Normal Based Estimation of the Curvature Tensor for Triangular Meshes

Holger Theisel Christian Rössl

Rhaleb Zayer

Hans-Peter Seidel

MPI Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany {theisel,roessl,zayer,hpseidel}@mpi-sb.mpg.de

Abstract

We introduce a new technique for estimating the curvature tensor of a triangular mesh. The input of the algorithm is only a single triangle equipped with its (exact or estimated) vertex normals. This way we get a smooth function of the curvature tensor inside each triangle of the mesh. We show that the error of the new method is comparable with the error of a cubic fitting approach if the incorporated normals are estimated. If the exact normals of the underlying surface are available at the vertices, the error drops significantly. We demonstrate the applicability of the new estimation at a rather complex data set.

1. Introduction

The problem of estimating the curvature tensor for a triangular mesh is a well-researched area which has a variety of applications. Many techniques for this have been developed in the last decade (see e.g. [11] for a recent survey). A common approach is to locally fit a surface patch and to then derive the differential quantities analytically. The surface is locally represented by a certain neighborhood of a vertex, usually its 1-ring. In [15] this is locally approximated by a second order Taylor polynomial which directly provides the first and second order partials. Most recently, Goldfeather and Interrante [5] propose the use of a cubic approximation scheme which takes into account vertex normals in the 1-ring (which can be interpreted as enlarging the neighborhood to a 2-ring). Moreover, Cazals and Pouget [2] discuss the patch fitting approach from an approximation theory point of view. Alternative approaches apply finite differences to first estimate normal curvatures along edges. Then the curvature tensor is estimated either directly [13, 10] or via least-squares fitting [9, 8]. Recently, in [3, 1] the curvature tensor is computed as an average over a certain mesh region.

All the approaches mentioned above have in common that they yield discrete estimations of the curvature tensor in the vertices of the mesh. In this paper we propose an alternative approach to estimating the curvature tensor: instead of computing it per vertex, we do the estimation per triangle. We consider each triangle of the mesh (together with the normals in its vertices) independently and compute the curvature tensor as a smooth function on the triangle. The basic idea for doing so comes from the well-known concept of Phong-shading ([4]): given a triangle of a mesh together with its vertex normals, two linear interpolations are applied. The linear interpolation for the vertices gives the current location, while the linear interpolation of the vertex normals gives the normal for the illumination model. Although a certain error is taken to account (the normal from the piecewise linear surface generally differs from the interpolated normal), this approach has been proven to produce smooth-looking representations of meshes.

Bearing in mind that the curvature tensor of a smooth surface is completely defined by its first order partials and the first order partials of its normals, we can use the idea of Phong shading to get an estimation of the curvature tensor on a single triangle: we use the linear interpolation of the vertices to get the surface and its first order partials, while the normals and its first order partials are obtained from the linearly interpolated vertex normals. Similar to Phong shading, this introduces a certain error which is due to the application of two different linear interpolations. However, we show that this error can compete with the errors of other estimation schemes of the curvature tensor.

Recently a similar approach has been proposed in [12] where a constant curvature tensor is estimated for each triangle. Its averaging per vertex gives an estimation which can compete with the results of [5].

The rest of the paper is organized as follows: section 2 recollects the concept of the curvature tensor of a smooth surface and particularly explains how to compute it from the partials of the surface and its normals. Section 3 explains our new approach to estimate the curvature tensor in a single triangle which is equipped with vertex normals. Section 4 collects properties of this estimation. Section 5 compares the new estimation with pre-existing estimation schemes. Section 6 demonstrates the application to a rather complex test data set.

2. The Curvature Tensor of a Surface

The curvature tensor \mathbf{T} of a smooth surface is a symmetric 3×3 matrix with the eigenvalues $\kappa_1, \kappa_2, 0$ and the corresponding eigenvectors $\mathbf{k}_1, \mathbf{k}_2, \mathbf{n}$ where κ_1, κ_2 are the principal curvatures, $\mathbf{k}_1, \mathbf{k}_2$ the corresponding principal directions, and n the surface normal. T can be interpreted as describing how the unit normal changes in a small neighborhood. For a regularly parameterized parametric surface $\mathbf{x}(u, v)$, it is completely defined by its partials and the partials of the unit normals.

Given $\mathbf{x}(u, v)$ and its partials \mathbf{x}_u and \mathbf{x}_v , we compute the normalized normal n and its partials as

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|} \quad , \quad \mathbf{n}_u = \frac{d\,\mathbf{n}}{d\,u} \quad , \quad \mathbf{n}_v = \frac{d\,\mathbf{n}}{d\,v}$$

It is a well-known fact that T is completely defined by x_u , $\mathbf{x}_v, \mathbf{n}_u, \mathbf{n}_v$. These four vectors are not completely independent, in fact they have the following dependencies:

- 1. $\mathbf{x}_u, \mathbf{x}_v, \mathbf{n}_u, \mathbf{n}_v$ are coplanar (they are in the tangent plane of \mathbf{x})
- 2. $\mathbf{n}_u \mathbf{x}_v = \mathbf{n}_v \mathbf{x}_u$.

To see this, compute the partials of the equations $n^2 = 1$, $\mathbf{n} \mathbf{x}_u = 0$ and $\mathbf{n} \mathbf{x}_v = 0$ which yield both 1 and 2. Figure 1a gives an illustration.



Figure 1. a) x_u , x_v , n_u , n_v completely define T; b) computing x_u, x_v, n_u, n_v on a triangle.

The computation of T from $\mathbf{x}_u, \mathbf{x}_v, \mathbf{n}_u, \mathbf{n}_v$ is a straightforward application of classical concepts of differential geometry. We compute the coefficients of the first and second fundamental form as

$$E = \mathbf{x}_u \, \mathbf{x}_u \,, \quad F = \mathbf{x}_u \, \mathbf{x}_v \,, \quad G = \mathbf{x}_v \, \mathbf{x}_v \qquad (1)$$

$$L = -\mathbf{n}_u \mathbf{x}_u \quad , \quad M_1 = -\mathbf{n}_u \mathbf{x}_v \qquad (2)$$
$$M_2 = -\mathbf{n}_v \mathbf{x}_v \quad , \quad N = -\mathbf{n}_v \mathbf{x}_v \qquad (3)$$

$$M_2 = -\mathbf{n}_v \,\mathbf{x}_u \quad , \quad N = -\mathbf{n}_v \,\mathbf{x}_v \qquad (3)$$

and note that for a smooth surface we have $M_1 = M_2$. Then we get the Weingarten curvature matrix

$$\mathbf{W} = \begin{pmatrix} \frac{LG - M_1 F}{E G - F^2} & \frac{M_2 G - N F}{E G - F^2} \\ \frac{M_1 E - LF}{E G - F^2} & \frac{N E - M_2 F}{E G - F^2} \end{pmatrix}$$
(4)

with its eigenvalues κ_1, κ_2 and its corresponding eigenvectors $\mathbf{w}_1 = \begin{pmatrix} w_{11} \\ w_{12} \end{pmatrix}$, $\mathbf{w}_2 = \begin{pmatrix} w_{21} \\ w_{22} \end{pmatrix}$. They define the Gaussian curvature K, the Mean curvature H and principal directions $\mathbf{k}_1, \mathbf{k}_2$ as

$$K = \kappa_1 \kappa_2 \quad , \quad H = \frac{1}{2} (\kappa_1 + \kappa_2)$$
(5)
$$\mathbf{k}_1 = w_{11} \mathbf{x}_u + w_{12} \mathbf{x}_v \quad , \quad \mathbf{k}_2 = w_{21} \mathbf{x}_u + w_{22} \mathbf{x}_v.(6)$$

Now we have enough information to construct T as

$$\mathbf{T} = \mathbf{P} \, \mathbf{D} \, \mathbf{P}^{-1} \tag{7}$$

with $\mathbf{P} = (\mathbf{k}_1, \mathbf{k}_2, \mathbf{n})$ and

$$\mathbf{D} = \begin{pmatrix} \kappa_1 & 0 & 0\\ 0 & \kappa_2 & 0\\ 0 & 0 & 0 \end{pmatrix}.$$
 (8)

3. Normal Based Estimation of T

The new approach we present here considers only a single (non-degenerate) triangle with the vertices x_0, x_1, x_2 , and the corresponding (un-normalized) normals n_0 , n_1 , n_2 . Then we can obtain a point and a normal on the triangle by applying a linear interpolation of x_i and n_i respectively. We describe these linear interpolations both in barycentric coordinates (a, b, c) with a + b + c = 1 and in local cartesian coordinates (u, v) with the origin \mathbf{x}_0 and the base vectors $\mathbf{x}_1 - \mathbf{x}_0$ and $\mathbf{x}_2 - \mathbf{x}_0$:

$$\begin{aligned} \tilde{\mathbf{x}} &= \tilde{\mathbf{x}}(a,b,c) = a \, \mathbf{x}_0 + b \, \mathbf{x}_1 + c \, \mathbf{x}_2 \\ &= \tilde{\mathbf{x}}(u,v) = \mathbf{x}_0 + u \, (\mathbf{x}_1 - \mathbf{x}_0) + v \, (\mathbf{x}_2 - \mathbf{x}_0) \\ \tilde{\mathbf{n}} &= \tilde{\mathbf{n}}(a,b,c) = a \, \mathbf{n}_0 + b \, \mathbf{n}_1 + c \, \mathbf{n}_2 \\ &= \tilde{\mathbf{n}}(u,v) = \mathbf{n}_0 + u \, (\mathbf{n}_1 - \mathbf{n}_0) + v \, (\mathbf{n}_2 - \mathbf{n}_0). \end{aligned}$$

The conversion between both coordinate systems is a simple affine transformation.

The main idea now is to use $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{n}}$ to get the necessary vectors $\mathbf{x}_u, \mathbf{x}_v, \mathbf{n}_u, \mathbf{n}_v$ to compute **T**. We compute the normalized normal n and its derivatives as

$$\mathbf{n}(u,v) = \frac{\tilde{\mathbf{n}}}{\|\tilde{\mathbf{n}}\|} \quad , \quad \mathbf{n}_u = \frac{d\,\mathbf{n}}{d\,u} \quad , \quad \mathbf{n}_v = \frac{d\,\mathbf{n}}{d\,v}. \tag{10}$$

For the partials of the surface we get

$$\tilde{\mathbf{x}}_u(u,v) = \frac{d\,\tilde{\mathbf{x}}}{d\,u} = \mathbf{x}_1 - \mathbf{x}_0 \quad , \quad \tilde{\mathbf{x}}_v(u,v) = \frac{d\,\tilde{\mathbf{x}}}{d\,v} = \mathbf{x}_2 - \mathbf{x}_0.$$

In order to fulfill condition 1 of section 2, we map $\tilde{\mathbf{x}}_u$ and $\tilde{\mathbf{x}}_v$ into the plane defined by \mathbf{n}_u and \mathbf{n}_v :

$$\mathbf{x}_u = \tilde{\mathbf{x}}_u - (\mathbf{n} \, \tilde{\mathbf{x}}_u) \, \mathbf{n}$$
, $\mathbf{x}_v = \tilde{\mathbf{x}}_v - (\mathbf{n} \, \tilde{\mathbf{x}}_v) \, \mathbf{n}$. (11)

Figure 1b gives an illustration. Now we have all ingredients to compute T: we apply (1)–(8) to \mathbf{x}_u , \mathbf{x}_v , \mathbf{n}_u , \mathbf{n}_v which are computed following (10) and (11). Doing so, we obtain nice closed formulations of the Gaussian curvature K and the Mean curvature H in barycentric coordinates:

$$K(a, b, c) = \frac{\det(\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2)}{\tilde{\mathbf{n}}^2 \cdot (\tilde{\mathbf{n}} \, \tilde{\mathbf{m}})}$$
(12)

$$H(a,b,c) = \frac{1}{2} \frac{(\tilde{\mathbf{n}} \mathbf{h})}{\|\tilde{\mathbf{n}}\| \cdot (\tilde{\mathbf{n}} \tilde{\mathbf{m}})}$$
(13)

with

$$\begin{split} \tilde{\mathbf{n}} &= a \, \mathbf{n}_0 + b \, \mathbf{n}_1 + c \, \mathbf{n}_2 \\ \tilde{\mathbf{m}} &= \mathbf{r}_2 \times \mathbf{r}_0 = \mathbf{r}_0 \times \mathbf{r}_1 = \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{h} &= (\mathbf{n}_0 \times \mathbf{r}_0) + (\mathbf{n}_1 \times \mathbf{r}_1) + (\mathbf{n}_2 \times \mathbf{r}_2) \end{split}$$

and

$$\mathbf{r}_0 = \mathbf{x}_2 - \mathbf{x}_1$$
, $\mathbf{r}_1 = \mathbf{x}_0 - \mathbf{x}_2$, $\mathbf{r}_2 = \mathbf{x}_1 - \mathbf{x}_0$.

This way, $\tilde{\mathbf{n}}$ is the (un-normalized) linearly interpolated normal and $\tilde{\mathbf{m}}$ is the (un-normalized) triangle normal (i.e. $\|\tilde{\mathbf{m}}\| = \frac{1}{2} \cdot \operatorname{area}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$).

4. Properties of the estimation of T

In this section we collect a number of properties of the normal based estimation of **T**. First we explain our visualization of **T**: given a triangle $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ with the assigned normals $(\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2)$, we represent *H* and *K* as focal surfaces ([6]):

$$\mathbf{x}_{K}(a,b,c) = \tilde{\mathbf{x}}(a,b,c) + s_{K} \cdot \|K(a,b,c)\| \cdot \mathbf{n}(a,b,c)$$

$$\mathbf{x}_{H}(a,b,c) = \tilde{\mathbf{x}}(a,b,c) + s_{H} \cdot \|H(a,b,c)\| \cdot \mathbf{n}(a,b,c)$$

where s_K and s_H are global positive scaling factors controlling the distance between the mesh and the focal surfaces. To visualize \mathbf{x}_K and \mathbf{x}_H , we only show their boundary curves in a green (for positive K/H) or red (for negative K/H) color. We preferred such a geometric representation of K and H to a color coding on the surface (which is the most common approach) because the human eye reacts far more sensitive to small perturbations in shape than in color.

To visualize the estimated principal directions \mathbf{k}_1 , \mathbf{k}_2 at a certain point (a, b, c), we show the two line segments

$$egin{array}{cccc} \left(egin{array}{ccccc} \mathbf{ ilde{x}} - s_L \cdot \mathbf{k}_1 &, & \mathbf{ ilde{x}} + s_L \cdot \mathbf{k}_1 \end{array}
ight) \ \left(egin{array}{cccccc} \mathbf{ ilde{x}} - s_L \cdot \mathbf{k}_2 &, & \mathbf{ ilde{x}} + s_L \cdot \mathbf{k}_2 \end{array}
ight) \end{array}$$



Figure 2. A single triangle with visualizations of H, K (focal surfaces) and k_1/k_2 in 6 points.

where s_L is a positive global scaling factor. Figure 2 gives an illustration for visualizing H, K and $\mathbf{k}_1/\mathbf{k}_2$ for a single triangle.

The first property of \mathbf{T} to be shown is that refining the triangular mesh, our estimated \mathbf{T} converges to the \mathbf{T} of the underlying smooth surface. To do so, we consider the height surface

$$z(x,y) = \frac{1}{2}\kappa_1 x^2 + \frac{1}{2}\kappa_2 y^2$$
(14)
+ f_{30} x^3 + f_{21} x^2 y + f_{12} x y^2 + f_{03} y^3

where κ_1, κ_2 are certain constants and $f_{30}, f_{21}, f_{12}, f_{03}$ are certain scalar function of (x, y) describing the higher order terms in the Taylor approximation of (14). Note that every surface can locally be represented by (14). For this surface, the curvature tensor is well-defined at (x = 0, y = 0):

$$\mathbf{T}(0,0) = \begin{pmatrix} \kappa_1 & 0 & 0\\ 0 & \kappa_2 & 0\\ 0 & 0 & 0 \end{pmatrix}.$$

Now we consider a triangulation of (14) and repeatedly refine it in the neighborhood of (x = 0, y = 0). In fact, we consider a triangle of the vertices

$$\mathbf{x}_{0} = (t x_{0}, t y_{0}, z(t x_{0}, t y_{0}))^{T} \mathbf{x}_{1} = (t x_{1}, t y_{1}, z(t x_{1}, t y_{1}))^{T} \mathbf{x}_{2} = (t x_{2}, t y_{2}, z(t x_{2}, t y_{2}))^{T}.$$
 (15)

where (x_0, y_0) , (x_1, y_1) , (x_2, y_2) are certain constants building a non-degenerate triangle in the domain. Since \mathbf{x}_0 , \mathbf{x}_1 , \mathbf{x}_2 are on the surface defined by (14), we compute \mathbf{n}_0 , \mathbf{n}_1 , \mathbf{n}_2 as the surface normals of (14):

$$\mathbf{n}_{0} = (-z_{x}(t x_{0}, t y_{0}), -z_{y}(t x_{0}, t y_{0}), 1)^{T} \mathbf{n}_{1} = (-z_{x}(t x_{1}, t y_{1}), -z_{y}(t x_{1}, t y_{1}), 1)^{T} (16) \mathbf{n}_{2} = (-z_{x}(t x_{2}, t y_{2}), -z_{y}(t x_{2}, t y_{2}), 1)^{T} .$$

Note that for $t \to 0$ the triangle $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ collapses to the single point $(0, 0, 0)^T$ with the normal $(0, 0, 1)^T$.

Now we compute $\mathbf{T}(a, b, c)$ by applying our estimation (1)–(11) to the triangle defined by (15) and (16). We are

interested in the behavior of $\mathbf{T}(a, b, c)$ for $t \to 0$. It is a straightforward exercise in algebra¹ to show that

$$\lim_{t \to 0} \mathbf{T}(1,0,0) = \lim_{t \to 0} \mathbf{T}(0,1,0) = \lim_{t \to 0} \mathbf{T}(0,0,1)$$
$$= \begin{pmatrix} \kappa_1 & 0 & 0 \\ 0 & \kappa_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

which proves the desired property.

The next property we mention is that our estimation of T does not only depend on x_i and the directions of n_i , but also on the length of n_i . Figure 3 illustrates different curvature values for changing the length of one normal n_0 .



H, K and $\mathbf{k}_1/\mathbf{k}_2$.

Other properties of our estimation of \mathbf{T} are that \mathbf{k}_1 and \mathbf{k}_2 are in general not perpendicular to each other (i.e. that \mathbf{T} is not symmetric), and that \mathbf{T} is not continuous at the junctions of the triangles. The first property is due to the fact that our estimation gives $M_1 \neq M_2$ in (2) and (3). The second one applies because we consider the triangles independently of each other. Figure 4 illustrates this. Both properties indicate an error in our estimation of \mathbf{T} . However, in section 5 we show that this error can compete with errors of other estimation methods of \mathbf{T} .



Figure 4. *K* across two adjacent triangles is not continuous.

5. Comparing T with other estimations

To compare our estimation of \mathbf{T} with pre-existing methods, we consider triangulations of well-defined smooth surfaces where the exact curvature tensor is available at every point. Then the comparison of the exact \mathbf{T} with the estimated one gives a measure of the quality of the estimation. For doing this comparison, we used two smooth test surfaces: a torus and the test surface used by Goldfeather and Interrante in [5]². Both surfaces cover a variety of different curvature configurations. They can be described in a closed parametric form, ensuring that the exact \mathbf{T} is available everywhere on the surface.

Most estimators give good results for rather regular triangulations, while irregular triangulations tend to drop the quality of the estimators significantly ([5]). In order to consider an interesting triangulation of the surface, we used two different triangulations for each of the two test surfaces. The first triangulation places the vertices on a regular grid in the domain while the triangulation of each grid cell is chosen randomly. This way a higher number of vertices with different valences is created - the resulting mesh contains vertices ranging from valence 4 to valence 8. The second triangulation does a perturbation of the grid point in the domain before a Delaunay triangulation is applied. This way we obtain an irregular triangulation where the majority of the vertices has a valence 6. Figure 5 illustrates the test triangulations considered in this paper. The two torus meshes consist of 20.000 triangles while the two Goldfeather meshes have 10.000 triangles.

Obviously, the quality of our new estimation method of **T** strongly depends on the quality of the available normals. If the underlying surface is known in an implicit form, the "perfect" normals (both in direction and length) are available by considering the gradient. In case of an available parametric description of the underlying surface, the exact normal direction can be obtained as well. However, in most cases they have to be estimated. To do so, a number of approaches exist. For instance, [7] estimates normals to be as possibly accurate for cubic surfaces while [8] approximate the mean curvature normal. For our computations we used the simplest weighting triangle normals by the triangle areas, which already provides good results.

To find a fair comparison between our new estimation and pre-existing approaches, we carefully have to choose an appropriate setup because of the different nature of the estimators: other methods yield discrete estimations of T in the vertices, while our method gives continuous functions of Tinside each triangle (including discontinuities at the boundaries and the vertices). To make both approaches compara-

¹ We used Maple for doing the computations.

² To be precise, we only used the middle part of the Goldfeather surface, since the remaining two parts do not fit in a curvature continuous way.



Figure 5. The four test triangulations: a) torus 1; b) torus 2 ; c) Goldfeather 1; d) Goldfeather 2.

ble, we have to adapt our method to compute \mathbf{T} in every vertex of the mesh. Note that our method generally gives n different values of \mathbf{T} for a vertex with a valence of n: depending on which triangle we compute the estimation, the resulting \mathbf{T} usually differs. However, to get a unique \mathbf{T} for every vertex, we compute the (non-weighted) average of all estimations of \mathbf{T} in this vertex. Figure 6 illustrates this.



Figure 6. Estimating T at a vertex x_i : a) each triangle sharing x_i gives another T; b) principal directions of the averaged T.

Another problem to do the comparison is to find a distance metric of the estimated T at the vertices of the mesh. We decided not to compare the **T** themselves but to extract Gaussian curvature, mean curvature and principal directions, and to compute distance functions based on these measures. In particular, we compute the distances of the Gaussian curvature between two estimations³ as

dist_K =
$$\frac{1}{n} \sum_{i=1}^{n} (K_1(i) - K_2(i))^2$$

where *i* is running over the indices of all *n* vertices, and $K_1(i)$, $K_2(i)$ are estimations of *K* at the *i*-th vertex using the two methods to be compared. In a similar way we compute the distance in terms of mean curvature:

dist_H =
$$\frac{1}{n} \sum_{i=1}^{n} (H_1(i) - H_2(i))^2$$
.

To compute the distance of the principal directions in a vertex \mathbf{x}_i , we use the following approach: suppose one estimator gives the (normalized) principal directions \mathbf{k}_1 , \mathbf{k}_2 while another estimator (or the exact computation) gives the (normalized) principal directions \mathbf{p}_1 , \mathbf{p}_2 , we compute the distance of the principal directions at \mathbf{x}_i as

dist_P(i) = min{
$$\frac{1}{2}(\arccos \|\mathbf{k_1}\mathbf{p_1}\| + \arccos \|\mathbf{k_2}\mathbf{p_2}\|), \\ \frac{1}{2}(\arccos \|\mathbf{k_1}\mathbf{p_2}\| + \arccos \|\mathbf{k_2}\mathbf{p_1}\|)$$
}

which gives the average angle deviation between the corresponding directions. This gives for the global distance of the principal directions

$$\operatorname{dist}_P = \frac{1}{n} \sum_{i=1}^n \operatorname{dist}_P(i).$$

To evaluate our estimation, we compared it with three wellestablished estimation methods which can be considered as being among the most powerful methods which are currently available ([5]). In fact, we compare the following estimations with the exact curvature values:

- The cubic fitting ([5]) incorporating normals which are estimated by a weighted average of the triangle normals. In the following we call this estimation CF.
- The quadratic fitting ([5]), in the following called QF.
- The quadratic fitting ([15]) QT
- The normal based estimation using the exact (perfect) normals NP.
- The normal based estimation using estimated normals by a weighted average of the triangle normals NE.

For each of the four test meshes we considered 5 versions: the original one and four noisy meshes. To create the noise versions, we randomly perturbed each vertex in



normal direction. Figure 7 shows the original torus and the

most noisy one considered here. Figure 8 shows error diagrams for the torus 1 data set concerning Gaussian curvature (left), mean curvature (middle) and principal directions (right). The horizontal axes of the diagrams show the amount of noise we added to the data set - we considered { 0, 0.5, 1, 2, 3 } $\cdot 10^{-2}$ · length of bounding box diagonal. The vertical axes show the errors between the estimation and the exact values for Gaussian curvature, mean curvature and principal directions respectively. The results for the five different estimation techniques are shown in different colors. As we can see in figure 8, our normal based estimation outperforms the other techniques if the exact normals are available (NP), e.g. if an implicit surface is sampled. If the normals have to be estimated, our method (NE) yields similar results as the cubic fitting (CF). In fact, in most cases NE performs slightly better than CF. The two quadratic fitting methods QF and QT show a rather similar behavior revealing a larger error than NE and CF. This corresponds to the fact that NE and CF incorporate a 2-ring to estimate T while QT and QF work only on a 1-ring around the vertex.

Figure 9 shows the error plot for the torus 2 data set using the same setup as for figure 8. We obtain a similar result as for torus 1: NP generally performs best, CF and NE have a similar performance (where NE is slightly better), while QT and QF tend to produce the largest error.

Figure 12 shows a collection of Gaussian curvature plots for torus 2 with $0.5 \cdot 10^{-2}$ added noise. Figure 12a shows the exact curvature plot, figure 12b shows the result for the CF estimation. Figures 12c and 12d show the quadratic fittings QF and QT respectively. Figure 12e shows the result for NP where every triangle is considered independently of its neighbors. Figure 12f shows NP where the results are averaged per vertex. Figure 12g shows the results for NE (triangles considered independently) while figure 12h shows NE with averaged values per vertex. Figure 12 clearly shows that NP comes closest to the correct curvature plot, that CF and NE have a similar behavior, and that QF and QT introduce the largest errors. A similar statement holds for figure 13 which shows a collection of mean curvature plots for the same data set.

Figure 10 shows the error plots for the Goldfeather 1 data set with the same setup as in figures 8 and 9. Figure 11 does so for the Goldfeather 2 mesh. Both plots reveal a rather similar behavior as observed for the tori: NP performs best, NE and CF have similar errors, while QF and QT give the largest error for slightly noisy data. The same trend can be seen in the curvature plots of the Goldfeather 2 surface with $0.5 \cdot 10^{-2}$ added noise. Figure 14 shows the Gaussian curvature plots, figure 15 the mean curvature plots, while figure 16 visualizes the principal directions.



Figure 16. Principal direction for Goldfeather 2: a) exact; b) CF; c) QF; d) QT; e) NP (independent triangles); f) NP (averaged at vertex); g) NE (independent triangles); h) NE (averaged at vertex).

6. Results

In this section we apply our estimation to a rather complex data set. Figure 17 shows the triangular mesh which was obtained by a range scan of the wooden sculpture "Freezing Old Woman" (Frierende Alte, 1937) by the German expressionist sculptor Ernst Barlach (1870 – 1938). This model consists of approx. 1.5 million triangles. It took 2.88 seconds on a 3 GHz Pentium 4 computer to compute the curvature tensor at each vertex. Figure 18a shows the Mean curvature plot of a closeup showing a part of the nose

³ or between one estimation and the exact values



Figure 17. Ernst Barlach: Freezing Old Woman (appr. 1.5 million triangles).

and one eye. Figure 18b shows the principal directions of a magnification showing two fingertips of the sculpture.

7. Conclusions

We have introduced a new technique for estimating the curvature tensor T in a triangular mesh. This technique estimates T for a single triangle equipped with (exact or estimated) surface normals. The result is a continuous function for T inside each triangle. To make it comparable with preexisting methods working on vertices, we average all values of T obtained from all triangles sharing the vertex. The new estimation technique generally shows a slightly better error behavior than a cubic fitting ([5]). If the exact normal of the underling surfaces is available at the vertices, the error drops significantly. The new approach is independent of rotations of the mesh, and it does not incorporate any parametrization or fitting approaches.

There are a number of improvements of our approach which can be considered in future research. In the current version we only considered unit normals at the vertices of the mesh. Since the lengths of the normals influence the estimation, they are additional parameters for improving the quality of the estimation. If an estimation of T at the vertices is desired, we used a non-weighted average of the T obtained from the triangles sharing the vertex. Here a weighted average may improve the estimation as well. Since our approach heavily depends on the quality of the quality of the normals, another possible extension is to consider quadratically interpolated normals ([14]) instead of the linearly interpolated normals considered here.

8. Acknowledgements

The authors would like to thank Dr. Volker Probst from the Ernst Barlach foundation Güstrow (Germany) for permitting the scanning of the sculpture. The scan was carried out by Heiko Wanning and Hendrik Lensch.

References

- P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, July 2003.
- [2] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. pages 177–199.
- [3] D. Cohen-Steiner and J.-M. Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth Conference on Computational Geometry (SCG-03)*, pages 312–321, New York, June 8–10 2003. ACM Press.
- [4] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley Publishing Company, Reading Massachusetts, 1996.
- [5] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal directions vectors. ACM *Transactions on Graphics*, 23(1):45–63, 2004.
- [6] H. Hagen et al. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, 1992.
- [7] N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.
- [8] M. Meyer, M. Desbrun, M. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2manifolds. In *Proceedings of VisMath*, 2002.
- [9] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. In E. E. Catmull, editor, *Proceedings* of the 19th Annual ACM Conference on Computer Graphics and Interactive Techniques, pages 167–176, New York, NY, USA, July 1992. ACM Press.
- [10] D. L. Page, A. Koschan, Y. Sun, J. Paik, and A. Abidi. Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *Proceedings on Computer Vision and Pattern Recongition*, 2001.
- [11] S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, (2), 2001.
- [12] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission,* 2004.
- [13] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings International Conference on Computer Vision*, pages 902–907, 1995.
- [14] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN triangles. In *Proceedings of the Symposium on Interactive 3D graphics*, pages 159–166, 2001.
- [15] W. Welch and A. Witkin. Free–Form shape design using triangulated surfaces. In A. Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 247–256. ACM Press, 1994.



Figure 8. Error diagram torus 1: error (vertical axes) against amount of present noise (horizontal axes).



Figure 9. Error diagram torus 2: error (vertical axes) against amount of present noise (horizontal axes.



Figure 10. Error diagram Goldfeather 1: error (vertical axes) against amount of present noise (horizontal axes).



Figure 11. Error diagram Goldfeather 2: error (vertical axes) against amount of present noise (horizontal axes).



Figure 12. Gaussian curvature plots for torus 2: a) exact; b) CF; c) QF; d) QT; e) NP (independent triangles); f) NP (averaged at vertex); g) NE (independent triangles); h) NE (averaged at vertex).







Figure 14. Gaussian curvature for Goldfeather 2: a) exact; b) CF; c) QF; d) QT; e) NP (independent triangles); f) NP (averaged at vertex); g) NE (independent triangles); h) NE (averaged at vertex).



Figure 15. Mean curvature for Goldfeather 2: a) exact; b) CF; c) QF; d) QT; e) NP (independent triangles); f) NP (averaged at vertex); g) NE (independent triangles); h) NE (averaged at vertex).



Figure 18. Details of "Freezing Old Woman": a) Mean curvature; b) Gaussian curvature; c) principal directions.