Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 01-006

Understanding Errors in Approximating Principal Direction Vectors

Jack Goldfeather

February 01, 2001

Understanding errors in approximating principal direction vectors

By Jack Goldfeather

Introduction

Suppose we are given only a surface mesh of vertices and polygons approximating some unknown smooth surface. There have been many methods proposed for approximating principal directions of the underlying surface [1,2,3,4,5,6]. In this paper we will examine a few of the known methods, showing how well they can work for some surfaces and how miserably they can fail for others. In particular we will show that very tiny approximation errors can be magnified into large directional errors. We also introduce a new method that we believe performs significantly better than many other proposed methods. In section I, we will outline the basic mathematics behind computing principal directions, stating the necessary formulas for the Weingarten curvature matrix and for its use in computing normal curvature in a given direction. In section II we describe in detail three methods each of which approximates the Weingarten curvature matrix at a vertex of the mesh. In Section III we apply each method to some surfaces using a number of different mesh schemes and then examine the direction errors. In section IV we develop some mathematical relationships between surface approximation errors and errors in principal directions.

I. A quick review of surface curvature.

Let p be a point on a smooth surface S, N_p the unit normal to S at p, and suppose $X(u,v)$ is a local parametrization of S in a neighborhood of p. Then using $X_u(p)$, $X_v(p)$, N_p as a local coordinate system, we can compute principal curvatures and principal directions as follows: Let λ_1, λ_2 ($\lambda_1 \geq \lambda_2$) be the eigenvalues, and p_1, p_2 associated unit eigenvectors of the Weingarten curvature matrix

$$W = \begin{pmatrix} \frac{eG - fF}{EG - F^2} & \frac{fE - eF}{EG - F^2} \\ \frac{fG - gF}{EG - F^2} & \frac{gE - fF}{EG - F^2} \end{pmatrix}$$

where

$$\begin{aligned} e &= N_p \circ X_{uu}(p) & \qquad E &= X_u(p) \circ X_u(p) \\ f &= N_p \circ X_{uv}(p) & \qquad F &= X_u(p) \circ X_v(p) \\ g &= N_p \circ X_{vv}(p) & \qquad G &= X_v(p) \circ X_v(p) \end{aligned}$$

Note that in the special case that X_u and X_v are orthogonal unit vectors, this becomes the symmetric matrix

$$W = \begin{pmatrix} e & f \\ f & g \end{pmatrix}$$

If u is a unit vector in the tangent plane to S at p, then

$$\kappa_u = u^T W u$$

is the **normal curvature** of the surface in the direction of u.

It follows that λ_1 and λ_2 are the maximum and minimum normal curvatures of the surface at p, and $p_1 = \begin{pmatrix} p_{11} \\ p_{12} \end{pmatrix}$ and $p_2 = \begin{pmatrix} p_{21} \\ p_{22} \end{pmatrix}$ are the principal curvature vectors expressed in local coordinates. That is,

$$v_1 = p_{11}X_u + p_{12}X_v$$

$$v_2 = p_{21}X_u + p_{22}X_v$$

are the principal direction vectors in R^3.

II. Three principal direction approximation methods

The first step in computing principal directions on a surface mesh is to compute at each vertex p a vector N'_p that approximates the true unit surface normal N_p at p. Most methods compute a "normalized average", i.e. a set of vectors is summed and the resulting vector is normalized to length 1. Here are a few methods that have been used:

1. The normalized average of the surrounding triangle unit normals.
2. The normalized average of the surrounding triangle unit normals, each weighted by the angle of the triangle at p.
3. The normalized average of the adjacent edges, each weighted by the sum of the cotangents of the angles opposite this edge in the two triangles sharing the edge [1].

We tried each of these schemes on a complex surface (S_2 in the next section), using a variety of local meshes and a large number of random points on the surface. The result was that they all worked very well on meshes that were not too skewed (i.e. no triangle "slivers", no one triangle significantly larger than the others, etc.), and method 2 worked marginally better than method 2 and much better than method 3 in the more pathological circumstances. For this reason, we chose method 2 as our approximation method.

The next step is to choose orthonormal vectors x_1 and x_2 in the plane tangent to N'_p to form a local orthonormal coordinate system $L = \{x_1, x_2, N'_p\}$ in R^3. All calculations are done with respect to this local coordinate system. Each of the methods outlined below approximates the Weingarten curvature matrix expressed in this local coordinate system.

The Normal Curvature Method

Suppose W is the unknown Weingarten matrix with respect to L at vertex p. Suppose there are n vertices adjacent to p and let q_i denote the i^{th} adjacent vertex. We denote by y_i the unit vector obtained by projecting the vector $p\bar{q}_i$ (expressed in local coordinates L) onto the plane tangent to N'_p and normalizing the result. Then using the result from section I, the normal curvature in the direction y_i is given by $\kappa_{y_i} = y_i^T W y_i$. An approximation to this normal curvature is given by:

$$\kappa'_{y_i} = 2\frac{(p - q_i) \circ N'_p}{(p - q_i) \circ (p - q_i)}$$

which is the curvature of the unique osculating circle passing through p and q_i with normal N'_p at p. This produces a system of equation:

$$y_i^T W y_i = \kappa'_{y_i} \qquad i = 1, 2, \ldots, n \tag{1}$$

that we wish to solve for W. In [1], these equations are weighted in the same manner as the weighting in the third normal approximation method mentioned at the beginning of this section, but we do not believe that this improves the result much. The fact is that the osculating circle only produces a second-order approximation to the true normal curvature and second-order approximations will introduce significant error in many cases.

The first step towards a solution is to reorganize (1) as follows. Let $y_i = (u_i, v_i)$ and

$$W = \begin{pmatrix} A & B \\ B & C \end{pmatrix}.$$

Then

$$y_i^T W y_i = \begin{pmatrix} u_i & v_i \end{pmatrix} \begin{pmatrix} A & B \\ B & C \end{pmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_i^2 & 2u_iv_i & v_i^2 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix}$$

If we let U be the $n \times 3$ matrix with rows $\begin{pmatrix} u_i^2 & 2u_iv_i & v_i^2 \end{pmatrix}$, $X = \begin{pmatrix} A & B & C \end{pmatrix}^T$, and d' be the n-vector whose i^{th} entry is κ'_{y_i} the entire system can be written as the matrix equation

$$UX = d'. \tag{2}$$

If d' was the vector of true curvatures, this linear system would have an exact solution (assuming the adjacent vertices do not have some degenerate pattern, like all lying on the same line.) In practice, the best we can hope to find is a least squares fit, i.e., a vector X' that minimizes $|UX' - d'|$. There are standard numerical methods for finding this least squares solution. The resulting matrix

$$W' = \begin{pmatrix} A' & B' \\ B' & C' \end{pmatrix}$$

is used to approximate the principal directions.

The Quadratic Surface Method

In this method we try to best-fit a quadratic surface to the adjacent vertices. We begin by transforming q_i to local coordinates (x_i, y_i, z_i). In these local coordinates, p becomes $(0, 0, 0)$, N'_p lies along the positive z-axis, and the quadratic surface looks like

$$z = f(x, y) = \frac{A}{2}x^2 + Bxy + \frac{C}{2}y^2$$

The Weingarten matrix for such a surface is

$$W = \begin{pmatrix} A & B \\ B & C \end{pmatrix}.$$

As in the Normal Curvature method, we plug in the adjacent vertices to get a system of equations

$$UX = Z$$

where the i^{th} row of U is $(\frac{1}{2}x_i^2 \quad x_i y_i \quad \frac{1}{2}y_i^2)$ and the i^{th} entry in Z is z_i. Again we look for a least-squares fit to the system.

At first glance, the Normal Curvature and Quadratic methods seem like they are pretty much the same since they both use a kind of second-order approximation. Indeed, this is their big failing, since we expect that there might be significant error for surfaces that cannot be approximated well by second-order data. However, they don't fail in quite the same way as we will see.

The Adjacent-Normal Cubic Method

In both the proceeding methods, we did not use all of the information available to us, namely, we did not use the known (approximated) normal vectors at q_i. We can use this information to create a third-order approximation method that we believe has not yet appeared in the literature. As we will see in the next section, this method seems to be significantly better than the first two.

As in the Quadratic method we try to fit a surface to the adjacent vertex data. Let

$$f(x, y) = \frac{A}{2}x^2 + Bxy + \frac{C}{2}y^2 + Dx^3 + Ex^2y + Fxy^2 + Gy^3.$$

The Weingarten matrix for this surface is still $W = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$, because in the local coordinate system the curvature only depends on the second degree terms. However, using the third degree terms in the least-squares fit will produce values for (A, B, C) different from the ones we get in the quadratic method. The normal to this surface is given by:

$$N(x, y) = (f_x(x, y), f_y(x, y), -1)$$

$$= (Ax + By + 3Dx^2 + 2Exy + Fy^2, Bx + Cy + Ex^2 + 2Fxy + 3Gy^2, -1).$$

Let (a_i, b_i, c_i) denote the normal at the data point (x_i, y_i, z_i) (both normal and point must be transformed to the local coordinates), and let $X = (A \quad B \quad C \quad D \quad E \quad F \quad G)^T$. Rewrite the normal as $(-\frac{a_i}{c_i}, -\frac{b_i}{c_i}, -1)$. For each point, we have an equation

$$(\tfrac{1}{2}x_i^2 \quad x_i y_i \quad \tfrac{1}{2}y_i^2 \quad x_i^3 \quad x_i^2 y_i \quad x_i y_i^2 \quad y_i^3) X = z_i$$

and for each normal we have two equations:

$$(x_i \quad y_i \quad 0 \quad 3x_i^2 \quad 2x_i y_i \quad y_i^2 \quad 0) X = -\frac{a_i}{c_i}$$

$$(0 \quad x_i \quad y_i \quad 0 \quad x_i^2 \quad 2x_iy_i \quad 3y_i^2)\, X = -\frac{b_i}{c_i}$$

We let U be the $3n \times 7$ matrix and Z be the $3n$-vector with entries at rows $3i$, $3i+1$, and $3i+2$ the left and right sides of the above three equations, respectively. As in the previous methods we find a least squares fit to

$$UX = Z$$

and use only A, B, and C from the result.

III. Testing the methods

We wanted to set up some fair comparison tests. We chose two different surfaces, one of which, $S_1(u,v)$, is simply a torus, and one of which, $S_2(u,v)$, is defined by:

$$f(u) = -2u^4 + 2u^2 + u/6 + 0.3$$

$$g(u) = \begin{cases} .9 + u + f(-.9), & -.9 - f(-.9) \le u \le -.9; \\ f(u), & -.9 \le u \le 1; \\ 1 - u - f(1), & 1 \le u \le 1 + f(1); \end{cases}$$

$$h(u) = \begin{cases} -.9, & -.9 - f(-.9) \le u \le -.9; \\ u, & -.9 \le u \le 1; \\ 1, & 1 \le u \le 1 + f(1); \end{cases}$$

$$x(u,v) = g(u)\cos(v)$$

$$y(u,v) = g(u)\sin(v)$$

$$z(u,v) = h(u) + 0.2\sin(2x(u,v)) + 0.15\cos(3x(u,v)y(u,v))$$

$$S_2(u,v) = (x(u,v), y(u,v), z(u,v)) \qquad .9 - f(-.9) \le u \le 1 + f(1), \qquad 0 \le v \le 2\pi.$$

Part of the complexity of this surface is to ensure that it is closed, and part is to create significant areas where the curvature is "interesting".

We also chose two different mesh schemes. Mesh scheme M_1 was created by starting with the rectangular parameter space $u1 \le u \le u2$, $v1 \le v \le v2$, letting $du = (u2 - u1)/50$ and $dv = (v2 - v1)/50$ and subdividing into subrectangles of size du by dv. Each subrectangle was then divided into two triangles by adding the diagonal from lower left to upper right. M_2 was created by starting with M_1 and moving each interior point (u_i, v_j) by a random amount (r_i, r_j) where $-du/3 \le r_i \le du/3$ and $-dv/3 \le r_j \le dv/3$.

The figures in Appendix 2 show the results for (S_i, M_i), $i = 1, 2$. The red lines indicate that the approximated principal direction are within 5° of the correct direction and the blue lines indicate where the direction is off by more than 5°.

We observe that all methods work very well for (S_1, M_1). Differences begin to appear for (S_2, M_1) and are quite apparent with (S_1, M_2) and (S_2, M_2). It seems that randomness can introduce serious errors. This led us to ask two simple questions:

(1) How do errors created by using second or third order approximations affect the Weingarten approximation?
(2) How do errors in the Weingarten approximation affect the eigenvector directions?

Before we consider these questions, we try to cast a bit of light on the situation. M_1 and M_2 are global mesh schemes for the entire surface. We wanted to try different mesh schemes to see if the kind of mesh made a difference. Since it is difficult to do this globally, we opted for creating a variety of local meshes around a given point (u, v) in parameter space. We chose a series of 100 random points (u_i, v_i) in parameter space and created 5 different local meshes around each one as follows:

(LM_1) The same as the regular mesh M_1, This has 6 adjacent vertices.
(LM_2) Start with the 4 subrectangles around (u_i, v_i) in M_1, and draw the 4 diagonals from (u_i, v_i) This has 8 adjacent vertices.
(LM_3) Start with 6 equally spaced points (s_j, t_j) on a circle centered at (u_i, v_i) with radius $r = min(du, dv)$. If $S(u, v)$ is the surface, we let $l = ||S(s_1, t_1) - S(u_i, v_i)||$ and **calibrate** the other vertices (s_j, t_j), $j = 2, \ldots, 6$ by moving them to $(\bar{s}_j, \bar{t}_j)$ along radial lines until $||S(\bar{s}_j, \bar{t}_j) - S(u_i, v_i)|| = l$. The idea is to create a mesh on the surface that is not biased in any direction.
(LM_4) The same as LM_3 except we use 8 points.
(LM_5) The same as LM_3 except we use 5 points.

We then took each of the local meshes and rotated them in 5 degree increments. After each rotation, we recalibrated LM_3, LM_4, and LM_5. Each of these rotated meshes was used for each of our principal direction methods. We were looking to see whether rotation of the mesh or the mesh type created the bigger errors. The results for surface S_2 are summarized in the tables in Appendix 1. In the tables, N, Q, and C are the Normal Curvature, Quadratic and Cubic methods, respectively, the first 3 columns show the average angle error and the last 9 columns show the number (out of 100) of random points where the error fell in the indicated range. We used the exact normal at each vertex, rather than an approximation, to give each method the best chance of working well.

We observe that the cubic method definitely is superior to the other two. We also note that the worst errors in all methods occurred for certain rotations using mesh LM_1. This is probably due to the fact that in some places, significant skewing took place when the mesh in parameter space was mapped onto the surface.

IV. A unified approach to error analysis

The previous section illuminates how well or badly our three methods can work. In this section we develop a unified approach to error analysis of all three methods. From a mathematical point of view, all three of the methods outlined in the previous section, have the same two steps:

(1) Find a least-squares fit to a linear system

$$Ux = b'$$

where b' is computed from data points in the surface mesh.

(2) Use entries in the least-squares solution x' to form a symmetric matrix

$$W' = \begin{pmatrix} A' & B' \\ B' & C' \end{pmatrix}.$$

and then find the eigenvalues and eigenvectors of W'.

The question we seek to answer is:

How do errors in b' affect errors in the principal directions?

In order to understand the subtlety of this question, consider the following example, taken directly from one of the test cases in the previous section. The matrix U computed using the Normal Curvature method is given by:

$$U = \begin{pmatrix} 0.119319 & -0.648328 & 0.880681 \\ 0.194421 & 0.791509 & 0.805579 \\ 0.459432 & 0.996703 & 0.540568 \\ 0.047944 & 0.427295 & 0.952056 \\ 0.136932 & .68755 & 0.863068 \\ 0.208186 & .812021 & 0.791814 \end{pmatrix}$$

the true principal curvatures are 1.84 and 0.578, and the true normal curvature vector is:

$$b = (1.31872, 1.82591, 1.6508, 1.47723, 1.83995, 1.82079).$$

Suppose d is an error vector of length 0.5, and we let $b' = b - d$. How this error will effect the principal directions depends on how we distribute it among the 6 components. For example, each of the approximations

$$b_1 = (1.49893, 1.70951, 2.03881, 1.43369, 1.63304, 1.72745)$$

$$b_2 = (1.03218, 1.93208, 1.91872, 1.21649, 1.89610, 1.93783)$$

$$b_3 = (0.97174, 2.00138, 1.64589, 1.29353, 2.02988, 1.99099)$$

satisfies $||b - b_i|| = 0.5$. Note also that we are not at an umbilical point (i.e. where the minimum and maximum curvatures are equal) where we might expect problems to occur. Which one produces the least principal direction angle error and which one produces the most? The surprising answer is:

b_1	Error of 66 degrees.
b_2	Error of 13.6 degrees.
b_3	Error of 0 degrees.

We will show in what follows how there could be such a disparity in the result. Suppose x is the exact solution to $Ux = b$. Suppose b' is an approximation to b and x' is the least-squares solution to $Ux = b'$. We call $d = b' - b$ the **input error** and $e = x' - x$ the **output error**. Consider the special case when $Ux' = b'$ so that $Ue = d$. The following theorem shows that the magnitude of e can be much larger than the magnitude of d.

Theorem 1 If $Ue = d$, then

$$\frac{1}{\sqrt{\lambda_{max}}}||d|| \leq ||e|| \leq \frac{1}{\sqrt{\lambda_{min}}}||d||$$

where λ_{min} and λ_{max} are the smallest and largest eigenvalues of U^TU.

Proof let $V = U^TU$ and note that V is symmetric and positive-definite so it has eigenvalues $0 < \lambda_1 \leq \ldots \leq \lambda_n$ with associated orthonormal eigenvectors $p_1, \ldots, p_n$. Then

$$||d||^2 = d^Td = e^TU^TUe = e^TVe.$$

Write $e = c_1p_1 + \ldots + c_np_n$ so that $||e||^2 = c_1^2 + \ldots + c_n^2$. Then

$$e^TVe = (c_1p_1 + \ldots + c_np_n)^TV(c_1p_1 + \ldots + c_np_n) =$$

$$(c_1p_1 + \ldots + c_np_n)^T(\lambda_1c_1p_1 + \ldots + \lambda_nc_np_n) = \lambda_1c_1^2 + \ldots \lambda_nc_n^2.$$

Hence

$$\lambda_1||e||^2 \leq ||d||^2 \leq \lambda_n||e||^2$$

from which the result follows easily.

In our example, $\frac{1}{\sqrt{\lambda_{min}}} = 3.83$ so we see that this scale factor may not be insignificant. In general, if e is a solution to $Ux = d$, we call $s_1 = ||e||/||d||$ the **first scale factor**.

Theorem 1 tells us that the magnitude of the output error can be greater than the magnitude of the input error. The next theorem tells us that matters can still get much much worse. We use e to construct an approximate Weingarten matrix. This error vector can create serious problems both because of its length and its direction.

Theorem 2 Suppose the Weingarten matrix W has eigenvalues λ_1, λ_2 with associated orthonormal eigenvectors p_1 and p_2 and $W' = W + \begin{pmatrix} e_a & e_b \\ e_b & e_c \end{pmatrix}$ has eigenvalues τ_1 amd τ_2 with associated orthonormal eigenvectors q_1 and q_2. Let $e = (e_a, e_b, e_c)$, α denote the angle between p_1 and the first local coordinate vector, and β denote the angle between $v = \begin{pmatrix} 2e_b \\ e_c - e_a \end{pmatrix}$ and $r_1 = \begin{pmatrix} \cos(2\alpha) \\ \sin(2\alpha) \end{pmatrix}$. Then the angle error θ between p_1 and the approximated eigenvector q_1 satisfies

$$\tan(2\theta) = \frac{||v|| \cos\beta}{||v|| \sin\beta + \lambda_1 - \lambda_2}.$$

In particular, the angle error will be largest when $\beta = 0$ and smallest when $\beta = \frac{\pi}{2}$

Proof Let P be the matrix with p_1 and p_2 as its columns and Q be the matrix with q_1 and q_2 as its columns. We begin the proof by noting that

(a) $p_1^Tp_1 = p_2^Tp_2 = 1$.

(b) $p_1^Tp_2 = p_2^Tp_1 = 0$.

(c) $W = P \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} P^T = \lambda_1 p_1 p_1^T + \lambda_2 p_2 p_2^T$.

(d) $W' = Q \begin{pmatrix} \tau_1 & 0 \\ 0 & \tau_2 \end{pmatrix} Q^T = \tau_1 q_1 q_1^T + \tau_2 q_2 q_2^T$.

Then from (c) and (d), $W' = W + E$ implies

$$\tau_1 q_1 q_1^T + \tau_2 q_2 q_2^T = \lambda_1 p_1 p_1^T + \lambda_2 p_2 p_2^T + E \tag{1}$$

Now, let θ be the angle between p_1 and q_1. Then

$$p_1^T q_1 = \cos\theta$$

$$p_1^T q_2 = \cos(\theta + \frac{\pi}{2}) = -\sin\theta$$

$$p_2^T q_1 = \cos(\frac{\pi}{2} - \theta) = \sin\theta$$

$$p_2^T q_2 = \cos\theta$$

If we multiply (1) on the left by p_1^T and on the right by p_1 we obtain:

$$\tau_1 \cos^2\theta + \tau_2 \sin^2\theta = \lambda_1 + p_1^T E p_1. \tag{2}$$

Similiarly, if we multiply (1) on the left by p_2^T and on the right by p_2, or on the left by p_1^T and on the right by p_2 we obtain, respectively:

$$\tau_1 \sin^2\theta + \tau_2 \cos^2\theta = \lambda_2 + p_2^T E p_2. \tag{3}$$

$$\tau_1 \sin\theta\cos\theta - \tau_2 \sin\theta\cos\theta = p_1^T E p_2. \tag{4}$$

Subtracting equation (3) from equation (2), and multiplying equation (4) by 2, we obtain, respectively:

$$(\tau_1 - \tau_2)\cos(2\theta) = p_1^T E p_1 - p_2^T E p_2 + \lambda_1 - \lambda_2. \tag{5}$$

$$(\tau_1 - \tau_2)\sin(2\theta) = 2 p_1^T E p_2. \tag{6}$$

Finally, dividing equation (6) by equation (5) we get the result:

$$\tan(2\theta) = \frac{2 p_1^T E p_2}{p_1^T E p_1 - p_2^T E p_2 + \lambda_1 - \lambda_2}. \tag{7}$$

We can rewrite this expression in order to make it easier to understand. Observe that since p_1 and p_2 are orthonormal they can be written in the form:

$$p_1 = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \qquad p_2 = \begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}$$

where α is the angle between the eigenvectors of A and the local coordinate vectors. Plugging this into equation (7) we obtain:

$$\tan(2\theta) = \frac{(e_c - e_a)\sin(2\alpha) + 2e_b\cos(2\alpha)}{2e_b\sin(2\alpha) - (e_c - e_a)\cos(2\alpha) + \lambda_1 - \lambda_2}. \tag{8}$$

Now let

$$r_1 = \begin{pmatrix} \cos(2\alpha) \\ \sin(2\alpha) \end{pmatrix} \qquad r_2 = \begin{pmatrix} \sin(2\alpha) \\ -\cos(2\alpha) \end{pmatrix} \qquad v = \begin{pmatrix} 2e_b \\ e_c - e_a \end{pmatrix}$$

Then we can rewrite (8) as

$$\tan(2\theta) = \frac{v^T r_1}{v^T r_2 + \lambda_1 - \lambda_2} \tag{9}$$

If we let β be the angle between v and r_1 we get

$$\tan(2\theta) = \frac{||v|| \cos\beta}{||v|| \sin\beta + \lambda_1 - \lambda_2} \tag{10}$$

Then the error will be largest when $\beta = 0$ and smallest when $\beta = \frac{\pi}{2}$. This completes the proof of Theorem 2.

Note that if $B = \begin{pmatrix} 0 & 2 & 0 \\ -1 & 0 & 1 \end{pmatrix}$

$$v = \begin{pmatrix} 2e_b \\ e_c - e_a \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 \\ -1 & 0 & 1 \end{pmatrix} e = Be.$$

Also note that

$$B^T B = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

has largest eigenvalue $\lambda = 4$, with associated eigenvector $(0, 1, 0)^T$. It is not hard to show in a manner similar to Theorem 1 that $||e||$ can be scaled by as much as $\sqrt{\lambda} = 2$ when multiplied by B.

Theorem 2 tells us that the principal direction error is caused by both the magnitude of v and its direction with respect to r_1. We can think of the magnitude of v as a product of two different scalings, s_1 and s_2, of the original input error d, first by multiplying d by U to get e, and then multiplying e by B to get v. The first scale factor can be as large as $1/\sqrt{\lambda_{min}}$ if d's direction is the same as the direction of p_{min} (Recall that λ_{min} is the smallest eigenvalue of U^TU with associated eigenvector p_{min}). The second scale factor can be as much as 2, if e is in the direction of $(0,1,0)^T$.

We used Theorem 1 and 2 to construct our three approximate curvature vectors. First note that the least-squares solution to $Ue = d$ is an exact solution to $U^TUe = U^Td$. We found d_1 by finding a solution to $U^Td = p_{min}$ and then scaled it down to have a length of 0.5. Then we let $b_1 = b - d_1$. This guaranteed that we would find e with length as large as possible.

We found d_2 and d_3 as follows. We know that $e = (U^TU)^{-1}U^Td$ so that $Be = B(U^TU)^{-1}U^Td$. We solved $B(U^TU)^{-1}U^Td = r_1$ to find d_2 and $B(U^TU)^{-1}U^Td = r_2$ to find d_3. We then scaled them both to have length 0.5 and let $b_2 = b - d_2$ and $b_3 = b - d_3$.

The following table summarizes the calculation for each of the three approximations. In the second to last column we show what the error would have been only due to β, i.e. we set $s = 1$.

Approx	s_1	s_2	$s = s_1s_2$	β	Error if s=1	Error
b_1	3.83	0.585	2.24	66.8°	24°	66.0°
b_2	0.78	0.835	0.65	0°	19.2°	13.6°
b_3	2.09	0.62	1.3	90°	0°	0°

We have shown that small errors in computing such quantities as normal curvature can significantly amplify the resulting angle error in principal direction. In particular, the problem is a combination of both the magnitude of the errors we make and the direction in which we make them. The former can be controlled to some extent. For example, the cubic method does a better job of controlling error magnitude than the others because it uses third-order approximations rather than second-order ones. The direction problem seems to be related to the local geometry of the mesh. It seems possible that some constraint on the mesh design might minimize rotation errors.

REFERENCES

1. Mathieu Desbrun, Mark Meyer, Peter Schroder,and Alan H. Barr, *Discrete Differential Geometry Operators in nD*, preprint, July 22, 2000.
2. Patrick J. Flynn and Anil K. Jahn, On Reliable Curvature Estimation, IEEE Computer Vision and Pattern Recognition, pp. 110-116, 1989.
3. Xin Chen and Francis Schmitt, *Intrinsic Surface Properties from Surface Triangulation*, Proceedings, European Conference on Computer Vision, May 1992.
4. Gabriel Taubin, *Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation*, Proc. 5th Intl Conf. on Computer Vision (ICCV'95), pp. 902-907,June 1995.
5. Phillippe Samson and Jean-Laurent Mallet, *Curvature Analysis of Triangulated Surfaces in Structural Geology*, Mathematical Geology, Vol. 29, No. 3, pp. 391-412, 1997.
6. Bernd Hamann, *Curvature Approximation of 3D manifolds in 4D space*, Computer Aided Geometric Design, 11, pp. 621-632, 1994.

APPENDIX 1

LM1	Ave. degree Err			0-5 degrees			5-10 degrees			> 10 degrees		
Rot	N	Q	C	N	Q	C	N	Q	C	N	Q	C
0	3.8	5.3	4.4	87	82	90	3	8	2	10	10	8
1	4.4	3.5	4.6	85	88	89	4	3	2	11	9	9
2	4.6	2.7	3.7	87	89	89	2	2	2	11	9	9
3	4.8	3.4	3.7	87	89	90	0	1	2	13	10	8
4	4.5	4.7	3.7	88	86	89	2	2	3	10	12	8
5	4.8	5.3	4.1	85	85	89	7	4	3	8	11	8
6	5.3	7.1	5.7	79	78	85	12	7	2	9	15	13
7	7.7	9.1	7.2	70	72	83	13	9	2	17	19	15
8	10.0	10.4	7.5	66	65	81	10	10	3	24	25	16
9	11.6	12.5	6.6	59	57	79	13	15	6	28	28	15
10	14.2	14.8	8.4	51	45	78	13	23	4	36	32	18
11	16.3	17.8	9.3	48	38	76	12	18	6	40	44	18
12	20.1	21.6	10.2	45	28	73	10	21	5	45	51	22
13	23.3	25.4	11.1	40	22	71	8	24	5	52	54	24
14	25.0	26.9	11.4	34	21	69	8	19	6	58	60	25
15	25.9	29.1	11.7	26	21	69	21	20	6	53	59	25
16	25.2	30.0	11.6	25	22	68	18	15	7	57	63	25
17	23.9	30.6	11.9	17	27	69	23	15	5	60	58	26

LM2	Ave. degree Err			0-5 degrees			5-10 degrees			> 10 degrees		
Rot	N	Q	C	N	Q	C	N	Q	C	N	Q	C
0	3.2	7.2	4.1	92	74	90	0	11	1	8	15	9
1	4.9	8.8	7.3	87	71	84	2	11	1	11	18	15
2	6.3	10.4	6.8	84	65	82	2	11	4	14	24	14
3	8.0	13.4	7.7	82	46	78	2	18	5	16	36	17
4	9.0	14.9	8.0	81	47	78	1	13	4	18	40	18
5	8.7	15.9	10.5	78	39	73	6	16	5	16	45	22
6	9.1	18.0	10.0	73	31	70	8	16	5	19	53	25
7	9.9	20.4	10.5	67	21	70	10	20	6	23	59	24
8	11.0	22.3	10.7	68	17	69	8	17	6	24	66	25
9	11.2	24.5	10.7	65	21	70	9	13	4	26	66	26
10	12.9	26.4	10.7	60	16	68	11	14	8	29	70	24
11	14.9	29.9	10.9	54	15	66	15	16	9	31	69	25
12	15.8	28.8	11.8	48	23	66	18	17	7	34	60	27
13	17.4	28.1	11.4	45	29	67	16	12	6	39	59	27
14	18.6	35.2	11.1	47	22	67	11	17	7	42	61	26
15	18.6	35.3	11.5	45	26	67	13	13	6	42	61	27
16	17.0	33.4	11.3	46	27	67	16	13	6	38	60	27
17	13.9	34.5	11.0	41	24	67	31	13	6	28	63	27

LM3	Ave. degree Err			0-5 degrees			5-10 degrees			> 10 degrees		
Rot	N	Q	C	N	Q	C	N	Q	C	N	Q	C
0	6.6	6.3	2.0	84	82	95	4	7	1	12	11	4
1	6.1	6.4	2.0	85	83	95	3	5	0	12	12	5
2	5.4	6.1	2.0	90	87	95	2	4	1	8	9	4
3	4.6	4.2	2.0	90	87	96	2	5	1	8	8	3
4	3.1	3.8	1.5	91	86	96	3	7	0	6	7	4
5	1.3	3.0	0.8	96	89	98	0	3	0	4	8	2
6	1.5	1.6	0.3	96	95	98	1	3	1	3	2	1
7	1.6	2.5	0.5	92	89	97	3	7	1	5	4	2
8	2.5	3.8	1.1	91	87	97	1	4	0	8	9	3
9	3.3	3.5	2.1	91	86	95	0	5	1	9	9	4
10	4.6	3.5	2.1	88	90	95	2	2	0	10	8	5
11	6.3	6.7	2.1	85	81	95	2	6	1	13	13	4
12	6.6	6.3	2.0	84	82	95	4	7	1	12	11	4
13	6.1	6.4	2.0	85	83	95	3	5	0	12	12	5
14	5.4	6.1	2.0	90	87	95	2	4	1	8	9	4
15	4.6	4.2	2.0	90	87	96	2	5	1	8	8	3
16	3.1	3.8	1.5	91	86	96	3	7	0	6	7	4
17	1.3	3.0	0.8	96	89	98	0	3	0	4	8	2

LM4	Ave. degree Err			0-5 degrees			5-10 degrees			> 10 degrees		
Rot	N	Q	C	N	Q	C	N	Q	C	N	Q	C
0	0.9	1.2	0.3	94	96	98	3	1	1	3	3	1
1	2.0	1.3	0.6	92	95	97	0	2	1	8	3	2
2	1.8	2.8	1.2	94	91	97	1	4	0	5	5	3
3	1.9	3.0	2.0	93	89	95	1	3	1	6	8	4
4	3.7	4.2	2.1	89	89	95	2	1	0	9	10	5
5	4.6	5.0	2.0	91	87	95	0	5	1	9	8	4
6	3.5	2.5	2.0	92	93	95	1	3	2	7	4	3
7	2.4	2.5	1.6	94	92	95	0	3	1	6	5	4
8	1.5	1.7	0.7	94	95	97	1	1	2	5	4	1
9	0.9	1.2	0.3	94	96	98	3	1	1	3	3	1
10	2.0	1.3	0.6	92	95	97	0	2	1	8	3	2
11	1.8	2.8	1.2	94	91	97	1	4	0	5	5	3
12	2.0	3.0	2.0	92	89	95	2	3	1	6	8	4
13	3.7	4.2	2.1	89	89	95	2	1	0	9	10	5
14	4.6	5.0	2.0	91	87	95	0	5	1	9	8	4
15	3.5	2.5	2.0	92	93	95	1	3	2	7	4	3
16	2.4	2.5	1.6	94	92	95	0	3	1	6	5	4
17	1.5	1.7	0.7	94	95	97	1	1	2	5	4	1

LM5	Ave. degree Err			0-5 degrees			5-10 degrees			> 10 degrees		
Rot	N	Q	C	N	Q	C	N	Q	C	N	Q	C
0	8.8	8.1	1.9	68	63	95	12	21	1	20	16	4
1	8.4	8.0	1.5	72	66	96	9	13	1	19	21	3
2	7.1	6.5	1.5	65	66	96	18	14	1	17	20	3
3	6.7	5.9	0.6	66	67	98	19	17	0	15	16	2
4	6.5	6.8	0.7	66	66	97	17	14	1	17	20	2
5	7.4	7.6	1.2	66	62	97	16	20	0	18	18	3
6	8.5	8.2	1.9	67	67	95	13	14	0	20	19	5
7	9.4	7.4	2.0	68	73	95	10	13	1	22	14	4
8	10.3	9.5	2.2	67	73	95	11	6	0	22	21	5
9	10.3	9.6	2.1	68	70	95	13	10	1	19	20	4
10	9.3	9.2	2.0	69	68	95	10	11	1	21	21	4
11	9.0	9.1	2.1	72	69	94	7	9	2	21	22	4
12	8.3	7.7	2.0	77	75	95	7	12	1	16	13	4
13	7.6	7.3	2.0	80	79	95	7	8	1	13	13	4
14	5.9	6.8	2.0	86	79	95	4	9	0	10	12	5
15	4.2	4.9	2.0	89	84	95	3	7	1	8	9	4
16	2.9	3.8	0.9	89	89	96	3	3	2	8	8	2
17	2.6	3.1	0.6	92	92	98	3	4	1	5	4	1