Estimating Gaussian Curvatures from 3D Meshes

Jingliang Peng[a], Qing Li[a], C.-C. Jay Kuo[a], Manli Zhou[b]

[a]University of Southern California, Los Angeles, CA 90089-2564, USA
[b]Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

ABSTRACT

A new approach to estimate the surface curvatures from 3D triangular mesh surfaces with Gaussian curvature's geometry interpretation is proposed in this work. Unlike previous work, the proposed method does not use local surface fitting, partial derivative computation, or oriented normal vector recovery. Instead, the Gaussian curvature is estimated at a vertex as the area of its small neighborhood under the Gaussian map divided by the area of that neighborhood. The proposed approach can handle vertices with the zero Gaussian curvature uniformly without localizing them as a separate process. The performance is further improved with the local Bezier curve approximation and subdivision. The effectiveness of the proposed approach for meshes with a large range of coarseness is demonstrated by experiments. The application of the proposed method to 3D surface segmentation and 3D mesh feature extraction is also discussed.

Keywords: Gaussian curvature, differential geometry, 3D mesh, 3D surface segmentation, 3D feature extraction

1. INTRODUCTION

The surface curvature gives a unique and viewpoint-independent description of a local shape.[1] The surface curvature property has been successfully employed in computer vision to carry out various tasks, ranging from image segmentation and feature extraction to scene analysis and object recognition. In this work, we are primarily interested in estimating Gaussian curvatures from 3D triangular meshes.

Several techniques have been developed to estimate the curvature information in the last decade. We can classify existing methods of curvature estimation from several different angles. From the mathematical viewpoint, the curvature information can be retrieved by the first and second partial derivatives of the local surface, the local surface normal or the tensor voting. From the computational viewpoint, there are analytic and numerical mehods (to be explained below). As to the input data type, the curvature information can be estimated from the range data, the intensity data[1, 14] or 3D sparse data.

Flynn and Jain[2] evaluated five curvature estimation methods and classified them into two categories: analytic methods[3–6] and numerical methods.[7–9] Analytic methods fit a surface to range values in a local patch and determine the curvature information using the first and the second partial derivatives computed from the surface equation. Analytical methods differ in their local surface fitting algorithms. Numerical methods do not fit a surface, but estimate either the curvature or the derivatives of the surface numerically, as done in the surface normal change methods[9] and the directional curvature methods.[7, 8] Note that the work in Ref. 9 is a little different in that it involves the surface normal rather than surface derivatives to estimate the principle curvature. Similarly, Shi *et al.*[10] also estimated the surface normal first. Tang and Medioni[11, 12] presented a novel approach based on the concept of "tensor voting" to recover the curvature information. Both the sign and the direction of principal curvatures are inferred directly from the input, which can be 3D sparse data. Since their approach does not demand local surface fitting, it can be viewed as a numerical method as well.

Even with extensive research on curvature estimation in the past, it is still difficult to generate satisfactory results in all cases. It was observed empirically in Ref. 2, 13 that qualitative curvature properties can be more reliably estimated than quantitative ones. Thus, some research efforts only focus on the recovery of the sign of the Gaussian curvature instead of its magnitude. Angelopoulou and Wolff[1] computed the sign of the Gaussian

E-mail: jingliap@usc.edu, qing_li@costard.usc.edu, cckuo@sipi.usc.edu, mlzhou@public.wh.hb.cn

curvature by checking whether the relative orientation of two local closed curves (one from the surface and the other from its corresponding curve on the Gauss sphere) is preserving or reversing.

Most previous work uses the range data as the input. Here, our work estimates Gaussian curvatures from the 3D mesh representation. On one hand, compared with the range data case, the sampling points (vertices) of a 3D mesh data are often irregularly and sparsely distributed so that surface fitting may not readily give an accurate approximation. On the other hand, there is one advantage with the 3D mesh representation. That is, vertex normals are often contained or implied in the 3D mesh data. Based on this observation and the geometry interpretation of the Gaussian curvature, we propose a simple yet efficient approach to estimate Gaussian curvatures from 3D meshes. Unlike most previous approaches, the proposed method does not use local surface fitting, partial derivative computation, or oriented normal vector recovery. The performance is further improved using the Bezier curve approximation.

The rest of this paper is organized as follows. The geometric interpretation of Gaussian curvature is introduced in Section 2. Two proposed approaches are described in Section 3 and 4. Experiments and analysis are given in Section 5. Some concluding remarks are given in Section 6.

2. GEOMETRIC INTERPRETATION OF GAUSSIAN CURVATURE

For detailed treatment of the Gaussian curvature, readers are referred to the classical differential geometry book by do Carmo.[15] Here, we briefly introduce some notations and basic concepts that are needed to understand this work.

Let $S \subset R^3$ be a surface and $S^2 \subset R^3$ be the surface of the unit sphere, *i.e.*

$$S^2 = \{(X, Y, Z) \in R^3; X^2 + Y^2 + Z^2 = 1\}.$$

The Gauss map N from S to S^2 is a mapping defined as

$$N(p) = (npx, npy, npz), \quad \forall p \in S,$$

where npx, npy and npz are the three components of the unit normal vector at point $p \in S$.

Let $\mathbf{x}(u, v)$ be a parameterization at point $p \in S$, we have

$$\begin{array}{rcl} N_u & = & a_{11}\mathbf{x}_u + a_{21}\mathbf{x}_v, \\ N_v & = & a_{12}\mathbf{x}_u + a_{22}\mathbf{x}_v, \end{array} \tag{1}$$

where subscripts u and v denote the partial derivatives of N and $\mathbf{x}$ with respect to u and v, respectively. The differential of the Gauss map N at point $p \in S$ is also a mapping denoted by

$$dN_p : T_p(S) \rightarrow T_p(S),$$

where $T_p(S)$ is the tangent plane at p. The Gaussian curvature K of S at p is defined as the determinant of dN_p. In mathematics, we have

$$K(p) = \det \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

where a_{ij} are coefficients in Equation (1).

Let V be a connected neighborhood of $p \in S$ where K does not change sign, and A be the area of a region $B \subset V$ containing p and A' be the area of B's image under the Gauss map N. Then, the area of B is given by

$$A = \iint_R |\mathbf{x}_u \times \mathbf{x}_v| du dv,$$

where R is the region in the $u - v$ plane corresponding to B, and the area of $N(B)$ is

$$A' = \iint_R |N_u \times N_v| du dv,$$

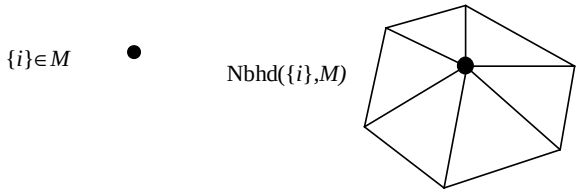


Figure 1. The neighborhood of a vertex in a mesh.

where $\times$ denotes vector outer product.

Using (1) and the definition of K, we have

$$A' = \iint_R K|\mathbf{x}_u \times \mathbf{x}_v| du dv.$$

Consider the limiting scenario where region A converges to point p. That is,

$$\lim_{A\to 0} \frac{A'}{A} = \lim_{R\to 0} \frac{A'/R}{A/R} = \frac{\lim_{R\to 0}(1/R)\iint_R K|\mathbf{x}_u \times \mathbf{x}_v| du dv}{\lim_{R\to 0}(1/R)\iint_R |\mathbf{x}_u \times \mathbf{x}_v| du dv} = \frac{K|\mathbf{x}_u \times \mathbf{x}_v|}{|\mathbf{x}_u \times \mathbf{x}_v|} = K(p),$$

where R denotes the area of region A. This gives the geometric interpretation of the Gaussian curvature introduced by Gauss himself.

The sign of Gaussian curvature $K(p)$ can be determined by checking whether the Gaussian map $N(p)$ changes the orientation within a small neighborhood around point p. For a basis $\{w_1, w_2\}$ in $T_p(S)$, we have

$$dN_p(w_1) \times dN_p(w_2) = det(dN_p)(w_1 \times w_2) = K(p)(w_1 \times w_2).$$

Thus, if the Gauss map $N(p)$ is orientation-preserving in the neighborhood of p, then $K(p) > 0$. Otherwise, $K(p) \leq 0$. Also, based on the geometry interpretation of the Gaussian curvature, if $A' = 0$, then $K(p) = 0$. Based on this observation, we can handle the points with the zero Gaussian curvature uniformly without a separate process.

3. BASIC APPROACH TO GAUSSIAN CURVATURE ESTIMATION

Let us examine a basic approach to Gaussian curvature estimation from 3D mesh data by following the geometric interpretation of the Gaussian curvature in this section. Consider the neighborhood of a vertex point p of mesh M. We can compute its area as well as the area of the corresponding image under the Gauss map. The product of their ratio and the sign of the Gaussian curvature at p is the resulting estimation of Gaussian curvature.

To be more specific, we adopt the mesh neighborhood definition given in Ref. 16:

$$nbhd(J, M) = \{s \in M; \exists s" \in J, s' \in M \text{ such that } s" \cup s \subset s'\},$$

where J can be in general a vertex, an edge or a face and M is a mesh. Fig. 1 shows an example of the neighborhood of vertex i.

For vertex $p \in M$, the area of its neighborhood is the sum of the area of its adjacent triangles:

$$A_p = \sum_{T\in nbhd(p)} A_T. \tag{2}$$

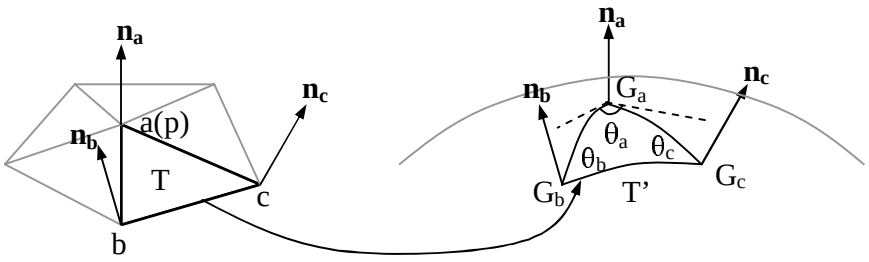


Figure 2. The image of a triangle in a mesh by the Gauss map.

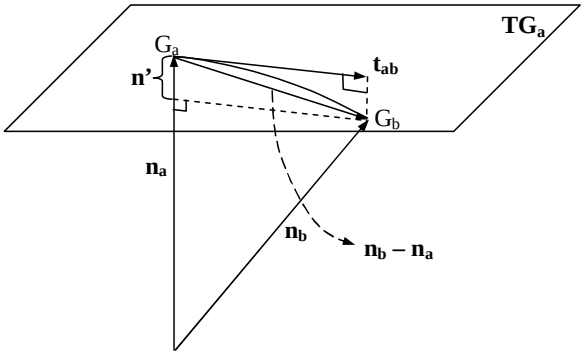


Figure 3. The relationship between $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{t}_{ab}$.

A polygonal mesh is a piecewise linear approximation of an object surface. Along each edge, the surface normal changes continuously from one end vertex to the other, and the image of an edge on the Gauss sphere is a major arc. As shown in Fig. 2, the image of triangle T under the Gauss map, denoted by T', is a spherical triangle enclosed by 3 major arcs on the Gauss sphere.

Let us denote the vertices of T by a, b and c with normals $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{n}_c$, respectively. The images of vertices of T under the Gauss map are vertices of T', which are denoted by G_a, G_b and G_c, respectively. Furthermore, we use θ_a, θ_b and θ_c to denote the inner angles of the spherical triangle T'. By the Gauss-Bonnet theorem, we know that

$$\theta_a + \theta_b + \theta_c > \pi.$$

The area A'_T of T' can be computed via Ref. 17

$$A'_T = (\theta_a + \theta_b + \theta_c - \pi) \times r^2 = \theta_a + \theta_b + \theta_c - \pi, \tag{3}$$

where $r = 1$ since the Gauss sphere is a unit sphere.

To compute A'_T, we need find the values of θ_a, θ_b and θ_c. Note that θ_a is not the angle between the line passing G_a, G_b and the line passing G_a, G_c but the angle between the projection of those two curved segments on the tangent plane, TG_a, at point G_a as shown with dashed lines in Fig. 2. We use $\mathbf{t}_{ab}$ and $\mathbf{t}_{ac}$ to denote the projection of the two curved segments connecting G_a, G_b and G_a, G_c on the unit sphere, respectively. Then, θ_a is the angle between $\mathbf{t}_{ab}$ and $\mathbf{t}_{ac}$.

A detailed ralationship between the unit normal vectors $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{t}_{ab}$ is plotted in Fig. 3. Let $\mathbf{n}'$ be the projection of $\mathbf{n}_b - \mathbf{n}_a$ on $\mathbf{n}_a$, *i.e.*

$$\mathbf{n}' = -((\mathbf{n}_b - \mathbf{n}_a) \cdot \mathbf{n}_a)\mathbf{n}_a,$$

where $\cdot$ denotes the vector inner product. Therefore, we have

$$\mathbf{t}_{ab} = (\mathbf{n}_b - \mathbf{n}_a) + \mathbf{n}' = (\mathbf{n}_b - \mathbf{n}_a) - ((\mathbf{n}_b - \mathbf{n}_a) \cdot \mathbf{n}_a)\mathbf{n}_a. \tag{4}$$

The vectors $\mathbf{t}_{ac}$ and $\mathbf{t}_{bc}$ can be derived in the same way. Then, we can find angles θ_a, θ_b and θ_c as

$$\theta_a = arccos(\frac{\mathbf{t}_{ab} \cdot \mathbf{t}_{ac}}{||\mathbf{t}_{ab} \cdot \mathbf{t}_{ac}||}), \tag{5}$$

$$\theta_b = arccos(\frac{\mathbf{t}_{bc} \cdot \mathbf{t}_{ba}}{||\mathbf{t}_{bc} \cdot \mathbf{t}_{ba}||}), \tag{6}$$

$$\theta_c = arccos(\frac{\mathbf{t}_{ca} \cdot \mathbf{t}_{cb}}{||\mathbf{t}_{ca} \cdot \mathbf{t}_{cb}||}). \tag{7}$$

Based on Eqs (3)-(7), we can obtain A'_T, the area of the spherical triangle T'. Thus, the area A'_p of the image of the neighborhood at p under the Gauss map can be computed via

$$A'_p = \sum_{T \in nbhd(p)} A'_T. \tag{8}$$

The Gaussian curvature sign $sign_T$ in a triangle T is determined by whether the Gauss map changes its orientation or not. Let vertices a, b and c of T be arranged in the right-hand order as shown in Fig. 2. Then, we have

$$sign_T = \begin{cases} 1, \text{ if } (\mathbf{t}_{ab} \times \mathbf{t}_{ac}) \cdot \mathbf{n}_a > 0 \\ 0, \text{ if } (\mathbf{t}_{ab} \times \mathbf{t}_{ac}) \cdot \mathbf{n}_a = 0 \\ -1, \text{ if } (\mathbf{t}_{ab} \times \mathbf{t}_{ac}) \cdot \mathbf{n}_a < 0 \end{cases}, \tag{9}$$

where $\times$ denots the vector outer product.

If the mesh neighborhood $nbhd(p, M)$ of vertex p is small enough, there should be no change of the Gaussian curvature sign. The sign should be the same in each triangle in $nbhd(p, M)$. Thus, we can calculate the signs for all triangles in $nbhd(p, M)$. If they are consistent to be either 1 or -1, it gives the sign $sign_p$ of the Gaussian curvature at p. On the other hand, if their values are not consistent (*i.e.* with some equal to 1 while others equal to -1), then the mesh at p is too coarse to approximate the original mesh and this basic approach is no more applicable. A more advanced approach is needed.

According to the Geometry interpretation of the Gaussian curvature given in (2), we can obtain the Gaussian curvature K_p at p via

$$K_p = sign_p \frac{A'_p}{A_p} = sign_{T_0} \frac{\sum_{T \in nbhd(p)} A'_T}{\sum_{T \in nbhd(p)} A_T}, \quad \forall T_0 \in nbhd(p), \tag{10}$$

where A'_T is computed using Eq. (3) and (5)- (7) and $sign_{T_0}$ is calculated by Eq. (9).

4. GAUSSIAN CURVATURE ESTIMATION WITH BEZIER CURVE APPROXIMATION AND SUBDIVISION

At point p, our estimation K_p in the above section would be exactly the same as the theoretical value only when the neighborhood of p is infinitely small. Since 3D meshes only provide a linear approximation to real object surfaces locally and a mesh neighborhood cannot be arbitrarily small, estimation errors are often inevitable. The coarser a 3D mesh is, the greater the estimation error would be. To improve the accuracy of our approach, we need refine the neighborhood of a vertex when it is too coarse. To do so, we need a mechanism to measure the mesh coarseness first.

In the 2D case, we often approximate a curve with a sequence of line segments, called polyline, as shown in Fig. 4. At a given vertex p_i, the accuracy of the approximation or the coarseness of the polyline can be measured

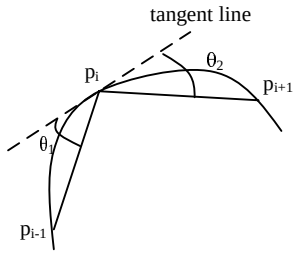


Figure 4. The coarseness measure for a polyline curve approximation.

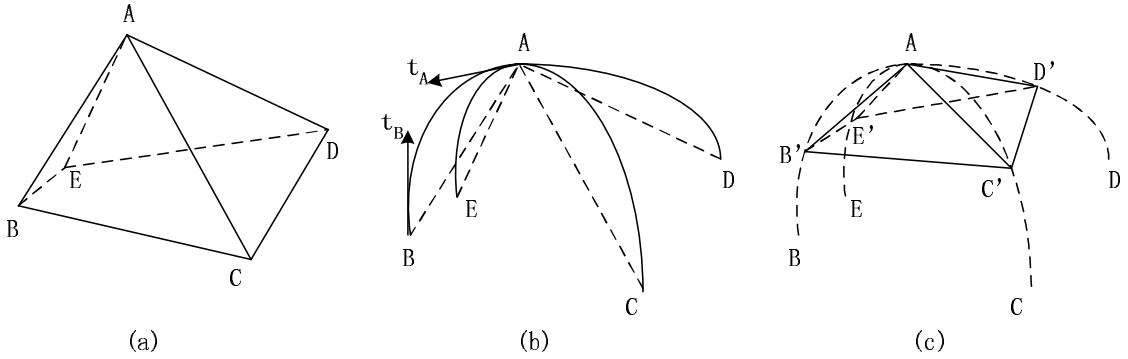


Figure 5. (a) The neighborhood around vertex A, (b) the Bezier curve approximation of edges incident to A, and (c) the refined neighborhood of A.

with the average angle between the tangent line at p_i and the two line segments adjacent to p_i. For instance, we can measure the coarseness at p_i via $(\theta_1 + \theta_2)/2$ as shown in Fig. 4. Similarly, at a vertex p of 3D meshes, we can use the average angle between each face adjacent to p and the tangent plane at p to measure the mesh coarseness at p.

Practically, we can set a coarseness threshold. When a vertex has a coarseness value higher than the threshold, we call it a coarse vertex and need to refine its neighborhood before applying our Gaussian curvature estimation method. Since 3D vertices are typically irregularly and sparsely distributed compared to data points in range data, local surface fitting complicates the computing but the result can still be very inaccurate. Instead, Bezier curves are used to approximate each edge incident to the vertex whose neighborhood is to be refined. This approach turns out to be much simpler.

For instance, Fig. 5(a) shows vertex A with its neighborhood. Let A be a coarse vertex, and each of its incident edges is approximated by a Bezier curve as shown in Fig. 5(b). Each vertex adjacent to A in the original mesh is replaced with the point on the corresponding Bezier curve with parameter value $t = 0.5$ in Fig. 5(c) to update the neighborhood of vertex A. Note that the neighborhood of vertex A is updated virtually for the curvature estimation purpose, while the input 3D mesh data are actually not changed.

To construct a Bezier curve, four parameters are needed. Two end points and the tangent vector at each of the end points. For instance, to approximate the edge between A and B in Fig. 5(b), we need to determine tangent vectors $\mathbf{t}_A$ and $\mathbf{t}_B$ at A and B, respectively. As shown in Fig. 6, we already know normals $\mathbf{n}_A$ and $\mathbf{n}_B$, thus tangent planes TG_A and TG_B at vertices A and B. The two tangent vectors $\mathbf{t}_A$ and $\mathbf{t}_B$ should be restricted on TG_A and TG_B. Assume TG_A and TG_B intersect at line l. Let the projections of A and B onto line l be I_A and I_B, respectively. We can search for point I_0 that belongs to the line segment between I_A and I_B so that the distance

$$||I_0 - A|| + ||I_0 - B||$$

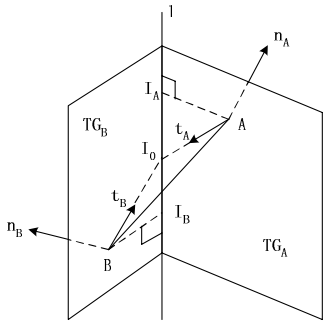


Figure 6. Determination of tangent vectors for the Bezier curve.

is minimized. Then, the two tangent vectors can be obtained from vertex A to point I_0 and from vertex B to point I_0. In the experiment, point I_0 is approximated by the midpoint between I_A and I_B, *i.e.*

$$I_0 \approx \frac{1}{2}(I_A + I_B),$$

which is simple yet producing good results.

Having determined the parameters for a Bezier curve, we use the recursive Bezier curve subdivision technique[18] to find the point with parameter $t = 0.5$, whose normal is simply approximated by the averaged sum of the two end points' normals. For instance, in Fig. 5(c), the approximated normal at vertex B' is

$$\frac{(\mathbf{n}_A + \mathbf{n}_B)/2}{||(\mathbf{n}_A + \mathbf{n}_B)/2||}.$$

Experimentally, the above simple Bezier curve approximation technique works well for our test mesh data. However, it is worth pointing out that when a 3D mesh is too coarse, it would be very difficult to recover the Gaussian curvatures accurately.

5. EXPERIMENTAL RESULTS

We present experimental results in two parts. In the first part, we consider three synthesized 3D mesh models, *i.e.* the ellipsoid, the cylinder and the hyperboloid, whose Gaussian curvatures can be calculated analytically[15] so that the accuracy of our estimation can be verified. In the second part, we apply the developed algorithm to several 3D test meshes to illustrate the applicatioin of the proposed Gaussian curvature estimation method to surface segmentation and feature extraction.

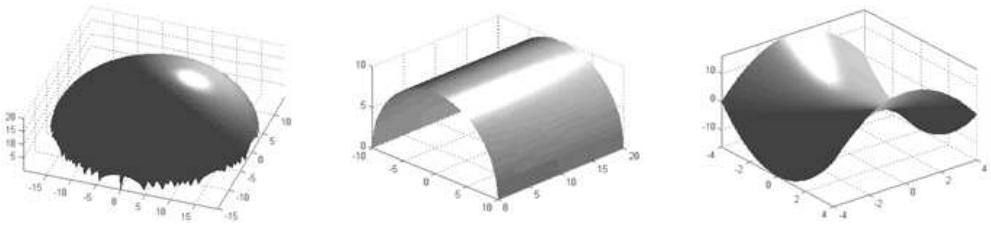
5.1. Synthesized Meshes

To obtain meshes for the ellipsoid, the cylinder and the hyperboloid, we first generate the range data from their analytical equations, and then construct meshes from the generated range data using the approach described in Ref. 19. To test the performance of our approach applied to meshes of different coarseness, we vary the sampling density in the range data generation to obtain meshes of different coarseness.

Fig. 7(a) shows the rendered result of the synthesized ellipsoid mesh with 2,397 vertices and 4,636 faces. The ellipsoid equation is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1.$$

In the experiment, we set $a = 20$, $b = 15$, $c = 20$. By varying the range image sampling density, we obtain a sequence of ellipsoid meshes of different coarseness. For each mesh, we take 100 non-boundary vertices evenly



(a) (b) (c)

Figure 7. Rendered images of synthesized meshes: (a) the ellipsoid, (b) the cylinder and (c) the hyperboloid.

Table 1. Estimation Error for Ellipsoid Meshes.

(v#,f#)	Average Coarseness	Basic Approach	Advanced Approach
(8,8)	0.597	54.3%	22.7%
(18,22)	0.353	32.1%	9.9%
(35,52)	0.275	18.8%	6.4%
(145,252)	0.138	12.2%	6.3%
(593,1108)	0.072	8.0%	6.7%
(2397,4636)	0.039	9.9%	10.6%

distributed on the surface*, and calculate the average coarseness and the average estimation error with our approach over all sampled vertices.

The experimental results are shown in Table 1, where the first column shows the number of vertices and faces in a mesh, the second column the average coarseness, the third column the average estimation error with the proposed basic approach given in Section 3 and the last column the average estimation error with the proposed advanced approach presented in Section 4. In Table 1, the coarseness is expressed in radian and the estimation error in percentage of difference of estimated values from theoretical values. The coarseness-error curve is shown in Fig. 8(a).

Fig. 7(b) shows the rendered image of the synthesized mesh with 4,225 vertices and 8,192 faces for a cylinder with equation $x^2 + z^2 = r^2$ with $r = 10$. The experimental results are given in Table 2. Since the theoretical Gaussian curvature of cylinder points is zero, we measure the estimation error with the absolute difference instead of the percentage. The coarseness-error curve is shown in Fig. 8(b). Note that the curves for the basic approach and the improved approach coincide.

Fig. 7(c) shows the rendered image of the synthesized mesh with 4,225 vertices and 8,192 faces for a hyperboloid with equation $z = \frac{x^2}{a^2} - \frac{y^2}{b^2}$, where $a = b = 1$. The experimental results are given in Table 3. The coarseness-error curve is shown in Fig. 8(c).

We see from the coarseness-error curves in Fig. 8 that our approach is particularly accurate for cylinder meshes, where the experiment results coincide with the theory values. For the other two kinds of meshes, to get

*If the mesh has less than 100 non-boundary vertices, we take them all.

Table 2. Estimation Error for Cylinder Meshes.

(v#,f#)	Average Coarseness	Basic Approach	Improved Approach
(9,8)	0.785	0	0
(25,32)	0.349	0	0
(35,52)	0.173	0	0
(81,128)	0.089	0	0
(1089,2048)	0.045	0	0
(4225,8192)	0.023	0	0

Table 3. Estimation Error for Hyperboloid Meshes.

(v#,f#)	Average Coarseness	Basic Approach	Improved Approach
(9,8)	0.955	90.5%	84.7%
(25,32)	0.402	24.1%	17.9%
(81,128)	0.182	11.1%	6.9%
(289,512)	0.090	5.5%	4.0%
(1089,2048)	0.043	3.4%	2.9%
(4225,8192)	0.021	2.9%	3.3%

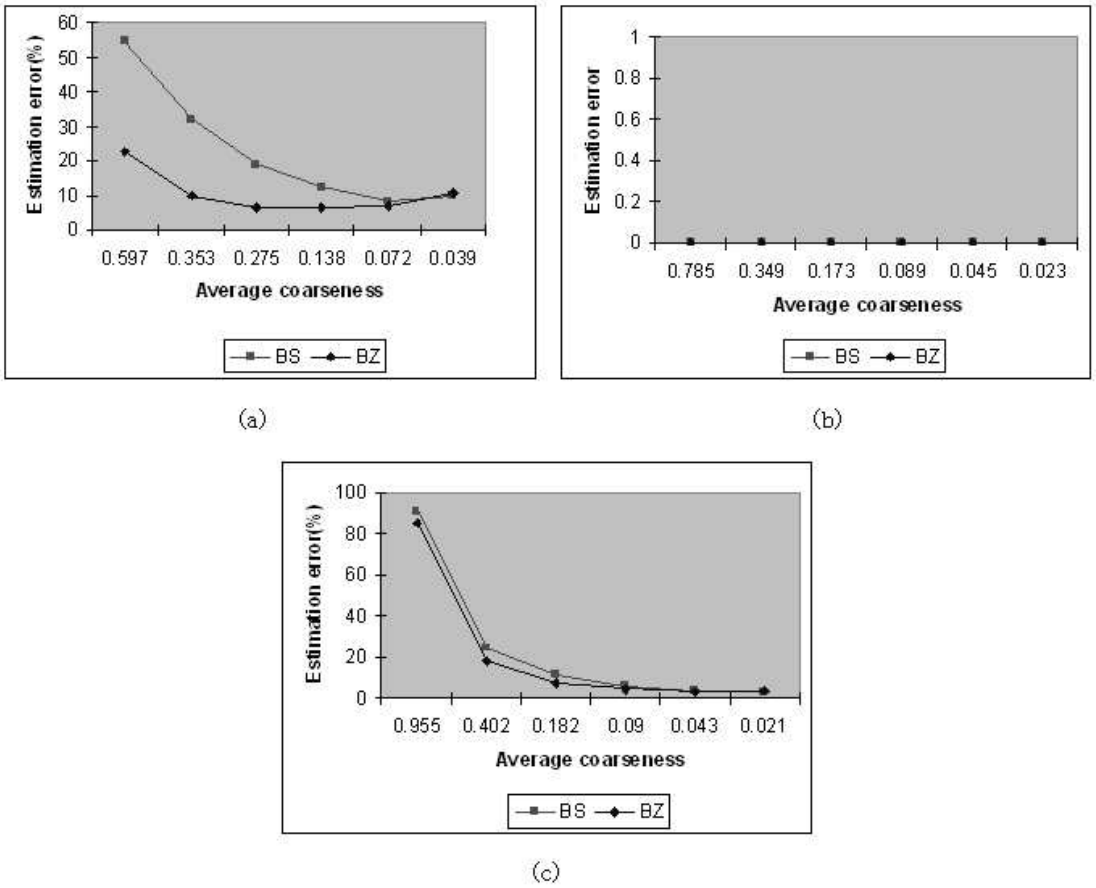


Figure 8. Error-coarseness curves of: (a) the ellipsoid meshes, (b) the cylinder meshes, and (c) the hyperboloid meshes, where BS and BZ denote the basic approach and the advanced approach respectively.

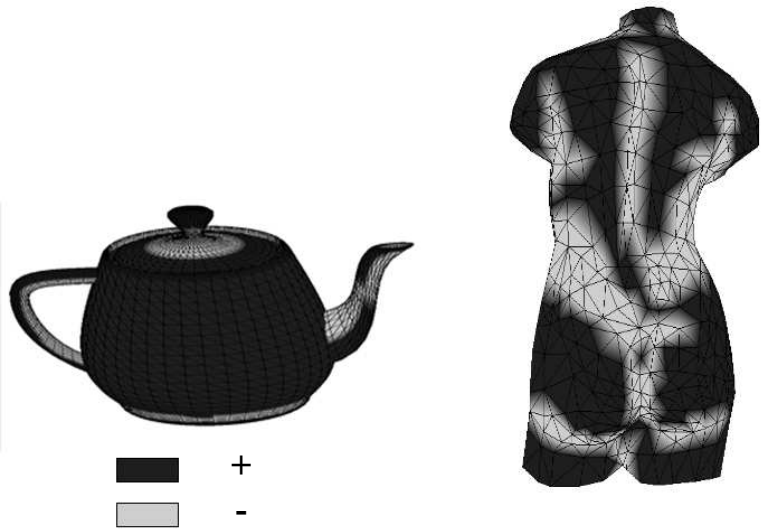


Figure 9. The sign of Gaussian curvature for (a) the teapot model, and (b) the Venus model, where the darker shade denotes the positive sign while the lighter shade for the negative sign.

an estimation error less than 10%, typically the average mesh coarseness should be no more than 0.3. Thus, we can set up a coarseness threshold as 0.3 for practical applications.

Actually, we have calculated the average coarseness for many popular 3D meshes and found that most of them have an average coarseness less than 0.3. Thus, the proposed Gaussian curvature estimation approach should have an error less than 10% when applied to these meshes. Note that the estimation error goes up a little bit at the tail of the curves in Fig. 8(a), which shows that, when the mesh is already very fine, the further refinement using the Bezier curve approximation may not help, and may even have a slightly negative effect on estimation accuracy.

5.2. Test Meshes

In this experiment, we calculate the signs of Gaussian curvatures for two test models, *i.e.* the teapot model and the Venus model and color triangular patches with different shades for different signs. The results are shown in Fig. 9, where regions with positive Gaussian curvatures are colored in a darker shade and regions with negative Gaussian curvatures in a lighter shade.

The surface patches with positive and negative Gaussian curvatures are neatly segmented in Fig. 9(a). Although the boundaries between negative and positive regions are not as clean in Fig. 9(b), the coloring still gives much hint about the shape of the body. These experiments on real world 3D meshes demonstrate the potential application of our work to practical problems such as mesh segmentation and feature extraction.

6. CONCLUSION AND FUTURE WORK

Two approaches to estimate Gaussian curvatures from 3D meshes based on Gaussian curvature's geometric interpretation were proposed in this research. They can recover both the sign and the magnitude of Gaussian curvature and can handle points with the zero Gaussian curvature uniformly. To improve estimation accuracy, we employed the Bezier curve approximation to refine the mesh neighborhood of a vertex. A mesh coarseness measure was given, and experiment results demonstrated that the proposed approach has an average error less than 10% for meshes with an average coarseness less than 0.3. Experiments on real world meshes demonstrated the potential application of our work to mesh segmentation or feature extraction.

Our experimental results are still preliminary. We would like to carry out more experiments and compare our approaches with other methods. Also, it is interesting to extend our approach to deal with noisy 3D meshes.

Being a tool, this developed technique may be useful in areas such as 3D feature extraction, 3D object recognition, etc.

REFERENCES

1. E. Angelopoulou and L. Wolff, "Sign of gaussian curvature from curve orientation in photometric space," *IEEE Transaction on Pattern Analysis and Machine Intelligence* **20**, pp. 1056–1066, Oct 1998.
2. P. J. Flynn and A. K. Jain, "On reliable curvature estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 110–116, 1989.
3. P. J. Besl and R. C. Jain, "Invariant surface characteristics for 3D object recognition in range images," *Computer Vision, Graphics and Image Processing* **33**, pp. 33–80, 1986.
4. F. P. Ferrie and M. D. Levine, "Deriving coarse 3D models of objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 345–353, 1988.
5. S. T. Li, *Research on the Key Techniques in Range Image Analysis*. PhD thesis, Tsinghua University, China, 2000.
6. B. C. Vemuri, A. Mitiche, and J. K. Aggarwal, "Curvature-based representation of objects from range data," *Image and Vision Computing* **4**, pp. 107–114, May 1986.
7. T. J. Fan, G. Medioni, and R. Nevatia, "Description of surfaces from range data using curvature properties," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 86–91, 1986.
8. T. J. Fan, G. Medioni, and R. Nevatia, "Surface segmentation and description from curvature features," in *Proceedings of DARPA Image Understanding Workshop*, pp. 351–359, 1987.
9. D. J. Ittner and A. K. Jain, "3-D surface discrimination from local curvature measures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 119–123, 1985.
10. P. Shi, G. Robinsont, and J. Duncan, "Myocardial motion and function assessment using 4D images," in *Proceedings of IEEE Conference on Visualization in Biomedical Computing*, pp. 148–159, Oct 1994.
11. C.-K. Tang and G. Medioni, "Inference of integrated surface, curve, and junction descriptions from sparse 3-d data," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(11), pp. 1206–1223, 1998.
12. C.-K. Tang and G. Medioni, "Robust estimation of curvature information from noisy 3d data for shape description," in *International Conference on Computer Vision*, pp. 426–433, 1999.
13. E. Trucco and R. B. Fisher, "Experiments in curvature-based segmentation of range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(2), pp. 177–182, 1995.
14. A. Blake and R. Cipolla, "Robust estimation of surface curvature from deformation of apparent contours," in *European Conference on Computer Vision*, pp. 465–474, 1990.
15. M. P. do Carmo, *Differential Geometry of Curves and Surface*, Prentice Hall, Englewood Cliffs, 1976.
16. H. Hoppe, *Surface Reconstruction from Unorganized Points*. PhD thesis, University of Washington, 1994.
17. D. W. Henderson, *Differential Geometry: A Geometric Introduction*, Prentice Hall, 1998.
18. E. Angel, *Interactive Computer Graphics: A Top-Down Approach with OpenGL, 2nd Ed.*, Addison Wesley, 2000.
19. G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *SIGGRAPH*, pp. 311–318, 1994.