

ESTIMATING THE TENSOR OF CURVATURE OF A SURFACE FROM A POLYHEDRAL APPROXIMATION

Gabriel Taubin

IBM T.J.Watson Research Center
P.O.Box 704, Yorktown Heights, NY 10598
taubin@watson.ibm.com

Abstract

Estimating principal curvatures and principal directions of a surface from a polyhedral approximation with a large number of small faces, such as those produced by iso-surface construction algorithms, has become a basic step in many computer vision algorithms. Particularly in those targeted at medical applications. In this paper we describe a method to estimate the tensor of curvature of a surface at the vertices of a polyhedral approximation. Principal curvatures and principal directions are obtained by computing in closed form the eigenvalues and eigenvectors of certain 3×3 symmetric matrices defined by integral formulas, and closely related to the matrix representation of the tensor of curvature. The resulting algorithm is linear, both in time and in space, as a function of the number of vertices and faces of the polyhedral surface.

1 Introduction

It is well established in the Computer Vision literature the use of differential invariant properties – principal curvatures and principal directions – for recognition and registration purposes, particularly for free-form surfaces [13, 2, 1, 6]. And this is particularly the case in the medical imaging domain [15, 10, 14, 19]. When surfaces are estimated, they are usually approximated by polyhedral surfaces that can be visualized in a computer. In this paper we address the problem of accurately estimating the principal directions and principal curvatures of a subjacent, unknown, smooth surface from a polyhedral approximation. Some existing techniques only apply to surfaces extracted from range images [1, 5, 7]. Other techniques that only apply to iso-surfaces [12], make use of the implicit function defined on a regular three-dimensional grid around the surface [14]. Lin and Perry [11] show how to estimate the Gaussian curvature at the vertices of a triangulated surface.

Chen and Schmitt [3] describe an algorithm to esti-

mate principal curvatures at the vertices of a triangulated surface. Both this algorithm and ours are based on constructing a quadratic form at each vertex of the polyhedral surface and then computing eigenvalues (and eigenvectors) of the resulting form, but the quadratic forms are different. In our algorithm the quadratic form associated with a vertex is expressed as an integral, and is constructed in time proportional to the number of neighboring vertices. In the algorithm of Chen and Schmitt, it is the least-squares solution of an overdetermined linear system, and the complexity of constructing it is quadratic in the number of neighbors.

2 The Tensor of Curvature

The *tensor of curvature* of the surface S is the map $p \mapsto \kappa_p$ that assigns each point p of S to the function that measures the *directional curvature* $\kappa_p(T)$ of S at p in the direction of the unit length vector T , tangent to S at p . The directional curvature $\kappa_p(T)$ of a surface S at a point p in the direction of a unit length tangent vector T is defined by the identity $x''(0) = \kappa_p(T)N$, where N is the unit length normal vector to S at p , and $x(s)$ is a *normal section* to S at p parameterized by arc length, and such that $x(0) = p$ and $x'(0) = T$ [4, 20]. The directional curvature function $\kappa_p(\cdot)$ is a quadratic form [4, 20], i.e., it satisfies the identity

$$\kappa_p(T) = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} \kappa_p^{11} & \kappa_p^{12} \\ \kappa_p^{21} & \kappa_p^{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix},$$

where, $T = t_1T_1 + t_2T_2$ is a tangent vector to S at p , $\{T_1, T_2\}$ is an orthonormal basis of the tangent space to S at p , $\kappa_p^{11} = \kappa_p(T_1)$, $\kappa_p^{22} = \kappa_p(T_2)$, and $\kappa_p^{12} = \kappa_p^{21}$. The vectors $\{T_1, T_2\}$ are called *principal directions* of S at p when $\kappa_p^{12} = \kappa_p^{21} = 0$. The corresponding directional curvatures are the *principal curvatures*, which we will denote κ_p^1 and κ_p^2 , instead of κ_p^{11} and κ_p^{22} .

If we add the normal vector N to the basis $\{T_1, T_2\}$ of principal directions, we obtain an orthonormal basis $\{N, T_1, T_2\}$ of three-dimensional space, which changes

from point to point. The following formula extends the definition of the directional curvature to non-tangent directions

$$\kappa_p(T) = \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix}^t \begin{pmatrix} 0 & 0 & 0 \\ 0 & \kappa_p^1 & 0 \\ 0 & 0 & \kappa_p^2 \end{pmatrix} \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix}, \quad (1)$$

where $T = nN + t_1T_1 + t_2T_2$ is an arbitrary vector. Tangent vectors to S at p are those for which $n = 0$. If we write the vector T as a linear combination $T = u_1U_1 + u_2U_2 + u_3U_3$ of another system of orthonormal vectors $\{U_1, U_2, U_3\}$, the directional curvature now has an expression

$$\kappa_p(T) = u^t K_p u,$$

where $u = (u_1, u_2, u_3)^t$, and K_p is a 3×3 symmetric matrix which has 0, and the two principal curvatures κ_p^1 , κ_p^2 , as eigenvalues. From now on we will use as the basis $\{U_1, U_2, U_3\}$ the same Cartesian coordinate system used to specify the coordinates of points on the surface, independently of the point p on S .

The principal curvatures and principal directions of S at p can be recovered by first restricting the matrix K_p to the tangent plane to S at p , and then computing the eigenvalues and eigenvectors of the resulting 2×2 matrix. And this computation can be done in closed form using just a few arithmetic operations. This is essentially the approach taken in this paper.

3 Estimating The Tensor of Curvature

In this section we define a matrix M_p by an integral formula. This matrix that has the same eigenvectors as K_p , and their eigenvalues are related by a fixed homogeneous linear transformation. Estimating principal curvatures and principal directions of S at p reduces to diagonalizing the matrix M_p , which can be done in closed form. We finish the section with a finite differences scheme to approximate directional curvatures.

For $-\pi \leq \theta \leq \pi$, let T_θ the unit length tangent vector $T_\theta = \cos(\theta)T_1 + \sin(\theta)T_2$, where $\{T_1, T_2\}$ are the orthonormal principal directions of S at p .

According to equation (1) above

$$\kappa_p(T_\theta) = \kappa_p^1 \cos^2(\theta) + \kappa_p^2 \sin^2(\theta).$$

Let us define the symmetric matrix

$$M_p = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \kappa_p(T_\theta) T_\theta T_\theta^t d\theta. \quad (2)$$

The normal vector N is an eigenvector of this matrix associated with the eigenvalue 0, because $T_\theta T_\theta^t$ is a rank 1 matrix for every θ , and T_θ is tangent to S at p . It follows that M_p can be factorized as follows

$$M_p = T_{12}^t \begin{pmatrix} m_p^{11} & m_p^{12} \\ m_p^{21} & m_p^{22} \end{pmatrix} T_{12},$$

where $T_{12} = [T_1, T_2]$ is the 3×2 matrix constructed by concatenating the column vectors T_1 and T_2 , and $m_p^{12} = m_p^{21}$ because of the symmetry. Regarding the parameters m_p^{ij} , we first observe that the off-diagonal elements are zero:

$$\begin{aligned} m_p^{12} &= T_1^t M_p T_2 \\ &= \frac{\kappa_p^1}{2\pi} \int_{-\pi}^{+\pi} \cos^3(\theta) \sin(\theta) d\theta \\ &\quad + \frac{\kappa_p^2}{2\pi} \int_{-\pi}^{+\pi} \cos(\theta) \sin^3(\theta) d\theta = 0, \end{aligned}$$

because both integrands are odd functions of θ . This means that the two remaining eigenvectors of M_p (other than N) are the principal directions T_1 and T_2 . The corresponding eigenvalues are not the principal curvatures, though:

$$\begin{aligned} m_p^{11} &= T_1^t M_p T_1 \\ &= \frac{\kappa_p^1}{2\pi} \int_{-\pi}^{+\pi} \cos^4(\theta) d\theta \\ &\quad + \frac{\kappa_p^2}{2\pi} \int_{-\pi}^{+\pi} \cos^2(\theta) \sin^2(\theta) d\theta = \frac{3}{8}\kappa_p^1 + \frac{1}{8}\kappa_p^2 \end{aligned} \quad (3)$$

With a similar derivation we obtain

$$m_p^{22} = T_2^t M_p T_2 = \frac{1}{8}\kappa_p^1 + \frac{3}{8}\kappa_p^2. \quad (4)$$

From equations (3) and (4) we obtain the principal curvatures as functions of the nonzero eigenvalues of M_p

$$\begin{aligned} \kappa_p^1 &= 3m_p^{11} - m_p^{22} \\ \kappa_p^2 &= 3m_p^{22} - m_p^{11} \end{aligned} \quad (5)$$

To estimate the directional curvature $\kappa_p(T)$ for a unit length vector T , tangent to S at p , we consider again a smooth curve $x(s)$ parameterized by arc-length, contained in S , and such that $x'(0) = T$. In such a case we also have $x''(0) = \kappa_p(T)N$. We now expand $x(s)$ in Laurent series up to second order

$$\begin{aligned} x(s) &= x(0) + x'(0)s + \frac{1}{2}x''(0)s^2 + O(s^3) \\ &= p + T s + \frac{1}{2}\kappa_p(T)N s^2 + O(s^3), \end{aligned}$$

and observe that

$$2N^t(x(s) - p) = \kappa_p(T) s^2 + O(s^3)$$

and

$$\|x(s) - p\|^2 = s^2 + O(s^3).$$

From the previous two equations we obtain

$$\frac{2N^t(x(s) - p)}{\|x(s) - p\|^2} = \kappa_p(T) + O(s). \quad (6)$$

It follows that the directional derivative $\kappa_p(T)$ is equal to the limit

$$\kappa_p(T) = \lim_{s \rightarrow 0} \frac{2N^t(x(s) - p)}{\|x(s) - p\|^2}.$$

If q is another point on the surface, close to p but different from p , and T is the unit length normalized projection of the vector $q - p$ onto the tangent plane $\langle N \rangle^\perp$, the directional curvature can be approximated as follows

$$\kappa_p(T) \approx \frac{2N^t(q-p)}{\|q-p\|^2}. \quad (7)$$

4 The Algorithm

We now consider a polyhedral surface that we will look upon as an approximation of an unknown surface. Our goal is to estimate principal curvatures and principal directions at the vertices of the polyhedral surface using approximations to the formulas described in the previous section.

Since our current implementation only processes triangulated surfaces, we will restrict our analysis to this case. The extension to general polyhedral surfaces is trivial, and is left to the reader. A triangulated surface is usually represented as a pair of lists $S = \{V, F\}$, a list of vertices $V = \{v_i : 1 \leq i \leq n_V\}$, and a list of faces $F = \{f_k : 1 \leq k \leq n_F\}$. Each face $f_k = (i_1^k, i_2^k, i_3^k)$ is a tern of non-repeated indices of vertices, that represents itself a three dimensional triangle. We will consider both closed triangulated surfaces, and triangulated surfaces with boundary, but we will assume that the surfaces are *oriented*, and *consistent* [8]. The set of vertices that share a face with v_i will be denoted V^i . If the vertex v_j belongs to V^i , we say that v_j is a *neighbor* of v_i . The number of elements of the set V^i will be denoted $|V^i|$. The set of faces that contain vertex v_i will be denoted F^i . If the face f_k belongs to F^i , we say that f_k is *incident* to v_i . The number of elements of the set F^i will be denoted $|F^i|$.

The first task is to estimate the normal vectors at the vertices of the surface. Since the faces of the surface are planar, each face f_k has a well defined unit length normal vector N_{f_k} . Since the surface is oriented all these normal vectors point to the *same side* of the surface. We define the normal vector at a vertex v_i as the normalized weighted sum of the normals of the incident faces, with weights proportional to the surface areas of the faces

$$N_{v_i} = \frac{\sum_{f_k \in F^i} |f_k| N_{f_k}}{\|\sum_{f_k \in F^i} |f_k| N_{f_k}\|}.$$

The second task is to estimate the matrices M_{v_i} . As we mentioned above, we approximate the matrix M_{v_i} with a weighted sum over the neighborhood V^i

$$\tilde{M}_{v_i} = \sum_{v_j \in V^i} w_{ij} \kappa_{ij} T_{ij} T_{ij}^t.$$

For each neighbor v_j of v_i , we define T_{ij} as the unit length normalized projection of the vector $v_j - v_i$ onto

the tangent plane $\langle N_{v_i} \rangle^\perp$

$$T_{ij} = \frac{(I - N_{v_i} N_{v_i}^t)(v_j - v_i)}{\|(I - N_{v_i} N_{v_i}^t)(v_j - v_i)\|}$$

We approximate the directional curvature $\kappa_{v_i}(T_{ij})$ using the formula of equation (7)

$$\kappa_{ij} = \frac{2N_{v_i}^t(v_j - v_i)}{\|v_j - v_i\|^2}.$$

We choose the weight w_{ij} proportional to the sum of the surface areas of all the triangles that are incident to both vertices v_i and v_j (two if the surface is closed, and one if both vertices belong to the boundary of the surface). We set the proportionality constant to make the sum of all the weights in the neighborhood of vertex v_i equal to one

$$\sum_{v_j \in V^i} w_{ij} = 1.$$

By construction, the normal vector N_{v_i} is an eigenvector of the matrix \tilde{M}_{v_i} associated with the eigenvalue 0. To compute the two remaining eigenpairs we restrict the matrix \tilde{M}_{v_i} to the tangent plane $\langle N_{v_i} \rangle^\perp$ using a Householder transformation [9], and then diagonalize the resulting 2×2 matrix in closed form with a Givens rotation [9]. In this way the computed principal directions are constrained to be orthogonal to the normal vector N_{v_i} , even if one of the eigenvalues of \tilde{M}_{v_i} is zero, or close to zero. Of course, if the two remaining eigenvalues of \tilde{M}_{v_i} are equal, the principal directions will not be uniquely determined. But this is a problem that every algorithm to estimate principal directions will have.

Let $E_1 = (1, 0, 0)^t$ be the first coordinate vector, and let

$$W_{v_i} = \frac{E_1 \pm N_{v_i}}{\|E_1 \pm N_{v_i}\|},$$

with a minus sign if $\|E_1 - N_{v_i}\| > \|E_1 + N_{v_i}\|$, and plus sign otherwise. The Householder matrix

$$Q_{v_i} = I - 2W_{v_i} W_{v_i}^t$$

is orthogonal and has its first column equal to N_{v_i} or $-N_{v_i}$, depending on the previous choice of sign. The two other columns define an orthonormal basis of the tangent space, but not necessarily the principal directions. Let us denote these two vectors \tilde{T}_1 and \tilde{T}_2 . Since N_{v_i} is an eigenvector of \tilde{M}_{v_i} with associated eigenvalue 0, we have

$$Q_{v_i}^t \tilde{M}_{v_i} Q_{v_i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \tilde{m}_{v_i}^{11} & \tilde{m}_{v_i}^{12} \\ 0 & \tilde{m}_{v_i}^{21} & \tilde{m}_{v_i}^{22} \end{pmatrix},$$

where $\tilde{m}_{v_i}^{21} = \tilde{m}_{v_i}^{12}$. Now the 2×2 nonzero minor can be diagonalized in closed form with a Givens rotation, obtaining an angle θ such that the vectors

$$\begin{aligned} T_1 &= \cos(\theta) \tilde{T}_1 - \sin(\theta) \tilde{T}_2 \\ T_2 &= \sin(\theta) \tilde{T}_1 + \cos(\theta) \tilde{T}_2 \end{aligned}$$

are the remaining eigenvectors of \tilde{M}_{v_i} , i.e., the principal directions of the surface at v_i . The principal curvatures are obtained from the two corresponding eigenvalues of \tilde{M}_{v_i} using equation (5).

5 Surface Smoothing

Since the directional curvatures are estimated by a finite differences formula, a smoothing preprocessing step is required for noisy surfaces. The noise could be due to measurement errors or just be a systematic problem. For example, iso-surface construction algorithms show a typical faceting effect that is more or less pronounced depending on which interpolation method is used to determine the location of the surface vertices from the grid function values. In our implementation we have used a new linear algorithm for surface smoothing that applies to polyhedral surfaces of arbitrary topology, and does not produce shrinkage [16, 18]. However, any other linear, non-shrinking smoothing method for polyhedral surfaces of arbitrary topology is acceptable.

6 Experimental Results

One way to determine the accuracy of the new algorithm introduced in this paper is to compare estimated values with true values in cases where ground truth is available. To do so we first consider a smooth surface described analytically, i.e., such that the matrices M_p can be computed by evaluating certain formulas derived from the analytic description of the surface. Then we generate a triangulated surface approximation of the smooth surface making sure that the vertices of the polyhedral approximation lie on the corresponding smooth surface, i.e., that the original surface interpolates the vertices of the polyhedral surface. Then, using the algorithm described in this paper, we estimate the matrices \tilde{M}_{v_i} for each vertex v_i of the triangulated surface, and compute the corresponding analytic version M_{v_i} by evaluating the defining formulas. Finally we measure the estimation error as follows

$$\epsilon_{v_i} = 1 - \frac{\langle M_{v_i}, \tilde{M}_{v_i} \rangle_F}{\|M_{v_i}\|_F \|\tilde{M}_{v_i}\|_F},$$

and we either look at the maximum error, or at the distribution of errors across the surface. The inner product used above is the Frobenius inner product. If A and B

are matrices of the same size, the Frobenius inner product of A and B is

$$\langle A, B \rangle_F = \text{trace}(A^t B).$$

The Frobenius norm is derived from the inner product as usual

$$\|A\|_F^2 = \langle A, A \rangle_F.$$

We have two reasons to use this notion of error. First of all, the Frobenius inner product of two matrices is invariant under Euclidean coordinate transformations. The second reason is that this notion of error compares all the estimated elements at the same time.

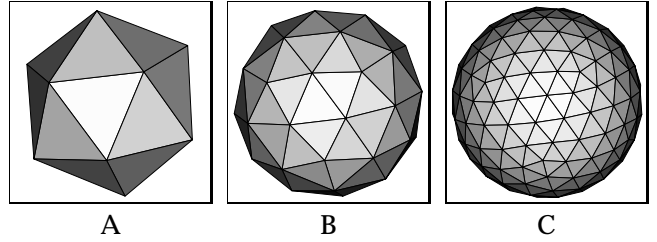


Figure 1: Polyhedral approximations of a sphere. A: Icosahedron B: Icosahedron subdivided once. C: Icosahedron subdivided twice.

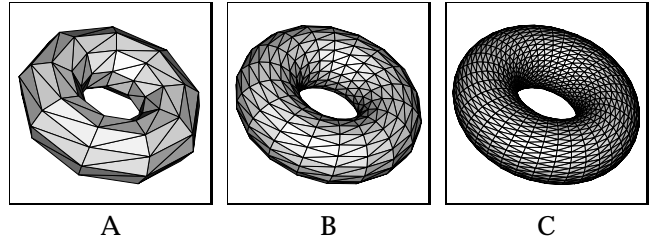


Figure 2: Polyhedral approximations of a torus. A: 9×9 grid. B: 18×18 grid. C: 36×36 grid.

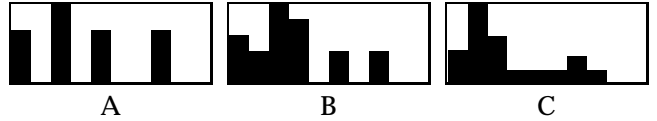


Figure 3: Estimation errors histograms for the torus. A: 9×9 grid. B: 18×18 grid. C: 36×36 grid. Horizontal axis (error value) goes from 0.00 to 0.05, and there are 20 vertical bars. No vertex showed error larger than 0.05. The vertical axis is scaled to make the maximum of each histogram equal to 1.

Figure 1 shows the simplest possible example, the case of a sphere of unit radius $\{(x, y, z) : x^2 + y^2 + z^2 = 1\}$. We approximate the sphere at the coarsest level by an icosahedron. Then we subdivide each triangular

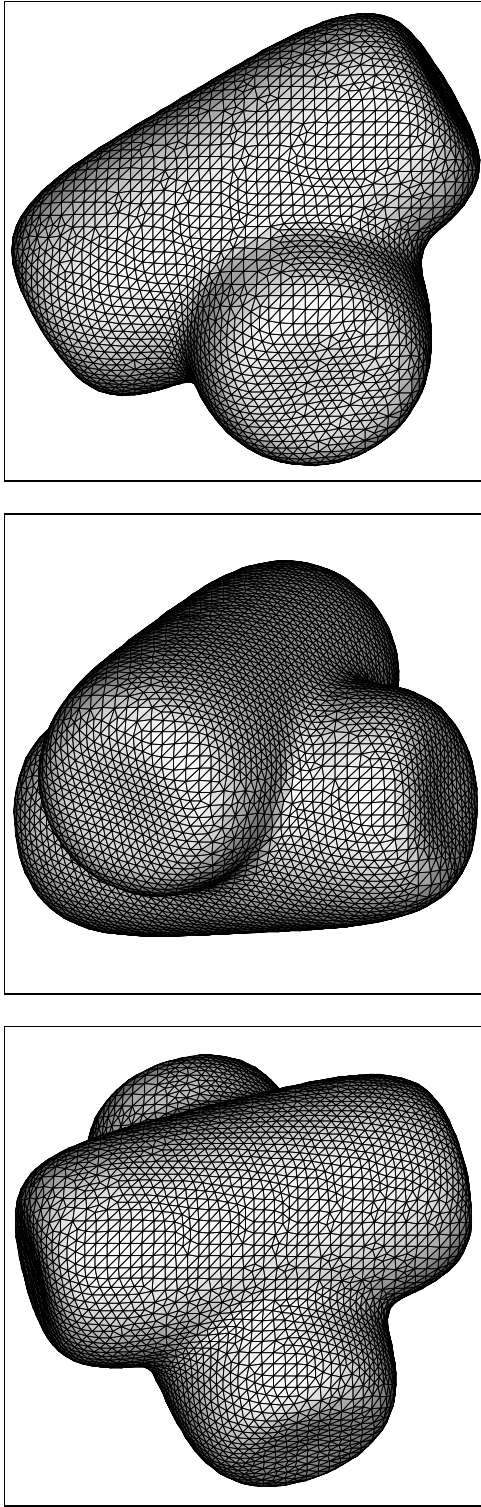


Figure 4: Three iso-surfaces constructed by evaluating the same analytic function on two grids. The three grids are of the same size, but their orientations with respect to the surface differ.

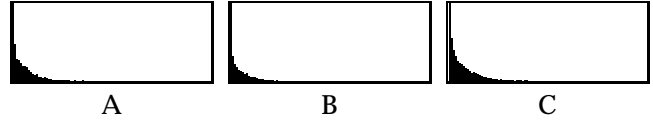


Figure 5: Estimation error histograms for the three iso-surfaces of Figure 4. Horizontal axis (error value) goes from 0.00 to 0.10, and there are 100 vertical bars. The vertical axis is scaled to make the maximum of each histogram equal to 1.

face into four triangular faces, and shift the new vertices radially to make them lie on the unit sphere. We repeat this process twice. In the three cases the errors were approximately constant for all the vertices, and less than 0.2×10^{-3} .

In the next example we test whether the error depends on the resolution of the grid or not for a parametric surface. Figure 2 shows a torus defined by the following parametric equations

$$\begin{cases} x &= (2 + \cos(u)) \cos(v) \\ y &= (2 + \cos(u)) \sin(v) \\ z &= \sin(u), \end{cases}$$

with $-\pi \leq u, v \leq \pi$. In this case we just subdivide the domain into a regular square mesh, and each square cell into two triangles. Since the surface is parameterized, the vertices of the resulting triangulated surface approximation automatically lie on the torus. Figure 3 shows the error histograms for the torus. In this case the errors were not approximately constant across the surfaces. The maximum errors were 0.0354 for 9×9 grid, 0.0359 for 18×18 grid, and 0.0364 for 36×36 grid. But, as the histograms show, the most typical (mean) error is around 0.01. It can be observed that the errors do not decrease significantly as the resolution increases. This is due to the fact that the number of neighbors of each vertex does not increase as the resolution increases.

In the last example we test whether the error depends on the triangulation or not.

In Figure 4 we consider the implicit surface $\{(x, y, z) : f(x, y, z) = 0\}$ defined by the following function

$$\begin{aligned} f(x, y, z) &= e^{-((x-1/2)^2 + y^2 - 1/4)^2 - (z^2/4)^2} \\ &+ e^{-((x+1/2)^2 + z^2 - 1/4)^2 - (y^2/4)^2} - 0.1. \end{aligned}$$

We subdivide the cube $[-3, 3] \times [-3, 3] \times [-3, 3]$ into a $51 \times 51 \times 51$ grid, evaluate the function at the nodes of this grid, and run an iso-surface construction algorithm similar to marching-cubes [12]. Then we moved the vertices of the resulting triangulated surface approximation to make them lie on the implicit surface, by line searching for a zero of the function along the straight line defined by the vertex and the gradient of

the function at the vertex. We generated Figure 4-A in this way. To see whether the algorithm was sensitive or not to the relative orientation of the grid with respect to the surface, we generated two other triangulated surfaces, shown in Figures 4-B and 4-C, with the same procedure, but with the cube that defined the grid rotated with respect to the Cartesian axes. So, the triangulated surfaces of Figures 4-B and 4-C are not equal to the triangulated surface of Figure 4-A after a rotation, but they are completely different surfaces. In fact, the surface of Figure 4-A has 5,842 vertices, the surface of Figure 4-B has 6,198 vertices, and the surface of Figure 4-C has 5,763 vertices.

Figure 5 shows the error histograms for the implicit surfaces of figure 4. Only the vertices with error less than 0.10 were considered in the histograms. In the three cases there were a few vertices with error larger than 0.10 (27 for surface A, 45 for surface B, and 70 for surface C). These outlier vertices were located in the regions of highest curvature, where the edge length should have been shorter to produce a good estimation. The mean error though, was around 0.01 in the three cases (0.011 for surface A, 0.009 for surface B, and 0.013 for surface C). This can clearly be seen in the histograms.

7 Conclusions

We have introduced in this paper a new simple algorithm for estimating the principal curvatures and principal directions of a surface at the vertices of an approximating triangulated surface.

The complexity of the algorithm is linear, both in time and in space, as a function of the number of vertices and faces of the polyhedral surface. All the computations are simple and direct. No expensive iterative numerical algorithms are needed, not even for computing the eigenvalues and eigenvectors of the matrices involved, because closed form expressions are used instead. The preliminary experiments shown in the paper make us believe that the accuracy of this new algorithm is not worse than the accuracy of other available algorithms, and perhaps much better. Further experimentation will be presented in forthcoming reports.

A longer version of this paper [17], as well as other related papers [16, 18], can be retrieved as PostScript files from the IBM Research World Wide Web server (<http://www.watson.ibm.com:8080>).

References

- [1] P.J. Besl and R.C. Jain. Invariant surface characteristics for 3d object recognition in range images. *Computer Vision, Graphics, and Image Processing*, 33:33–80, 1986.
- [2] M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. *Computer Vision, Graphics, and Image Processing*, 32(1):1–28, 1985.
- [3] X. Chen and F. Schmitt. Intrinsic surface properties from surface triangulation. In *Proceedings, European Conference on Computer Vision*, pages 739–743, Santa Margherita Ligure, Italy, May 1992.
- [4] M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.
- [5] T. Fan, G. Medioni, and R. Nevatia. Description of surfaces from range data using curvature properties. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 86–91, May 1986.
- [6] J. Feldmar and N. Ayache. Rigid and affine registration of smooth surfaces using differential properties. In Jan-Olof Eklundh, editor, *Proceedings, European Conference on Computer Vision*, number 801 in Lecture Notes in Computer Science, pages 397–406. Springer-Verlag, 1994.
- [7] P.J. Flynn and A.K. Jain. On reliable curvature estimation. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–116, June 1989.
- [8] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, MA, second edition, 1992.
- [9] G. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1983.
- [10] A. Guézic and N. Ayache. Smoothing and matching of 3d space curves. *International Journal of Computer Vision*, 12(1):79–104, January 1994.
- [11] C. Lin and M.J. Perry. Shape description using surface triangulation. In *Proceedings, IEEE Workshop on Computer Vision: Representation and Control*, pages 38–43, 1982.
- [12] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, pages 163–169, July 1987. (Proceedings SIGGRAPH).
- [13] G. Medioni and R. Nevatia. Description of 3-D surfaces using curvature properties. In *Proceedings, DARPA Image Understanding Workshop*, pages 291–299, New Orleans, LA, October 3–4 1984.
- [14] O. Monga, S. Benayoun, and O. Faugeras. From partial derivatives of 3d density images to ridge lines. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 354–359, June 1992.
- [15] P.T. Sanders and S.W. Zucker. Inferring surface trace and differential structure from 3D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):833–854, September 1990.
- [16] G. Taubin. Curve and surface smoothing without shrinkage. Technical Report RC-19536, IBM Research, April 1994. (also in Proceedings ICCV'95).
- [17] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. Technical Report RC-19860, IBM Research, December 1994. (also in Proceedings ICCV'95).
- [18] G. Taubin. A signal processing approach to fair surface design. Technical Report RC-19923, IBM Research, February 1995. (also in Proceedings SIGGRAPH'95).

- [19] J.P. Thirion. The extremal mesh and the understanding of 3d surfaces. In *IEEE Workshop on Biomedical Image Analysis*, pages 3–12, Seattle, WA, June 24–25 1994.
- [20] J.A. Thorpe. *Elementary Topics in Differential Geometry*. Springer-Verlag, 1979.