scales the angles between two edges in 3D so that they sum to 2π in 2D. We give two possible definitions for **flat**:

- $\textbf{flat}(\phi_i) = \phi_i \frac{2\pi}{\sum_i \phi_i}$ uniformly scales the 3D angle in a straightforward way. This will work with any configuration of 3D angles. [9]

- **flat**$(\phi_i) = \phi_i + \frac{\varepsilon}{n}$ with $\varepsilon = 2\pi - \sum_i \phi_i$ uniformly distributes the angular deficit in 3D among all 2D angles. This definition is optimal in a sense of projecting the vector $(\phi_i)_i \in \mathbb{R}^n$ of 3D angles onto the hyper plane $\{x \in \mathbb{R}^n | \langle x - \frac{2\pi}{\sqrt{n}} | (\frac{1}{\sqrt{n}})_i \rangle = 0\}$ (least squares approximation of 3D angles in 2D), where angles sum up to 2π. For rather asymmetric configurations it may result in negative angles, though.

2.2 Surface fitting

The surface $F(u, v)$ can locally be approximated by a biquadratic Taylor polynomal

$$F(u,v) = uF_u + vF_v + \frac{u^2}{2}F_{uu} + uvF_{uv} + \frac{v^2}{2}F_{vv}$$

Recall that we shifted the origin and chose our parameterization so that $Q = (0,0,0) = F(0,0)$. Fitting a second order surface to a vertex V and its neighbors is straightforward: By utilizing the parameterization $F(u_i, v_i) = Q_i$ we get a system of n linear equations

$$\mathbf{VF} = \mathbf{Q}$$

with $\mathbf{V} = (u_i, v_i, \frac{u_i^2}{2}, u_i v_i, \frac{v_i^2}{2})_i$, $\mathbf{F} = (F_u, F_v, F_{uu}, F_{uv}, F_{vv})^\top$ and $\mathbf{Q} = (Q_i)_i^\top$. The (least squares resp. least norm) solution of this linear system is

$$\mathbf{F} = \begin{cases} \mathbf{V}^\top(\mathbf{V}\mathbf{V}^\top)^{-1}\mathbf{Q} & : n < 5 \\ \mathbf{V}^{-1}\mathbf{Q} & : n = 5 \\ (\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{Q} & : n > 5 \end{cases}$$

An alternative approach proposed in [9] is to switch to another set of basis functions if the $\mathbf{V}$ matrix is ill-conditioned or $n < 5$.

The resulting vector $\mathbf{F}$ contains approximations of the Taylor coefficients F_u, F_v, F_{uu}, F_{uv}, F_{vv} of the surface. This enables us to estimate further differential parameters at the vertex V of the triangulation as needed.

Tests with a variety of triangle meshes showed that this method yields $\{F_u, F_v\} \perp \{F_{uu}, F_{uv}, F_{vv}\}$ as expected for an isometric parameterization.

3 Test Results

We use a sphere and a torus as test surfaces. For each surface we construct a set of regular triangle meshes with increasing resolution of the parameter grid. In addition to those regular meshes, two variations are compared also: first, edges in the regular mesh are randomly flipped. Therefore vertices may have less or more than six neighbors. Second, parameter points are randomly displaced in addition. This results in differently shaped triangles including obtuse angled ones. The test surfaces are defined as follows:

- *Sphere:*
  $F(u,v) = \begin{pmatrix} \sin u \cos v \\ \sin u \sin v \\ \cos u \end{pmatrix}$,
  $\frac{\pi}{4} \le u, v \le \frac{3\pi}{4}$
- *Torus:*
  $F(u,v) = \begin{pmatrix} (\cos u + 2)\cos v \\ (\cos u + 2)\sin v \\ \sin u \end{pmatrix}$,
  $0 \le u, v \le 2\pi$

The parameter intervals for u and v are each uniformly divided so that 10, 13, 15, ... , 80, 100 sample points were taken from each interval. Setting up a regular triangle mesh and the randomly flipped one is straightforward. For the second variant the parameter points were displaced in each coordinate by $\frac{\Delta u}{8}, \frac{\Delta v}{8}$ times a uniformly distributed random number from the interval $[-1, 1]$. Fig. 1 shows a hidden line view the same part of the triangulated torus with regular grid, flipped edges and additional noise.

For the sphere the vicinity of the poles is ignored. For the random displacement 10 configurations were evaluated and averaged. Fig. 2 shows the standard deviation of the approximated values from the values calculated for a smooth surface (y-axis) over the spacing of the parameter grid in u and v direction (x-axis). Both parameterizations perform well. For the sphere the projection into the tangent plane is slightly better, for the torus the exponential map shows better results.

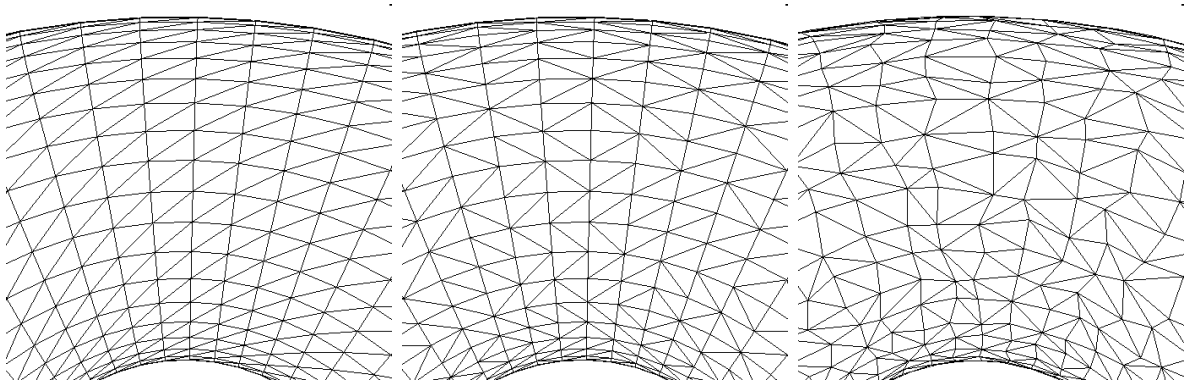

Figure 1: Triangulated torus with 50×50 parameter points. From left: regular mesh, randomly flipped edges, flipped edges and random displacement of parameter points

For both parameterizations the resulting curves show nicely the quadratic convergence of the approximation errors (error= $O(h^2)$, where h is the step width) as known from the univariate case with central differences.

4 Visualization of curvature data

For visualization of curvature data such as Gaussian curvature or mean curvature on triangulated surfaces a RGB color value is assigned to every vertex. The graphics subsystem then does the interpolation of color values over the triangles. Our aim in this section is to provide a suitable and intuitive color coding of scalar curvature values.

4.1 Global color table

Assume a scalar value d_i is given for every vertex V_i, e.g. d_i may denote any type of curvature. Now let $d_{\max} := \max\{d_i\}$ and $d_{\min} := \min\{d_i\}$. Data values are scaled by the following function $\mathbf{scale} : [d_{\min}, d_{\max}] \to [-1, 1]$ with

$$\mathbf{scale} : d \mapsto \begin{cases} -d/d_{\min} & : d < 0 \\ d/d_{\max} & : d \geq 0 \end{cases}$$

Positive and negative values are scaled separately such that the zero level is preserved. Notice that the value 0 is usually of special interest. So $d_{\min} \leq 0 \leq d_{\max}$ is assumed. If not so, the origin ("green line", see below) should be shifted appropriately.

The red and blue color components are used to indicate positive resp. negative data values. All vertices and all displayed pixels shall have equal intensity (r+g+b=1). So the green component is used to "fill up" intensity. Assume color components range from 0 to $c_{\max}$, e.g. $c_{\max} = 255$. The function $\mathbf{rgb} : [-1, 1] \to [0, c_{\max}]^3$ assigns to each value a RGB triple with intensity $c_{\max}$.

$$\mathbf{rgb} : d \mapsto \begin{cases} (0, (1+d)c_{\max}, -dc_{\max}) & : d < 0 \\ (dc_{\max}, (1-d)c_{\max}, 0) & : d \geq 0 \end{cases}$$

Fig. 3 shows the RGB mapping on the right side. Zero values are displayed pure green, $d_{\min}$ and $d_{\max}$ result in blue and red respectively.

Data values $d \in [d_{\min}, d_{\max}]$ can now be mapped to RGB values $\mathbf{rgb}(\mathbf{scale}(d))$. Many applications need enhanced contrast in the vicinity of zero and less near $d_{\min}$ resp. $d_{\max}$. Therefore a new parameter $\gamma \in (0, 1]$ is introduced that adjusts the "contrast" of the visualized data. Then value d is mapped to a RGB triple by

$$\mathbf{rgb}(\mathbf{scale}(d)^\gamma)$$

For $\gamma = 1$ we obtain the original mapping. With decreasing γ, resolution increases for values near 0, i.e. a greater range in the color table is used for those values. Fig. 3 illustrates the color coding for $\gamma = 1$ and $\gamma < 1$ (left side).

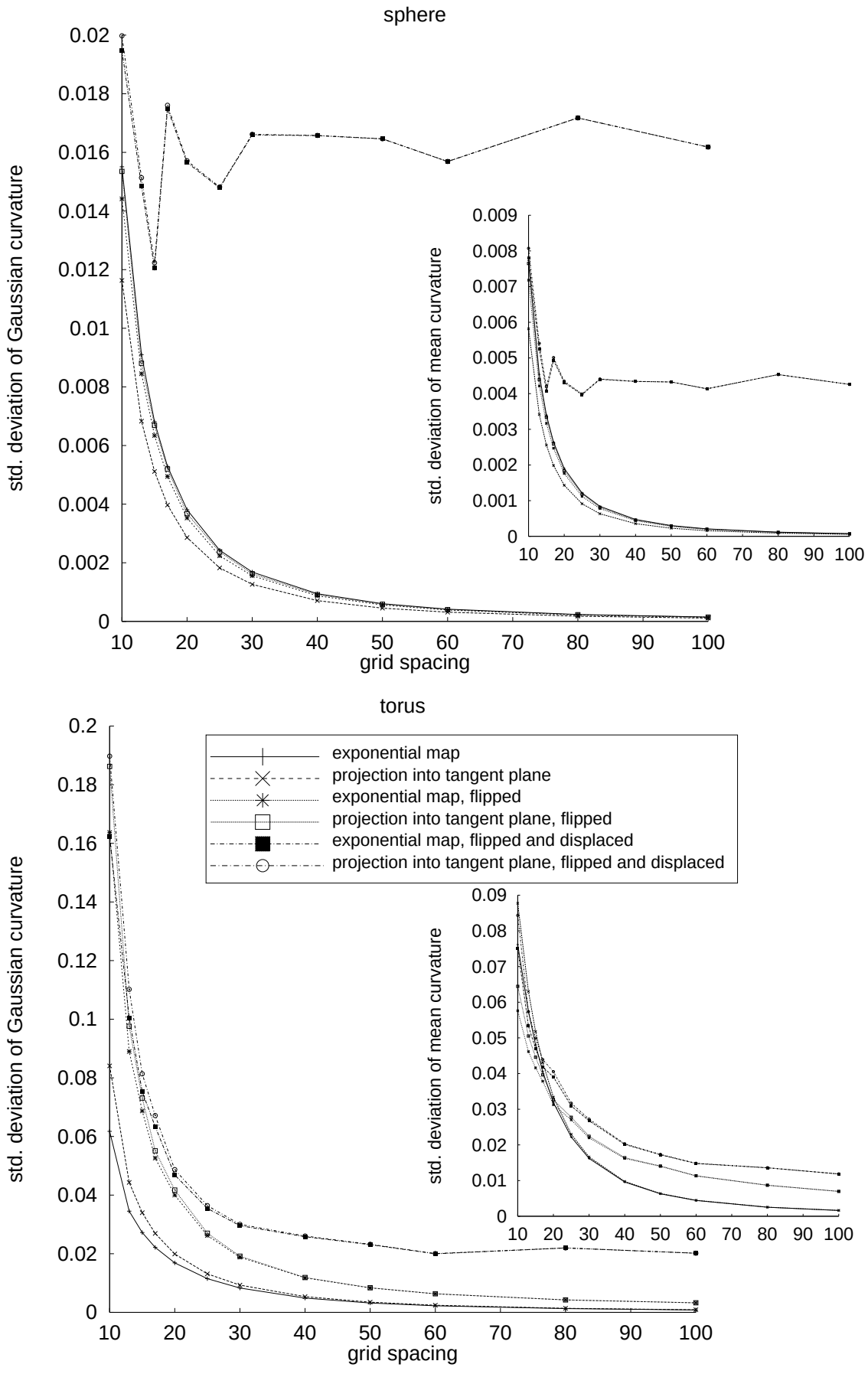

Figure 2: Approximization errors. First row: Sphere. Gaussian (large) and mean curvature (small); second row: Torus. Gaussian (large) and mean curvature (small)

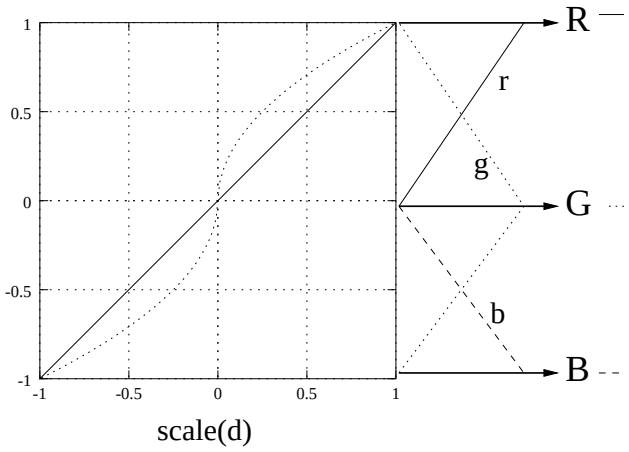

Figure 3: Color coding. d is mapped to $[-1, 1]$ by **scale**, resolution near 0 may be enhanced by using $\gamma < 1$ (left), the transformed value is then coded as (r, g, b) (right).

Figure 4: Maximal curvature on the bust model with contrast $\gamma = \frac{1}{2}$, without (left) and with (right) median filtering.

4.2 Filtering

Filtering addresses two problems: reducing high frequency noise and detecting outliers that disturb the color table.

If the input data are point samples from a real object, we have to deal with high frequency noise. Operating directly on the values obtained from curvature approximation might give undesirable results. Therefore some filtering should be applied. Fig. 4.2 shows such a sampled surface with and without filtering. Notice how artifacts on surface are suppressed by the filter.

A simple median filter that considers a vertex and its neighbors gives good results. Also this task is efficiently performed, because enumerating the neighbors of a vertex is a simple operation, if appropriate data structures are used [1].

Outliers on the input data may entail an unusable color table by significantly enlarging the $[d_{\min}, d_{\max}]$ interval. Such values should be ignored when calculating $d_{\min}$ and $d_{\max}$. So we disregard the 5% highest and lowest values each when setting up the color table. For the color mapping all $d > d_{\max}$ are clamped to $d_{\max}$, analog for $d_{\min}$.

4.3 Local color table

If a user is not interested in absolute curvature values but more in the local changes of curvature it might be better to increase the contrast in the color coding scheme. If the local data values lie in a small interval compared to $[d_{\min}, d_{\max}]$, the only way of getting a usable view is to shift the "green-line" as needed and to decrease the contrast parameter γ.

It is more convenient to provide a locally adapted color table for every vertex. Therefore the considered interval of curvature values should not be globally defined but determined locally for every vertex.

Let $[d^n_{\min,i}, d^n_{\max,i}]$ be that interval for every vertex V_i. Then $d^n_{\min,i}$ is the minimal d_j in a n-neighborhood of V_i, analog $d^n_{\max,i}$. A *n-neighborhood* of V_i is defined as $\mathbf{nhd}^n\{V_i\}$, where

$$\mathbf{nhd}\{V_i\} = \{V_i\} \cup \{V_j | \exists\, \mathrm{edge}(V_i, V_j)\}$$
$$\mathbf{nhd}\{V_{i,1}, \ldots, V_{i,k}\} = \bigcup_{1 \le \mu \le k} \mathbf{nhd}\{V_{i,\mu}\}$$
$$\mathbf{nhd}^{n+1}\{V_i\} = \mathbf{nhd}(\mathbf{nhd}^n\{V_i\})$$

Given a triangle mesh with N vertices and six neighbors per vertex on average, it is rather expensive ($O(6^n N)$) to compute a n-

neighborhood for every vertex using the above definition. Finding a minimum and/or maximum value in $\mathbf{nhd}^n\{V_i\}$ is fortunately much cheaper: one can get all extrema by iterating n times over all vertices ($O(nN)$). The following algorithm collects $d^n_{\min,i}$ and $d^n_{\max,i}$ from $d_i\,(1 \le i \le N)$:

1. `for all vertices` V_i, $i = 1, \ldots, n$:
   (a) `initialize` $d^n_{\min,i} := d_i$, $d^n_{\max,i} := d_i$
2. `for` $k = 1, \ldots, n$:
   (a) `for` $i = 1, \ldots, N$: `copy`
      i. $t_{\min} := d^n_{\min,i}$, $t_{\max} := d^n_{\max,i}$
   (b) `for` $i = 1, \ldots, N$:
      i. `for` $j = 1, \ldots, \#\text{neighbors}(V_i)$:
         A. $t_{\min} = \min\{t_{\min}, d^n_{\min,j}\}$
         B. $t_{\max} = \max\{t_{\max}, d^n_{\max,j}\}$
   (c) `for` $i = 1, \ldots, N$: `copy`
      i. $d^n_{\min,i} := t_{\min}$, $d^n_{\max,i} := t_{\max}$

It is easy to realize that the algorithm works if you recall that $\max\{A \cup B\} = \max\{\max A, \max B\}$. Initially all vertices take $d^n_{\min,i} = d^n_{\max,i} = d_i$. In the first round ($k = 1$) every vertex V_i collects the extrema of $\mathbf{nhd}\{V_i\}$ and sets its $d^n_{\min,i}$ and $d^n_{\max,i}$ accordingly. Therefore V_i collects the extrema of $\mathbf{nhd}(\mathbf{nhd}\{V_i\})$ in the second iteration ($k = 2$), and so on.

The algorithm terminates with $d^n_{\min,i}$ and $d^n_{\max,i}$ as the extreme values of all vertices in $\mathbf{nhd}^n\{V_i\}$. Color coding for vertex V_i is then done with the RGB triple

$$\mathbf{rgb}(\mathbf{scale}^n_i(d)^\gamma)$$

where

$$\mathbf{scale}^n_i : d \mapsto \begin{cases} d/d^n_{\min,i} & : d < 0 \\ d/d^n_{\max,i} & : d \ge 0 \end{cases}$$

Fig. 5 shows the maximal curvature on a technical model with about 18000 vertices. For the global color table two different contrast levels are displayed. The local color table uses a 3- and and a 6-neighborhood. The median filter is applied to all images of the technical model.

We implemented a software tool that allows the user to view a histogram of the data values d_i. This histogram is color coded in the

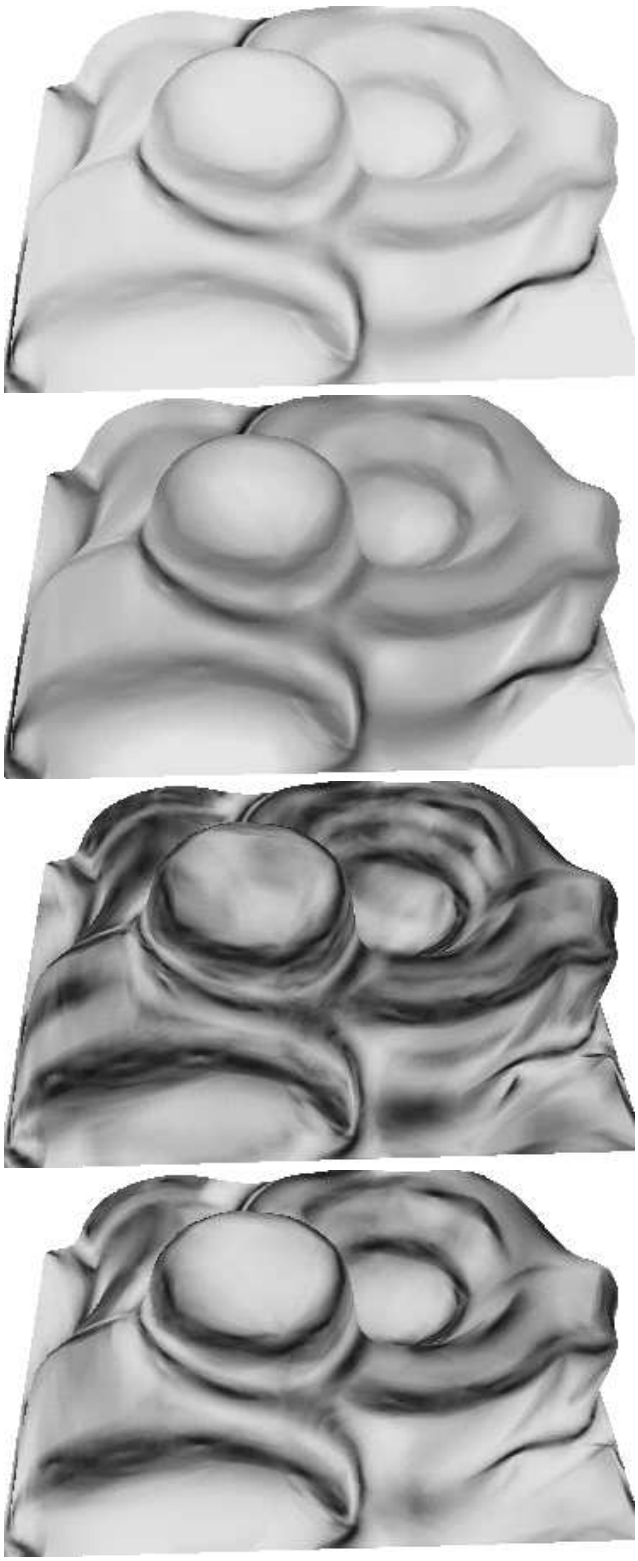

Figure 5: Median filtered maximal curvature on a technical model. From upper left to lower right: global color table with $\gamma = 1$, $\gamma = \frac{1}{2}$; local color table with $\gamma = 1$ and 3-neighborhood resp. 6-neighborhood.

same way as the surface. The user may interactively chose a data set and global or n-local color table, apply noise filtering, control the contrast parameter γ, shift the green origin-line in the histogram as well as restrict the $[d^*_{\min}, d^*_{\max}]$ interval. Fig. 6 shows a snapshot of a histogram window.

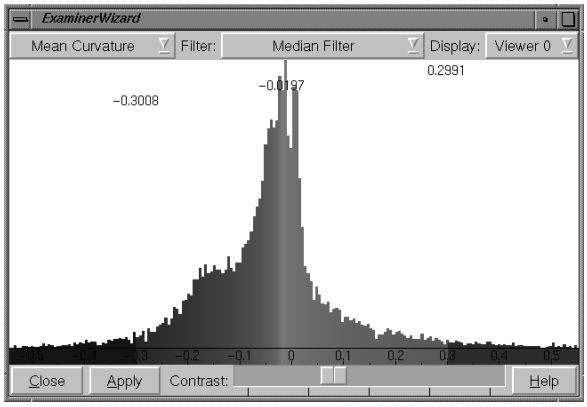

Figure 6: Snapshot of a histogram window. Median filtered mean curvature is selected, contrast is set to about 0.3, the interesting interval has been refined to about $[-0.3, 0.3]$, and the zero/green level has slightly been moved to about -0.02.

5 Conclusion

We presented a method for approximating discrete curvature on triangle meshes. Two different locally isometric parameterizations were tested. Both produce good result with the approximation error decreasing quadratically for higher grid resolution in the parameter domain.

The calculated curvature data can be visualized on the surface by the described methods. According to the application one can use a global or a locally adapted color table. Noise is effectively reduced by a simple median filter. The user is allowed to interactively vary the color coding function. With this kind of visualization technique a very helpful tool for investigation and exploration of triangulated surfaces is available.

References

[1] S. Campagna, L. Kobbelt, H.-P. Seidel. "Directed Edges — A Scalable Representation for Triangle Meshes", to appear in *The Visual Computer Journal*

[2] M. P. Do Carmo. "Differential Geometry of Curves and Surfaces", Prentice Hall, 1976

[3] G. Farin. "Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide", Academic Press, 3. ed., 1992

[4] G. Häusler, and S. Karbacher. "Reconstruction of Smoothed Polyhedral Surfaces from Multiple Range Images", *3D Image Analysis and Synthesis '97 (Proceedings)*, November 1997, pp. 192-198

[5] L. Kobbelt. "Discrete Fairing", *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, Information Geometers, 1997, pp. 101-131

[6] L. Kobbelt. "Variational Design with Parametric Meshes of Arbitrary Topology", *Creating fair and shape preserving curves and surfaces*, Teubner, 1998, pp. 189-198

[7] P. Krsek, G. Lukács, and R.R. Martin. "Algorithms for Computing Curvatures from Range Data", *The Mathematics of Surfaces VIII*, 1998, pp. 1-16

[8] Josef Stoer. "Numerische Mathematik 1", Springer, 6. ed., 1993

[9] W. Welch, and A. Witkin. "Free-Form Shape Design Using Triangulated Surfaces", *Computer Graphics (SIGGRAPH '94 Proceedings)*, July 1994, pp. 247-256