

Optimizing 3D Triangulations Using Discrete Curvature Analysis

Nira Dyn, Kai Hormann, Sun-Jeong Kim, and David Levin

Abstract. A tool for constructing a “good” 3D triangulation of a given set of vertices in 3D is developed and studied. The constructed triangulation is “optimal” in the sense that it locally minimizes a cost function which measures a certain discrete curvature over the resulting triangle mesh. The algorithm for obtaining the optimal triangulation is that of swapping edges sequentially, such that the cost function is reduced maximally by each swap. In this paper three easy-to-compute cost functions are derived using a simple model for defining discrete curvatures of triangle meshes. The results obtained by the different cost functions are compared. Operating on data sampled from simple 3D models, we compare the approximation error of the resulting optimal triangle meshes to the sampled model in various norms. The conclusion is that all three cost functions lead to similar results, and none of them can be said to be superior to the others. The triangle meshes generated by our algorithm, when serving as initial triangle meshes for the butterfly subdivision scheme, are found to improve significantly the limit butterfly-surfaces compared to arbitrary initial triangulations of the given sets of vertices. Based upon this observation, we believe that any algorithm operating on triangle meshes such as subdivision, finite element solution of PDE, or mesh simplification, can obtain better results if applied to a “good” triangle mesh with small discrete curvatures. Thus our algorithm can serve for modelling surfaces from sampled data as well as for initialization of other triangle mesh based algorithms.

§1. Introduction

Triangle meshes are commonly used for representing 3D surfaces. Given a set of vertices sampled from a smooth surface, the triangle mesh with these vertices serves as a representation (approximation) of the sampled surface. This representation depends on the choice of the connections among the vertices.

In this paper we investigate good choices of triangulations for a fixed set of vertices under the assumption that the sampled surface is smooth. Our point of view is that the given discrete set of points represents the surface in the sense that its most prominent features (creases, curvatures, etc.) can be

extracted from the data and that the representation as a triangle mesh should not add features not present in the data.

This leads us to choices of optimal triangulations relative to cost functions which measure different kinds of discrete curvatures. Starting with an arbitrary initial triangulation of the given vertices, the main algorithm we present swaps edges in a greedy way so as to maximally reduce the cost function at each step and terminates at a local minimum of the cost function. Such an algorithm was also used for optimizing 2D triangulations (approximation of bivariate functions), *e.g.* for deriving data dependent triangulations [3,6,8,9].

Alboul and van Damme, in a series of papers [1,2,11], consider a cost function which is a discrete measure of the L_1 -norm of the Gaussian curvature over the triangle mesh. Although this cost function requires heavy computations, it has a very important property. As proved in [2], for data sampled from a convex surface (convex data), swapping with this cost function leads to its unique global minimum which corresponds to the unique convex triangulation. For two of the cost functions we introduce here, the convex triangulation also seems to be the global minimum while their computation is simpler. The theoretical investigation of this observation is beyond the scope of this paper and we leave it to future work. Unfortunately, swapping edges with the greedy swapping algorithm does not always lead to the convex triangulation because the cost functions may have local minima.

We have made many numerical experiments with our cost functions and conclude that we have at hand a tool which improves significantly the visual appearance of a triangle mesh for a fixed set of vertices and enhances feature lines (“sharp edges”). For complex models, our different cost functions yield very similar results. For simple models, we also tested the approximation quality of the triangle meshes generated by the swapping algorithm with our cost functions and found all of them to reduce the approximation error significantly (see the torus example in Sec. 4).

We also realize that our algorithm can serve as a preprocessor for other algorithms operating on 3D triangle meshes (such as subdivision, finite element, simplification, etc.) by providing a better starting point for these algorithms, which results in better performance.

The outline of the paper is as follows: Section 2 presents a calculation of several discrete measures of curvature. In Section 3 we discuss the swapping algorithm and the various cost functions. Section 4 includes experiments with our different methods of optimizing 3D triangulations. An appendix with explicit formulas needed for the computation of our cost functions is the last section of the paper.

§2. Discrete Curvature Computation

From a theoretical point of view triangle meshes do not have any curvature at all, since all faces are flat and the curvature is not properly defined along edges and at vertices because the surface is not C^2 -differentiable there. But thinking of a triangle mesh as a piecewise linear approximation of an unknown

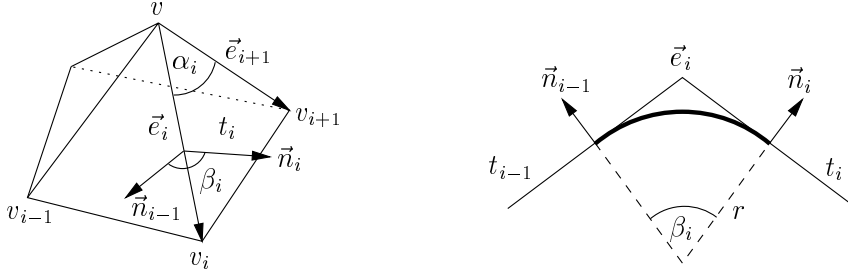


Fig. 1. A vertex v and the related variables for this local configuration (left). The blending cylinder along \vec{e}_i between triangles t_{i-1} and t_i , seen from the side (right).

smooth surface one can try to estimate the curvatures of that unknown surface using only the information that is given by the triangle mesh itself. We are particularly interested in computing the **Gaussian curvature** K and the absolute mean curvature $|H|$ at the vertices of the triangle mesh, since we base the cost functions to be minimized in the mesh optimization process on these values. But let us first fix the notation before explaining how to derive K and $|H|$ from the given data.

Notation. A triangle mesh M consists of a set of vertices $V = \{v_i\}_i \subset \mathbb{R}^3$, which are connected by a set of edges $E = \{e_j = (v_{j_1}, v_{j_2})\}_j$ and a set of triangles $T = \{t_k = \Delta(v_{k_1}, v_{k_2}, v_{k_3})\}_k$. Let $v \in V$ be a vertex of a triangle mesh M and let v_1, \dots, v_n be the ordered neighboring vertices of v (cf. Fig. 1). We define the edges $\vec{e}_i = v_i - v$ and the angles between two successive edges $\alpha_i = \angle(\vec{e}_i, \vec{e}_{i+1})$. The triangle between \vec{e}_i and \vec{e}_{i+1} is named $t_i = \Delta(v, v_i, v_{i+1})$, the corresponding face normal $\vec{n}_i = \frac{\vec{e}_i \times \vec{e}_{i+1}}{\|\vec{e}_i \times \vec{e}_{i+1}\|}$. The **dihedral angle** at an edge \vec{e}_i is the angle between the normals of the adjacent triangles, $\beta_i = \angle(\vec{n}_{i-1}, \vec{n}_i)$. Note that in these definitions we identify the index 0 with n and the index $n + 1$ with 1.

Now we can define the **integral Gaussian curvature** $\bar{K} = \bar{K}_v$ and the **integral absolute mean curvature** $|\bar{H}| = |\bar{H}_v|$ with respect to the area $S = S_v$ attributed to v by

$$\bar{K} = \int_S K = 2\pi - \sum_{i=1}^n \alpha_i \quad \text{and} \quad |\bar{H}| = \int_S |H| = \frac{1}{4} \sum_{i=1}^n \|\vec{e}_i\| |\beta_i|. \quad (1)$$

These formulas are also used by other authors [1,2,4,11], and can be understood in the following way. Suppose we replace each edge by a small cylinder of radius r that joins the adjacent faces tangentially (cf. Fig. 1) and blend these cylinders smoothly at the vertices in a C^2 manner. Now the triangle mesh is approximated by a smooth surface, K and $|H|$ are integrable functions on it and we can apply well-known theorems from differential geometry [5]. A straightforward computation [1,11] finally results in formulas (1) that depend neither on the choice of r nor on the specific blending method at the vertices.

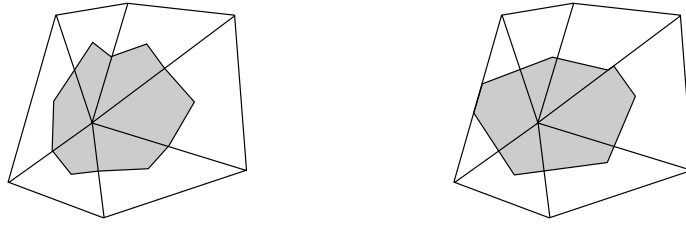


Fig. 2. Barycentric area S^B (left) and Voronoi area S^V (right) around a vertex.

To derive the curvatures at the vertex v from these integral values, we assume the curvatures to be uniformly distributed around the vertex, and simply normalize by the area:

$$K = \frac{\bar{K}}{S} \quad \text{and} \quad |H| = \frac{|\bar{H}|}{S}.$$

Of course there are different ways of defining the area S_v attributed to a vertex v , which result in different curvature values. We restrict ourselves to those methods for which the areas around all vertices sum up to the area of the triangle mesh M , *i.e.*, $\sum_{v \in V} S_v = M$, since this enables us to write an integral over M as the sum of integrals over the single area patches, *e.g.* $\int_M K = \sum_{v \in V} \int_{S_v} K$. The area that is most commonly used in the literature is the **barycentric area** S^B which is one third of the area of the triangles adjacent to v , and can be constructed by connecting the edge midpoints with the barycenters of the adjacent triangles (cf. Fig. 2). However, inspired by [12], we decided to use the **Voronoi area** S^V instead, which sums up the areas of v 's local Voronoi cells restricted to the triangles adjacent to v , according to the Euclidean distance to the vertices of the triangle mesh (cf. Fig. 2). The explicit formulas for computing the Voronoi area are given in the appendix (Sec. 5).

Besides the Gaussian and the absolute mean curvature, we are also interested in the sum of the absolute principle curvatures $|\kappa_1|$ and $|\kappa_2|$. From the relations $K = \kappa_1 \kappa_2$ and $H = \frac{1}{2}(\kappa_1 + \kappa_2)$, we get $\kappa_{1,2} = H \pm \sqrt{H^2 - K}$. Moreover, we can get the sum of the absolute principle curvatures without knowing H but only $|H|$:

$$|\kappa_1| + |\kappa_2| = \begin{cases} 2|H|, & \text{if } K \geq 0, \\ 2\sqrt{|H|^2 - K}, & \text{otherwise.} \end{cases}$$

Note that $|\kappa_1| + |\kappa_2|$ is always a real number, even if $|H|^2 = H^2 < K$, which corresponds to complex principle curvature values. Of course, this cannot happen for smooth surfaces, but since we are dealing with discrete curvatures, it can occur for some vertices.

§3. Mesh Optimization

Over the past few years the problem of **fairing** (or **smoothing**) meshes has received a lot of attention. The need for these methods ranges from technical

applications, where the noise that is due to measurement errors has to be removed from measured data, to entertainment applications, that require triangulated 3D models with a pleasing visual appearance. The usual approach in mesh fairing is to move the vertices of the mesh such that a certain energy functional is minimized [4,7,10]. However, these methods cannot be applied whenever the position of the original data points must not be changed, *e.g.* in numerical simulations or surface interpolation. The only parameter that is left to change is the triangulation of the data points. While the optimization of 2D triangulations has been studied thoroughly in the early nineties [3,6,8,9], little is known so far for the 3D case [1,2,11].

One of the energy functionals that has often been used for the fairing of meshes as well as for the fairing of continuous surfaces is the thin plate energy

$$F_{\text{TP}} = \int 4a H^2 + 2(1 - a - b) K,$$

with certain parameters $a, b \in \mathbb{R}$. However, for closed surfaces or surfaces with a fixed boundary this expression can be simplified to $F_{\text{TP}} = 4a \int H^2$, because the theorem of Gauss-Bonnet states that $\int K$ is constant in these cases. Note that this also holds for the discrete version of the integral Gaussian curvature:

$$\int_M K = \sum_v \int_{S_v} K = \sum_v \left(2\pi - \sum_{i=1}^{n_v} \alpha_{v_i} \right) = (\#V) 2\pi - (\#T) \pi = 2\pi \chi(M),$$

where $\chi(M)$ is the Euler characteristic of M . Since we are not going to swap the boundary edges of meshes with boundary, we can assume constant integral Gaussian curvature in our setting.

Minimizing F_{TP} equals the minimization of H in the L_2 -norm, $\|H\|_2 = (\int H^2)^{1/2}$. Likewise, the minimization of the integral absolute mean curvature relates to the L_1 -norm of H , $\|H\|_1 = \int |H|$. Besides these two energy functionals we have also used the L_1 -norm of the principle curvatures, $\|\kappa\|_1 = \int |\kappa_1| + |\kappa_2|$ as an optimization criterion. Note that the minimum of $\|\kappa\|_2$ equals the minimum of $\|H\|_2$, since $\kappa_1^2 + \kappa_2^2 = 4H^2 - 2K$. Using (1), these three energy functionals are given by

$$F_1 = \|H\|_2^2 = \int_M |H|^2 = \sum_{v \in V} \frac{1}{S_v} |\bar{H}_v|^2,$$

$$F_2 = \|H\|_1 = \int_M |H| = \sum_{v \in V} |\bar{H}_v|,$$

$$F_3 = \|\kappa\|_1 = \int_M |\kappa_1| + |\kappa_2| = \sum_{v \in V} \begin{cases} 2|\bar{H}_v|, & \text{if } \bar{K}_v \geq 0, \\ 2\sqrt{|\bar{H}_v|^2 - S_v \bar{K}_v}, & \text{otherwise.} \end{cases}$$

Choosing one of these three energy functionals as a cost function F and starting with an initial triangle mesh M , we perform a local swapping algorithm that decreases the cost function in each step. The key ingredients of this

algorithm are the determination of a swap value s_j for each edge e_j and the use of a priority queue P . The swap value is the difference between the value of the cost function before and after swapping the corresponding edge and indicates the reduction of the cost function caused by this edge swap. Note that the swapping operation is not defined for boundary edges, and should be forbidden for edges that connect to a vertex of valence three, since it would result in two identical edges and two triangles glued together (cf. Fig. 3). We avoid swapping those edges by simply setting their swap value to $-\infty$. The priority queue P is a permutation on the set of integers $\{1, \dots, \#E\}$, such that $s_{P(i)} \geq s_{P(j)}$ for all $1 \leq i < j \leq \#E$. The main advantage of using such a priority queue is the low complexity in building and updating it. Furthermore, $P(1)$ is always the index of the edge with the largest swap value and testing $s_{P(1)} > 0$ tells whether the cost function can be further reduced by swapping one of the edges or not. When an edge swap is actually carried out, the swap values of the edges in the 2-neighborhood of this edge change (cf. Fig. 3) and the priority queue has to be updated. Defining the 1-neighborhood $N(I)$ of a set of indices $I \subset \{1, \dots, \#E\}$ by

$$N(I) = \{j : \exists i \in I \text{ and } t \in T : e_i \text{ and } e_j \text{ are edges of } t\},$$

the index set of the 2-neighborhood of an edge e_i is given by $N^2(\{i\}) = N(N(\{i\}))$. With these definitions, the local swapping algorithm can be stated in the following way.

The local swapping algorithm.

```

swap_value (j)
  if  $e_j$  is swappable
    return ( $F_{\text{before}} - F_{\text{after}}$ )
  else
    return ( $-\infty$ )

initialization
  for  $j = 1, \dots, \#E$  do
     $s_j := \text{swap\_value}(j)$ 
  set_up ( $P$ )

optimization
  while  $s_{P(1)} > 0$  do
    swap  $e_{P(1)}$ 
    for  $j \in N^2(\{P(1)\})$  do
       $s_j := \text{swap\_value}(j)$ 
    update ( $P$ )

```

As the number of all possible triangulations of the given data is finite and the cost function is decreased by each swap, this algorithm is guaranteed to terminate in a finite number of steps. Unfortunately, it is generally impossible to determine whether the algorithm reaches a global minimum or not. For the L_1 -norm of the Gaussian curvature, $\|K\|_1$, Alboul and van Damme could show

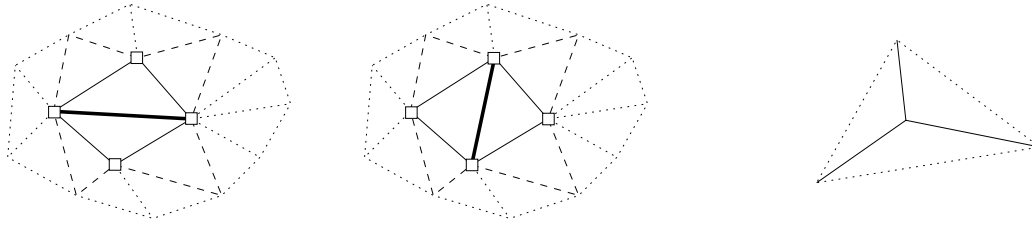


Fig. 3. Swapping an edge (left), the computation of its swap value involves only the discrete curvatures at the vertices marked by squares. After the edge swap, the swap values of the swapped edge (thick line), the edges in the 1-neighborhood (normal lines) and the edges in the 2-neighborhood (dashed lines) have to be updated. Swapping edges that connect to vertices of valence three is forbidden (right).

that in case of convex data this optimization strategy always converges to the global minimum, which is the convex triangulation [2]. We have also tested the three cost functions on convex data and observed that the convex triangulation seems to be the global optimum for F_2 and F_3 . Proving this property is a subject for future investigations. Nevertheless, all three cost functions may have local minima at which the local swapping algorithm might get trapped, *i.e.*, it stops at a triangulation of the convex data that still contains concave edges.

We have also tested a slight modification of the local swapping algorithm that is guaranteed to find the same or a better local minimum. This **look-ahead** strategy determines the swap value s_j in a different way [13]. Whenever $s_j \leq 0$, it is tested whether swapping one of the four edges e_k , $k \in N(\{j\})$ in the 1-neighborhood can decrease the cost function, *i.e.*, the combined swap value $s_{j,k} = s_j + s_k$ is positive for some k . If such a **double swap** can be found, the largest $s_{j,k}$ is used as a sorting criterion for the priority queue instead of s_j and both edges e_j and e_k are swapped when e_j reaches the head of the queue, *i.e.*, $P(1) = j$.

§4. Examples

We have tested our optimization method with the various cost functions on many different data sets, ranging from small sets sampled from simple objects (*e.g.* sphere, cylinder, or torus) to complex models with several thousand vertices. The first class of data sets we investigated were convex sets of vertices, since there are many reasons (*e.g.* shape preservation or tightness [11]) to judge the convex triangulation of the vertices to be the best among all possible triangulations. In almost all cases we have tested, minimizing any of the three cost functions led to the convex triangulation, regardless of the initial triangulation. But there exist configurations for which the local as well as the look-ahead swapping algorithm terminate at a non-convex triangulation (cf. Fig. 4).

Furthermore we have tested the approximation quality of the triangle meshes generated by the swapping algorithm with our three cost functions for simple objects like the torus in Fig. 5. The initial triangle mesh consists of

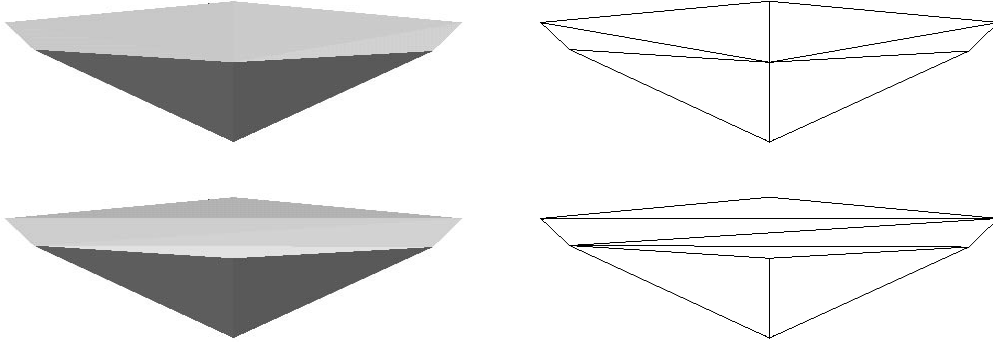


Fig. 4. A convex data set with 7 vertices $v_1 = (-12, 0, 0)$, $v_2 = (-10, -2, -\frac{1}{10})$, $v_3 = (0, -\frac{5}{2}, -\frac{1}{2})$, $v_4 = (10, -2, -\frac{1}{5})$, $v_5 = (12, 0, 0)$, $v_6 = (0, \frac{5}{2}, -\frac{1}{2})$, $v_7 = (0, 0, -10)$, for which the 10 triangles $\Delta(v_1, v_2, v_3)$, $\Delta(v_3, v_5, v_6)$, $\Delta(v_1, v_3, v_6)$, $\Delta(v_3, v_5, v_6)$, $\Delta(v_7, v_2, v_1)$, $\Delta(v_7, v_3, v_2)$, $\Delta(v_7, v_4, v_3)$, $\Delta(v_7, v_5, v_4)$, $\Delta(v_7, v_6, v_5)$, $\Delta(v_7, v_1, v_6)$ form a non-convex triangulation (top) that corresponds to a local minimum for any of the cost functions F_1, F_2, F_3 . The convex triangulation of the data is shown on the bottom.

$m = 12$ by $n = 6$ data points

$$P_{ij} = \begin{pmatrix} \cos \sigma_i (R + r \cos \varphi_{ij}) \\ \sin \sigma_i (R + r \cos \varphi_{ij}) \\ r \sin \varphi_{ij} \end{pmatrix}, \quad i = 0, \dots, m-1, \quad j = 0, \dots, n-1,$$

$$\text{with } \sigma_i = \frac{i}{m} 2\pi, \text{ and } \varphi_{ij} = \begin{cases} \frac{j}{n} 2\pi, & \text{if } i \text{ is even,} \\ \frac{j+0.5}{n} 2\pi, & \text{otherwise,} \end{cases}$$

sampled from the surface of a torus with radii $R = 5$, $r = 2$ and $2mn$ triangles $\Delta(P_{ij}, P_{i+1,j}, P_{i,j+1})$, $\Delta(P_{i,j+1}, P_{i+1,j}, P_{i+1,j+1})$, $i = 0, \dots, m-1$, $j = 0, \dots, n-1$.

Fig. 5 shows the initial triangle mesh as well as the results of the optimization process using the different cost functions, their approximation errors to the torus are listed in Tab. 1. By testing many choices of the parameters (m, n, R, r) , we observed that the minimization of any of the three cost functions reduces the approximation error significantly but none of the results can be said to be superior to the others.

	L_1	L_2	L_∞
initial	0.2342743546	0.2824122092	0.7939588898
F_1	0.1581226238	0.1887044119	0.4019238949
F_2	0.1640362102	0.1931971435	0.4019238949
F_3	0.1660948138	0.1945930432	0.3892151477

Tab. 1. Approximation error of the initial and the optimized meshes.

One important aspect of our optimization process is, that it can be used as a preprocessor for other algorithms operating on triangle meshes. As an

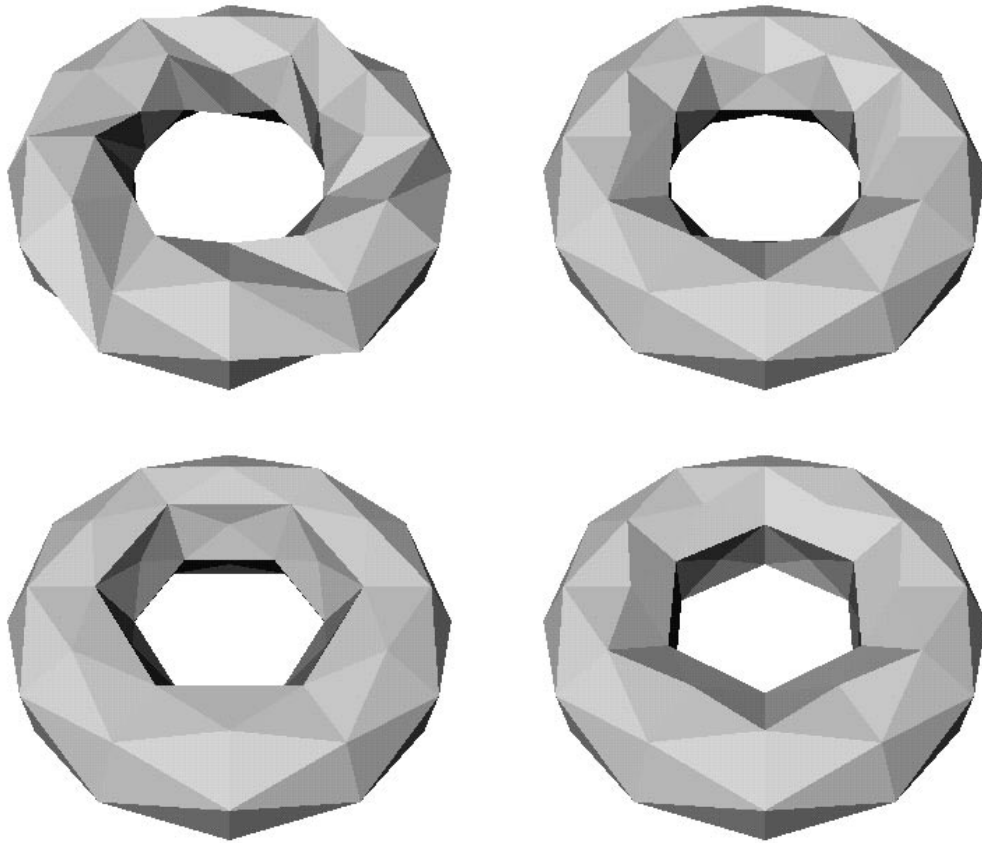


Fig. 5. Optimizing a triangle mesh sampled from a torus (top left) using different cost functions: F_1 (top right), F_2 (bottom left), and F_3 (bottom right).

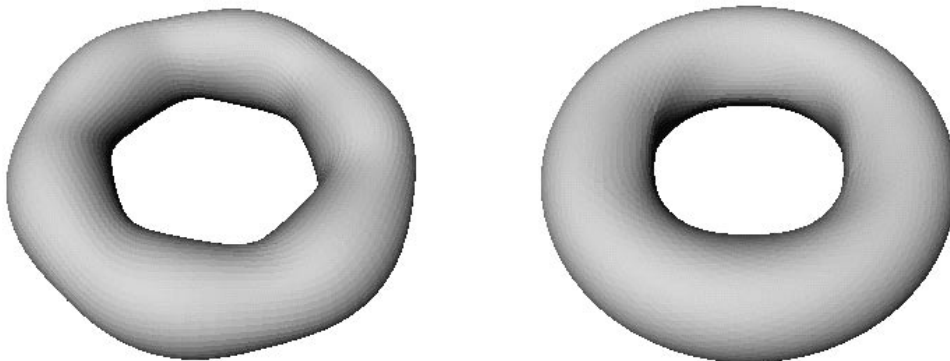


Fig. 6. The initial (left) and the optimized mesh (right) after three butterfly-subdivision steps.

example, Fig. 6 shows the result of applying three butterfly-subdivision steps to the initial triangle mesh and to the one obtained by minimizing F_1 .

We have also tested our method with more complex data sets, like the Spock head in Fig. 7 with 16,386 vertices, the technical data set with 4,100

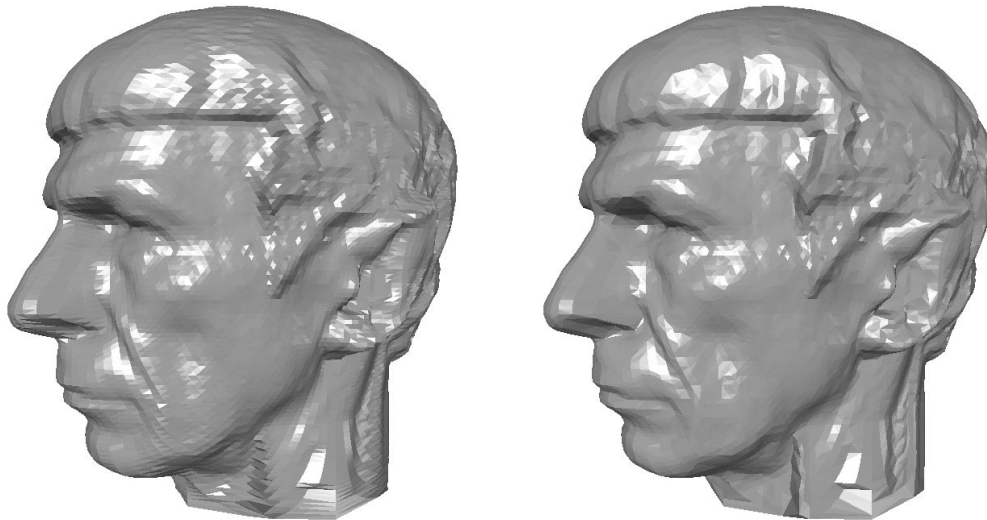


Fig. 7. Mr. Spock's head: initial (left) and optimized triangle mesh (right).

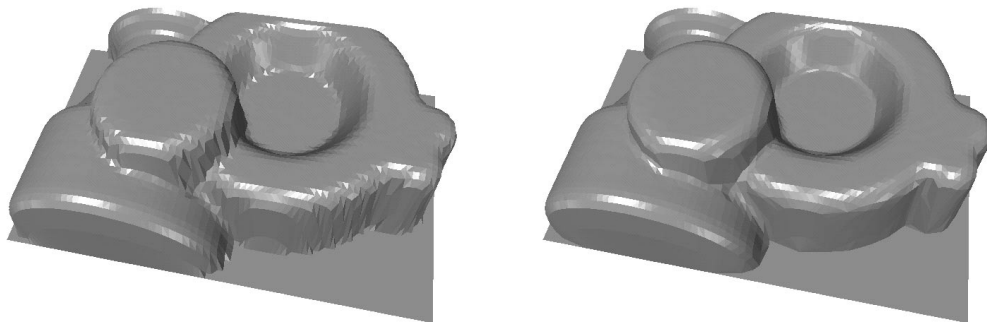


Fig. 8. A technical data set: initial (left) and optimized triangle mesh (right).

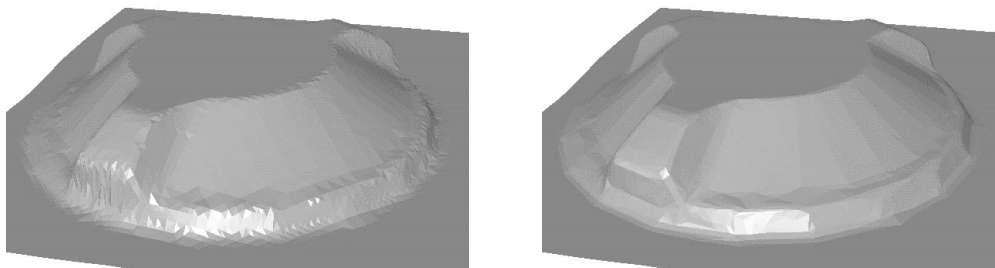


Fig. 9. Data set of a tank cap: initial (left) and optimized triangle mesh.

vertices shown in Fig. 8, or the data set of a tank cap with 3,374 vertices in Fig. 9. All figures show the initial triangle mesh and the one obtained by minimizing F_2 ; the results of minimizing the other two cost functions are very similar. Note how the optimized meshes look much smoother and how the feature lines are enhanced in Fig. 8 and Fig. 9.

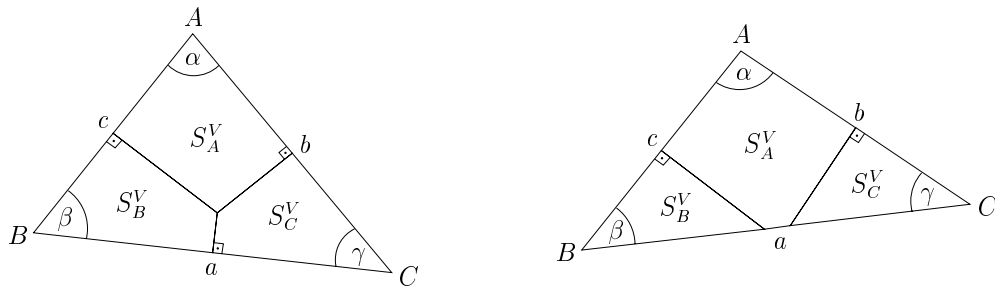


Fig. 10. Areas of the local Voronoi cells restricted to a triangle for a non-obtuse (left) and an obtuse (right) triangle.

As a concluding remark we would like to stress again that all three cost functions behaved very similar within the scope of our investigations except for convex data, and it is hard to tell which one performs best. But since the computation of the absolute mean curvature is the simplest and does not involve the calculation of the area around the vertices, and since with F_2 , in the convex data case the optimal triangulation seems to be the convex one, we favor the use of F_2 .

§5. Appendix

To determine the areas of the local Voronoi cells restricted to a triangle, we have to distinguish between obtuse and non-obtuse triangles (cf. Fig. 10). In the latter case they are given by

$$S_A^V = \frac{1}{8} (b^2 \cot(\beta) + c^2 \cot(\gamma)),$$

and likewise for S_B^V and S_C^V . For obtuse triangles,

$$S_B^V = \frac{1}{8} c^2 \tan(\beta), \quad S_C^V = \frac{1}{8} b^2 \tan(\gamma), \quad S_A^V = S_\Delta - S_B^V - S_C^V.$$

Acknowledgments. This work was supported in part by the European Union research project “Multiresolution in Geometric Modeling (MINGLE)” under grant HPRN-CT-1999-00117, by the Deutsche Forschungsgemeinschaft through the Sonderforschungsbereich 603 “Modellbasierte Analyse und Visualisierung komplexer Sensordaten” and by The Israel Science Foundation – Center of Excellence Program.

References

1. Alboul, L. and R. van Damme, Polyhedral metrics in surface reconstruction, in *The Mathematics of Surfaces VI*, G. Mullineux (ed.), Clarendon Press, Oxford, 1996, 171–200.

2. Alboul, L. and R. van Damme, Polyhedral metrics in surface reconstruction: tight triangulations, in *The Mathematics of Surfaces VII*, T. Goodman and R. Martin (eds.), Clarendon Press, Oxford, 1997, 309–336.
3. Baszenski, G. and L. L. Schumaker, Use of simulated annealing to construct triangular facet surfaces, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press, New York, 1991, 27–32.
4. Desbrun, M., M. Meyer, P. Schröder, and A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, *Computer Graphics (SIGGRAPH '99 Proceedings)* **33** (1999), 317–324.
5. do Carmo, M. P., *Differential Geometry of Curves and Surfaces*, Prentice-Hall Inc., New Jersey, 1976.
6. Dyn, N., D. Levin, and S. Rippa, Data dependent triangulations for piecewise linear interpolation, *IMA J. Numer. Anal.* **10** (1990), 137–154.
7. Kobbelt, L., Discrete fairing, in *The Mathematics of Surfaces VII*, T. Goodman and R. Martin (eds.), Clarendon Press, Oxford, 1997, 101–131.
8. Quak, E. and L. L. Schumaker, Cubic spline interpolation using data dependent triangulations, *Comput. Aided Geom. Design* **7** (1990), 293–301.
9. Schumaker, L. L., Computing optimal triangulations using simulated annealing, *Comput. Aided Geom. Design* **10** (1993), 329–345.
10. Taubin, G., A signal processing approach to fair surface design, *Computer Graphics (SIGGRAPH '95 Proceedings)* **29** (1995), 351–358.
11. van Damme, R. and L. Alboul, Tight triangulations, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1995, 517–526.
12. Warren, J., private communication, 1999.
13. Yu, X., B. Morse, and T. Sederberg, Image reconstruction using data-dependent triangulations, preprint.

Nira Dyn, Sun-Jeong Kim and David Levin
Dept. of Applied Mathematics
School of Mathematical Sciences
Tel Aviv University
Tel Aviv 69978, Israel
{niradyn,sjkim,levin}@math.tau.ac.il

Kai Hormann
Computer Graphics Group
University of Erlangen-Nürnberg
Am Weichselgarten 9
91058 Erlangen, Germany
hormann@informatik.uni-erlangen.de