

CMPUT 701

Carolina Díaz-Goano

1998

First Reader: Ben Watson

Second Reader: John Buchanan

1	INTRODUCTION	3
2	BACKGROUND	4
2.1	LOD ALGORITHMS.....	4
2.2	CURVATURE MEASURES.....	8
2.2.1	<i>Differential Geometry</i>	9
2.2.2	<i>Curvature algorithms</i>	15
3	A NEW CURVATURE MEASURE	16
3.1	OVERVIEW	16
3.2	THE CONTROL SPHERE ALGORITHM	18
3.2.1	<i>Calculation of the best fit plane</i>	21
3.2.2	<i>Statistical analysis</i>	23
3.2.3	<i>The curvature function</i>	24
4	IMPLEMENTATION	25
5	RESULTS	27
6	DISCUSSION	30
7	CONCLUSIONS AND FUTURE WORK	32
8	REFERENCES	33

Analysis of surface curvature for polygonal models

Carolina Díaz-Goano
carolina@cs.ualberta.ca

Abstract

Highly detailed geometric models have become commonplace in computer graphics. However, in many cases the complexity of such models exceeds the ability of the graphics hardware to render them interactively. Polygonal simplification methods have emerged as a solution to this problem.

In this paper we present a new technique to calculate surface curvature on a polygonal model at multiple scales. Given an input model, our algorithm calculates curvature values at the vertices of the model. This curvature analysis provides us with a measure that can be used in fundamental polygonal simplification operations such as adaptive subdivision and refinement, using the curvature (shape) information to decide which area to simplify in order to best preserve the original overall model appearance.

1 INTRODUCTION

Models in computer graphics are usually represented using polygonal meshes, which specify the vertices of the polygons and their orientation. This kind of representation is desirable since it can be efficiently rendered and smoothly shaded for curved surfaces.

Many computer graphics applications require complex, highly detailed polygonal models in order to provide a convincing degree of realism. However, the total complexity of these models is not always required and the cost of displaying such models is directly proportional to the level of their complexity.

In order to render at interactive frame rates models containing many polygons have to be simplified and substituted by less complex representations. Having multiple representations for one model facilitates a balance between realism and speed of display. These alternative representations are referred to as Levels of Detail (LODs).

Lately, much research effort has been focused on the automatic production of simplified models. Many algorithms have been proposed for model simplification. Most of the techniques involve developing an error measure that guides the algorithm through the simplification process quantifying the difference between the original polygonal model and its simplified versions. An error metric guarantees that the modified mesh stays within a prescribed tolerance to the original data thus preserving its original appearance.

In this paper we propose a new algorithm to measure curvature on a polygonal model surface. Our algorithm takes a polygonal model as an input and provides us with the information concerning its shape (curvature) without the need of having a mathematical representation of the surface. We believe this curvature measure can be useful for the LOD algorithms by providing a new criterion for polygon simplification decisions.

Other potential uses of this curvature measure are: in design, as a means to detect changes of the curvature of a surface helping, in this way, to improve the manufacture and visualization of sculptured surfaces; in computer graphics, as a way to compare appearance and shape of models.

The rest of this document is organized as follows: Section 2 presents a survey on polygonal model simplification techniques, followed by some

background in differential geometry and an overview of previous work on curvature measures. Section 3 introduces our new algorithm for estimating curvature on a polygonal model. Section 4 addresses implementation issues. Sections 5 and 6 present and discuss our experimental results. Finally, Section 7 states some conclusions concerning the contributions of this work.

2 BACKGROUND

2.1 LOD Algorithms

A number of methods have been proposed to create Levels of Detail. In general, these algorithms operate on a polygonal representation of an object and require a well-defined input model.

Given a polygonal representation of an object, it can be simplified in many different ways:

- Selectively removing vertices. (Schroeder *et al.*) [1]
- Redistributing vertices over a surface. (Turk) [2]
- Clustering vertices (Rossignac & Borel, Lindstrom & Turk) [3] [9]
- Merging nearly coplanar polygons. (Hinker & Hansen) [4]
- Removing polygons in order of curvature. (Hamann) [5]
- Using Wavelet theory. (Eck *et al.*) [6]
- Minimizing an energy function. (Hoppe) [7]
- Using bounding surfaces. (Cohen *et al.*) [8]

In the rest of this section we describe the basic ideas of these techniques.

Schroeder *et al.* (1992) [1] present an algorithm that uses vertex removal to reduce the number of polygons of a model. The algorithm makes multiple passes over the set of vertices of the triangle mesh and removes vertices according to predefined criteria. As a result of a removal a hole is created which a triangulation process then fills. The algorithm continues until the mesh reaches a decimation criterion defined by the user. There are three main stages in this algorithm:

- 1- Vertex local geometry and topology¹ characterization: here the vertices are grouped in one of five defined types.
- 2- Decimation criteria evaluation: there are two criteria for deleting a vertex. The criterion used depends upon the vertex type. The first criterion is the vertex distance to an average plane (calculated using the normals, centers and areas of all the triangles that use that vertex). If the vertex is within a user defined distance then it is deleted and re-triangulation follows. The second criterion is the vertex distance to an edge.
- 3- Re-triangulation of the holes: a recursive loop splitting process is used to produce new triangles to fill the holes left. Each loop is divided into halves. The division is along a line defined from two non-neighboring vertices in the loop. The recursion continues until only three vertices remain forming a new triangle.

Turk (1992) [2] introduces an automatic method for creating surface models at several levels of detail from an original polygonal model. The idea is to reduce the number of polygons by adding new points on the surface of the model, connecting these new vertices into triangles forming a mutual tessellation (an intermediate polygonal surface that incorporates both the old vertices and the newly placed ones) and then deleting the old vertices, thus re-tiling the surface. The first step of the method is the selection of the new set of points. This is accomplished by distributing points randomly over the surface and then using a relaxation procedure to obtain a uniform distribution. In order to maintain accurately the features of the surface, if the surface presents a great variation in the amount of curvature at different locations, the point distribution is modified so that more new vertices are placed in regions of higher curvature. Here curvature is calculated by examining the edges and normals of the polygonal mesh. An estimate of the radius of curvature is computed. By estimating the curvature the polygons can be placed in a way that better reflects the actual shape of the model.

¹ Topology refers to the structure of a connected 2-D manifold, or mesh.

Rossignac and Borel (1992) [3] describe a technique to reduce polygonal models by merging vertices based on their spatial proximity. Their scheme neither requires nor preserves a valid topology; hence it can deal with degenerate models. The algorithm starts by determining the importance of each vertex based on two criteria: the area of the face associated with a given vertex and curvature. Next the polygonal model is subdivided in a regular way forming a 3-D grid. All vertices within each cell of the grid are collapsed into a single representative vertex for the voxel. The resolution of the grid determines the level of detail achieved. Rossignac and Borel's algorithm is very robust and can be implemented efficiently. However, it suffers from some disadvantages. One of them is that the algorithm is sensitive to the orientation of the clustering grid. Also, since topology is not preserved and no explicit error bound is guaranteed the resulting simplification can be less visually pleasing than those slower algorithms. Finally, there is no way to predict the number of polygons in the final mesh since this will be the result of a specified grid resolution.

Hinker and Hansen (1993) [4] present another technique for reducing polygonal models based on identifying coplanar or nearly coplanar polygons, merging them together into a larger complex polygon and finally re-triangulating them into fewer polygons than the original description. Their method has the advantage over other methods (i.e. surface re-tiling, mesh decimation) that auxiliary information is retained and the optimization is fast. This method obtains impressive reductions for large models with nearly flat surfaces. However, as noted by the authors, the algorithm offers small gains for surfaces of high curvature.

Hamann (1994) [5] introduces a data reduction scheme for triangulated surfaces. Given a surface triangulation in the 3-D space, each triangle is weighted according to the principal curvature at its vertices. These curvature values are pre-computed based on the discrete triangulated representation of the surface. Curvature is calculated based on a parametric representation of the surface. A triangle is associated with a surface region with low curvature if the sum of the absolute curvatures at its vertices is low. This measure is used as a weight to determine a triangle's significance. The lower the curvature value at the vertices of a triangle the lower its weight. The triangle with lowest weight is identified and removed. A new point that lies on a local surface replaces this triangle. The region affected by the removal is re-triangulated and new weights are computed for all the new introduced triangles. The triangle removal algorithm allows the user to specify a percentage of triangles to be removed or an error tolerance.

Eck *et al.* (1995) [6] present an adaptive subdivision algorithm that uses a compact wavelet representation to guide the recursive subdivision process, overcoming the subdivision connectivity restriction². The multiresolution of a mesh consists of a sample mesh together with a sequence of local correction terms (i.e. wavelet coefficients) that capture the detail present in the model at various resolutions. Multiresolution mesh representations are particularly convenient for compression simplification, progressive display and LOD control. In their multiresolution analysis Eck *et al.* smoothly interpolate between LOD by adding or subtracting wavelet coefficients. The algorithm uses enough wavelet coefficients to meet a user specified error bound (the simplified surface lies within a user specified distance from the original model). An important contribution of this method is that it overcomes the subdivision connectivity restriction, thus all arbitrary meshes can be converted to multiresolution form.

Hoppe (1996) [7] presents in his work “Progressive Meshes” a new scheme for storing and transmitting arbitrary triangle meshes. He also introduces a new mesh simplification procedure. A Progressive Mesh consist of a base mesh created by a sequence of edge collapses together with a sequence of detail records that indicate how to incrementally refine the base mesh exactly back to the original mesh. He defines two dual transformations: vertex split and edge collapse. Each vertex split replaces a vertex by two edge-connected vertices, thus creating a new vertex and two triangles. The sequence of collapses during a simplification is determined explicitly as an energy function to be minimized. All edges to be collapsed are evaluated according to this energy function and sorted into a priority queue. The operation with the lowest energy cost is performed and removed from the front of the queue. After this operation nearby edges are re-evaluated and the process continues until the topological constraints prevent from further simplification. This algorithm is suitable for real time LOD management.

Cohen *et al.* (1996) [8] proposed a method called “Simplification envelopes” for generating a hierarchy of LOD approximations for a given polygonal model. Their approach guarantees an error bound from the original model and enforces global as well as local topology. Simplification envelopes of a surface consists of two offset surfaces. The outer and inner envelope are generated by displacing each vertex of the original mesh along its normal by a defined distance ε and $-\varepsilon$ respectively. The envelopes are not allowed to self intersect, where such case occurs, the distance is locally decreased. Once the envelopes have been created, they guide the simplification

² A mesh is said to have subdivision connectivity if it can be obtained from a simple base mesh by recursive 1-to-4 splitting.

process; the new model is formed within them. Cohen *et al.* present two decimation approaches that remove triangles or vertices in an iterative way and re-triangulate the resulting holes.

Lindstrom and Turk (1998) [9] introduce another way to simplify models. Their work uses edge collapse as the method for simplifying geometry. Their approach selects an edge and replaces it with a single vertex. This removes one vertex, three edges and two faces. They consider this operation adequate because it allows the new vertex to be placed in a manner that helps to preserve the location and shape, and does not require the invocation of a triangulation algorithm. Two decisions are central to the method they use: the position of the new vertex created by the edge collapse and the ordering of the edges to be collapsed. In their work they use volume and surface information to make both decisions. The placement of the new vertex is constrained by maintaining the volume of the closed model constant. If a new vertex is near a boundary of the model, they preserve the surface area in the region surrounding the edge that is being collapsed. Since these conditions do not fully determine the position of the new vertex, they also optimize additional geometry properties. These properties are unified by describing each of them as one or more planes where the new vertex must lie. Per triangle volume and area differences are used to dictate the priority of the edge collapses.

2.2 Curvature measures

In order to measure the curvature of a surface it is necessary to first define a means for extracting this geometric information. To do so we should make use of the classic mathematical descriptions of surface shape provided by differential geometry.

2.2.1 Differential Geometry

Essentially there exist two distinct and important notions of curvature: **Extrinsic Curvature** and **Intrinsic Curvature**. Extrinsic curvature depends on the surface particular embedding while intrinsic curvature is detectable without the information of the space surrounding the surface. For a curve in a 3-D space, all the curvature is extrinsic [11]. On the other hand, surfaces have both intrinsic and extrinsic curvature.

The main measures for curvature are the **Mean curvature** (H), the **Gaussian curvature** (K), and the **Weingarten map**. The Gaussian curvature is intrinsic, whereas Mean curvature and the Weingarten map are not.

To understand what the Gaussian curvature of a point on a surface is, it is helpful to understand the concept of curvature of a curve. On a curve in the plane, the line best approximating the curve that passes through a given point is the tangent line to the curve at that point. The best approximating circle that passes through that point and is tangent to the curve can also be found. The reciprocal of the radius of that circle is the curvature of the curve at that point.

The tangent plane and the normal to the surface are some other useful concepts necessary to understand the notion of curvature on a surface.

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, and let $p \in M$ be a point. A vector v in \mathbb{R}^3 is a tangent vector to M at p if there is a curve $c: (-\varepsilon, \varepsilon) \rightarrow M$ for some number $\varepsilon > 0$ such that $c(0) = p$ and $c'(0) = v$. The collection of all tangent vectors to M at p is denoted by T_pM and is called **tangent plane** to M at p . Figure 2.1 illustrates this concept.

If the smooth surface $M \subset \mathbb{R}^3$, then T_pM is a two dimensional vector space. Moreover, if $x: U \rightarrow M$ is a coordinate patch such that $p \in x(U)$, then T_pM is the subspace of \mathbb{R}^3 spanned by the vectors $\{x_1, x_2\}$ evaluated at $x^{-1}(p)$. [10]

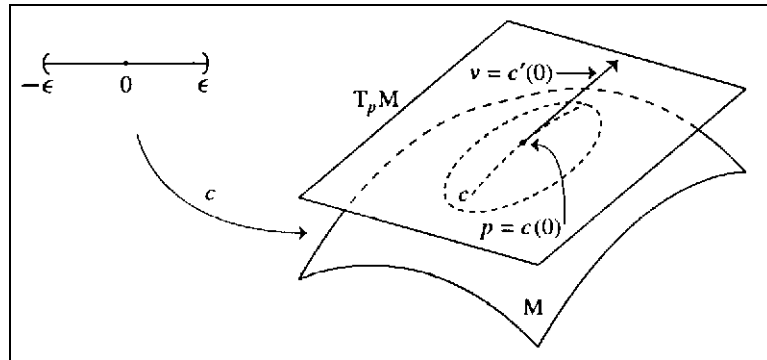


Figure 2.1- Tangent Plane (T_pM) at a point p .

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, and let $x: U \rightarrow M$ be a coordinate patch such that $p \in x(U)$. Then the normal vector function $n: U \rightarrow S^2$ with respect to x is defined by:

$$n(p) = \frac{x_1(p) \times x_2(p)}{\|x_1(p) \times x_2(p)\|}$$

for each $p \in x(U)$. Where x_1 and x_2 are the vectors that span the tangent plane to M at p [10].

A generalization of curvature for surfaces is the **Normal Section Curvature**. Given a point on a surface and a direction lying in the tangent plane of the surface at that point, the normal section curvature can be computed by intersecting the surface with the plane spanned by the point, the normal to the surface at that point and the chosen direction. The normal section curvature is the curvature of this curve at the point of interest.

Considering all directions in the tangent plane to the surface at a given point a maximum and a minimum value of normal section curvature can be found (k_1 and k_2 respectively). The Gaussian Curvature is the product of these maximum and minimum values. The Mean curvature is the average of them. We can then write Gaussian Curvature as:

$$K(p) = k_1 \cdot k_2$$

And the Mean Curvature as:

$$H(p) = \frac{k_1 + k_2}{2}$$

We can view curvature as a measure of how a surface deviates from being planar [10].

The **Gaussian curvature** at a point p can also be expressed by the ratio between the area of a region spanned by the normals to the surface (dAn), to the area of a small region on the surface containing p in its interior (dAp) [10]. Figure 2.2 shows the curvature at point p for a given surface.

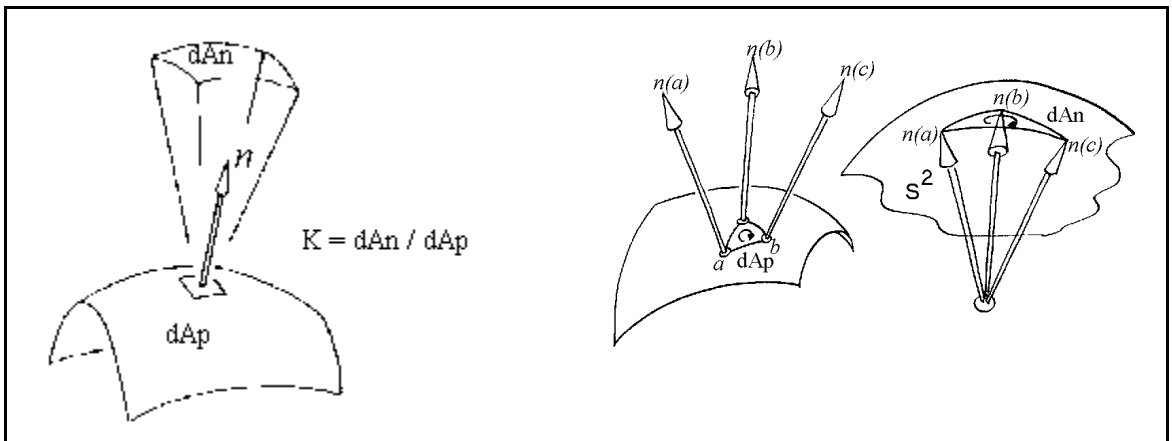


Figure 2.2 - Gaussian Curvature as a ratio of areas

The area ratio can be written as:

$$K(p) = \lim_{Area(p) \rightarrow p} \frac{Area(n(p))}{Area(p)}$$

$$Area(p) \rightarrow p$$

A positive Gaussian curvature value means that the surface is locally either a peak or a valley. A negative value means the surface locally has saddle points. A zero value means the surface is flat in at least one direction.

Table 2.1 shows the classification of a point p on a regular surface $M \subset \mathbb{R}^3$ based on the sign of the Gaussian Curvature at that point ($K(p)$):

Sign	Point
$K(p) > 0$	Elliptic
$K(p) < 0$	Hyperbolic
$K(p) = 0$	Parabolic or Planar

Table 2.1 – Classification of a point p based on Gaussian Curvature

Surfaces with constant Gaussian curvature include cones, cylinders, planes and spheres.

The Gauss Map can be defined as the function that maps points on a surface to their normals placed in a sphere in \mathbb{R}^3 centered at the origin [11].

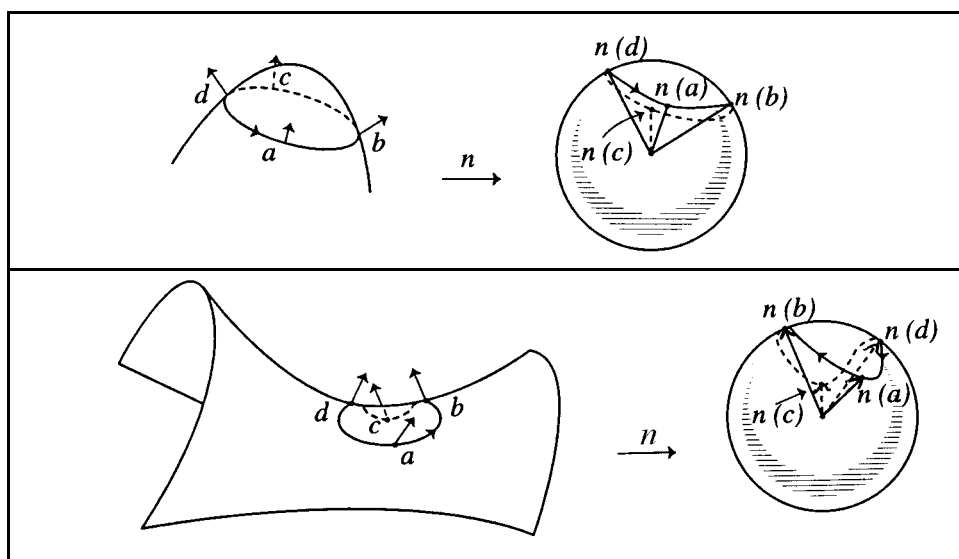


Figure 2.3- Gauss Maps

Another useful function is the Weingarten map, also called the Shape Operator.

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, let $x: U \rightarrow M$ be a coordinate patch and let $p \in x(U)$ be a point. The Weingarten map of M at p is the map $L: TpM \rightarrow TpM$ given by:

$$L(v) = -\nabla_v n$$

for all $v \in TpM$.

We can use the vectors $\{x_1, x_2\}$ basis for TpM to compute matrices for the Weingarten map:

$$L = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}$$

Where:

$$L_{ij} = \langle x_i, x_j \rangle^{-1} \langle n, x_{ji} \rangle \quad i = 1, 2 \quad j = 1, 2$$

The Weingarten map very much depends upon the way in which a surface sits in \mathbb{R}^3 .

Gaussian and Mean curvature can now be expressed as functions of the Weingarten map :

$$K(p) = \det(L)$$

$$H(p) = \frac{\text{tr}(L)}{2}$$

Also note that k_1 and k_2 (principal curvatures) are the eigenvalues of the Weingarten map of M at p , and that their eigenvectors are the principal directions of M at p .

Three definitions of the so called **Fundamental Forms** are extremely important and useful in determining the properties of a surface, such as Gaussian and Mean curvature. The most important are the First and Second Fundamental Forms (since the third can be expressed in terms of these two).

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, and let $p \in M$ be a point. The **First Fundamental Form** of M at p , is the function: $I_p: TpM \times TpM \rightarrow \mathbb{R}$ defined by:

$$I_p(v, w) = \langle v, w \rangle$$

where $\langle v, w \rangle$ is the standard inner product in \mathbb{R}^3 of the vectors $v, w \in TpM \subset \mathbb{R}^3$. The first fundamental form of M is the collection, denoted I , of all functions I_p at all points $p \in M$.

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, let $x: U \rightarrow M$ be a coordinate patch and let $p \in x(U)$ be a point. The **Second Fundamental Form** of M at p is the bilinear form $II_p: TpM \times TpM \rightarrow \mathbb{R}$ induced by the linear map $L: TpM \rightarrow TpM$, that is:

$$II_p(v, w) = \langle L(v), w \rangle$$

for all $v, w \in TpM$. L is the Weingarten Map defined by the negative derivative of the unit normal vector field of the surface. The second fundamental form of M is the collection, denoted II of all functions II_p at all points $p \in M$. [10]

As for the Weingarten map, the second fundamental form is an extrinsic quantity. II can be thought as of an operator that reveals how the tangent planes change in the neighborhood of any point on the surface [11].

Definition: Let $M \subset \mathbb{R}^3$ be a smooth surface, let $x: U \rightarrow M$ be a coordinate patch and let $p \in x(U)$ be a point. The **Third Fundamental Form** of M at p is the bilinear form $III_p: TpM \times TpM \rightarrow \mathbb{R}$ induced by the linear map $L: TpM \rightarrow TpM$, that is:

$$III_p(v, w) = \langle L(v), L(w) \rangle$$

for all $v, w \in TpM$. The third fundamental form of M is the collection, denoted III of all functions III_p at all points $p \in M$.

The Third Fundamental Form can also be written in terms of the First and Second Fundamental Forms by:

$$III - 2H \cdot II + K \cdot I = 0$$

Where:

H : Mean Curvature.

K : Gaussian Curvature.

The different operators defined above can be used to extract geometrical information about surfaces. Thus they can be used to obtain information about the shape of 3-D polygonal models.

2.2.2 Curvature algorithms

By combining results from differential geometry with raster graphics, Dill (1981) [12] presents a formulation for calculating and displaying the Gaussian and Mean curvature of parametric surfaces. In his research he calculates curvature values applying the mathematical definition of curvature to known parametric curves that describe the surfaces to be rendered. Then the curvature results are presented at a polygonal level by averaging the values obtained at each vertex and using a color encoded curvature value.

Hamann (1993) [13] presents a curvature approximation for triangulated surfaces. Given a set of points and normals he presents a scheme for approximating principal curvature. The approximation is based on the fact that a surface can be locally represented as a graph of bivariate functions. Quadratic polynomials are used for this local approximation. The principal curvature at a point on the graph of such a quadric is used as the approximation of the principal curvatures at the original surface point. The quality of the curvature approximation is tested for triangulated surfaces obtained from a known parametric surface.

Elber *et al.* (1997) [14] develop a method using hybrid symbolic and numeric operators to create trimmed surfaces each of which is solely convex, or a saddle and partitions the original surface. The method is also used to identify regions whose curvature lies within a specified bound. Their work is based on the symbolic computation of the operators they defined (variants of Gaussian and Mean curvature) and in the use of NURBS (Non Uniform Rational B-Splines). Thus their work is a second order analysis of parametric surfaces.

Interrante (1997) [15] presents a way to explicitly indicate the relative depth of a transparent surface and communicate the essential features of its 3-D shape. In her work she calculates the principal directions and the curvature along these directions. She computes these measurements by calculating a smoothly curving surface for the data points. Using parametric functions eigenvalues and eigenvectors of the Second Fundamental Form for the surface are computed. Thus curvature values and direction of principal curvature are calculated.

3 A NEW CURVATURE MEASURE

3.1 Overview

There are two purposes for this research. The first one is to study possible ways of measuring curvature of a polygonal mesh. The second one is to implement an algorithm to efficiently compute one of these possible curvature measures.

In section 2.2.2 we reviewed previous work where different ways to calculate curvature on a polygonal mesh were presented. All the work mentioned above approximate the model's surface by parametric functions. We would like to develop an algorithm that takes as input a polygonal model (vertices and their arrangement) and provides us with information concerning its shape (curvature) without the need of having a mathematical representation of the surface.

This approach is not as accurate as the previous ones but still gives us the necessary information to determine the surface curvature at different scales. It is important to note that on a polygonal model, there exist multiple scales of curvature [16]. Curvature can be severe at a small scale, while negligible at a large scale and vice versa. Figures 3.1 and 3.2 illustrate this idea.

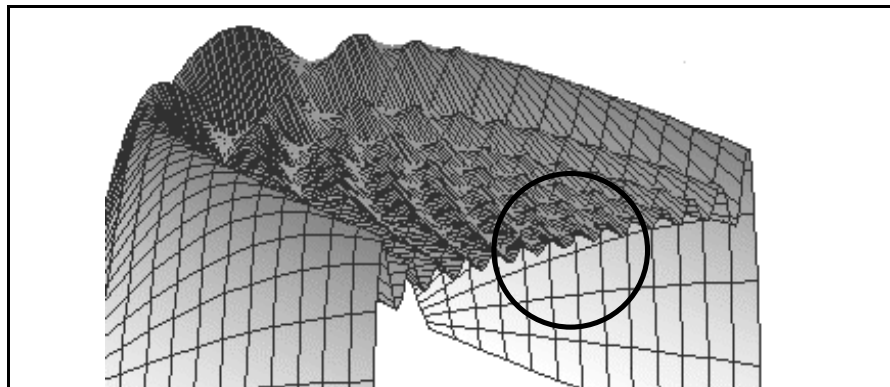


Figure 3.1: A surface with high small-scale curvature and little large-scale curvature.

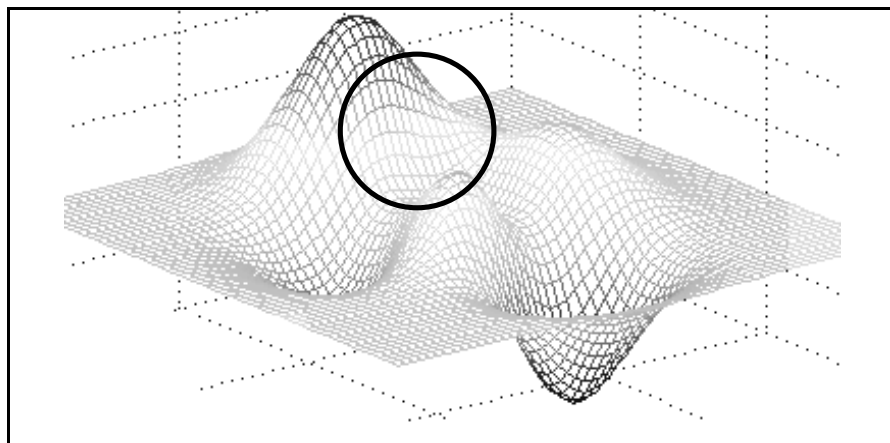


Figure 3.2: A surface with little small-scale curvature and high large-scale curvature.

In the following sections we present our new curvature algorithm: the Control Sphere algorithm.

3.2 The Control Sphere Algorithm

The main idea of this algorithm is to calculate curvature for sampled points on a model at different scales. This is achieved by creating a “control sphere”, that is a sphere whose radius varies according to the desired scale of analysis, and voxelizing the space within the sphere (sphere’s volume). Each voxel inside the sphere is marked according to whether it is intersected by the polygons of the model or not. With those voxels that are intersected by the model, a plane is found (that best fits that set of points). Finally a statistical test is computed to evaluate the goodness of that fit. This goodness of fit is then transformed into a curvature value.

It should be clear that a perfect fit means that the surface is a plane and thus curvature must be 0. On the other hand, a curved surface will give a poor fit.

Figure 3.3 shows the control sphere setting for different scales. For each case the planes are found with different sets of voxels according to the radius of the control sphere.

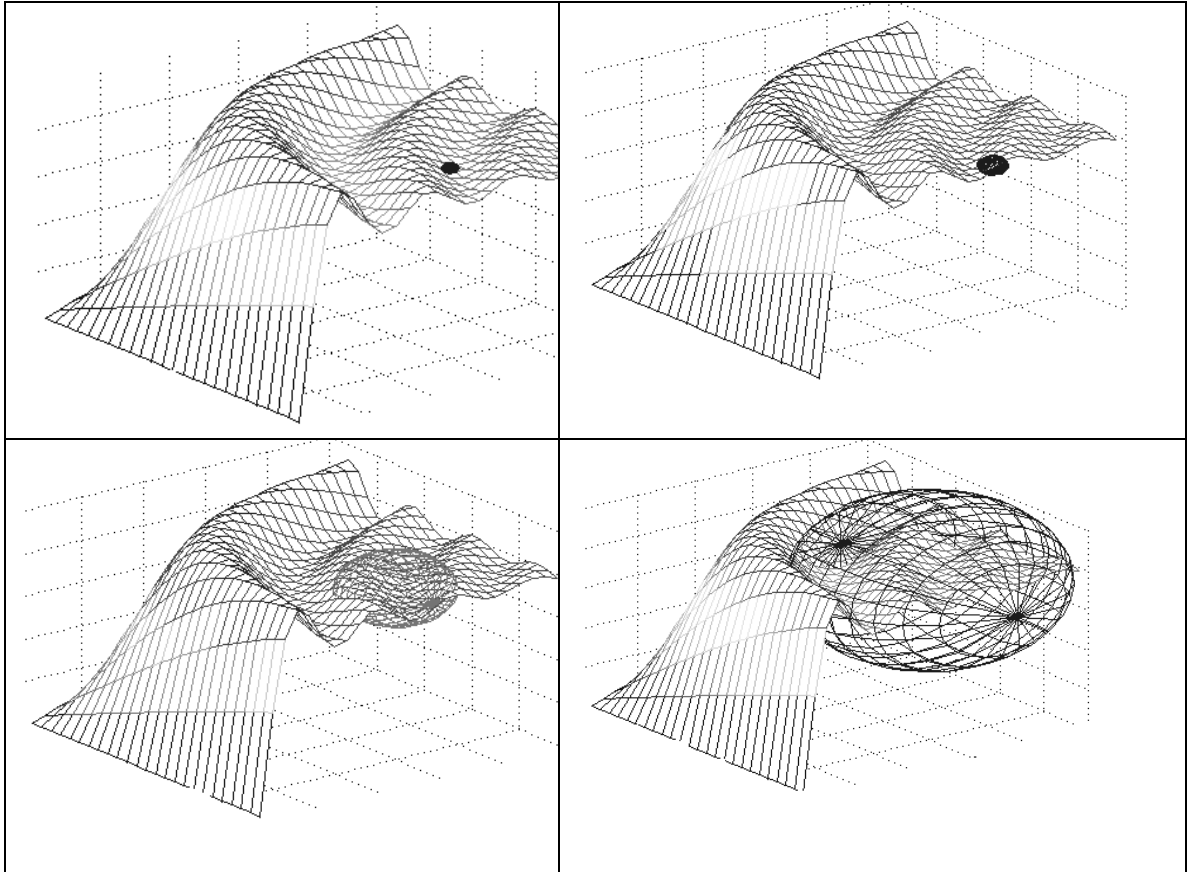


Figure 3.3 Control Sphere Algorithm- Different scales

The Control Sphere algorithm is computed for a sampled point on the polygonal model as follows:

1. Create a control sphere centered at the given point and whose radius is related to the desired analysis scale.
2. Voxelize the control sphere's volume.
3. For each voxel of the control sphere:
 - Check if it is intersected by the polygonal model.
 - If there is an intersection store the intersection point in a list, L .

4. With the list of intersection points, L , calculate the best fit plane through the points in L .
5. Perform a statistical analysis of the goodness of fit.
6. Return a value of curvature that is inversely proportional to the goodness of fit.

Steps 4, 5, and 6 are explained in the following sections. Figure 3.4 illustrates the algorithm's idea.

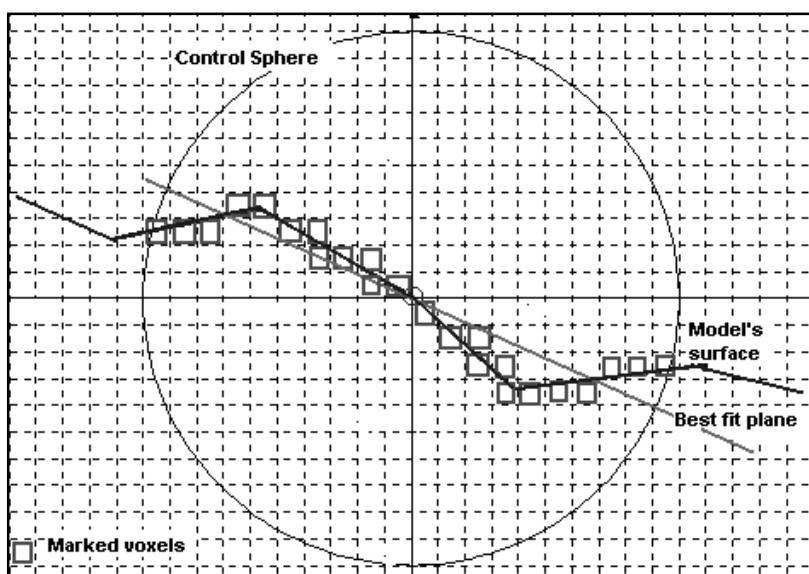


Figure 3.4 – Fitting a plane through the marked voxels in the control sphere.

3.2.1 Calculation of the best fit plane

The best fit plane for the set of points in L is calculated using the Least Square method. A system of linear equations can be written by relating the plane equation and the points in L .

The plane equation is given by:

$$A x + B y + C z + D = 0$$

For a given point $p = (x, y, z)$ this equation can be rewritten as:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 0$$

Applying the plane equation to all the points in L the following overdetermined system is obtained:

$$P x = b$$

Where:

$$P = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_n & y_n & z_n & 1 \end{bmatrix}$$

P is the matrix formed by the coordinates of the points $p_i = [x_i \ y_i \ z_i] \in L$ augmented by a $[1 \ 1 \ 1 \ \dots \ 1 \ 1]$ column vector.

$$x = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}, \text{ the plane coefficient vector that we want to find.}$$

$$b = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

The coefficients of the plane equation can then be found by setting and solving the following system of normal equations:

$$P^T P x = P^T b$$

Where:

P^T : is P transposed.

Provided the determinant of $P^T P$ does not vanish the system can be immediately solved using Gaussian Elimination.

Due to the symmetry of some models, $P^T P$ could be in some cases close to singular. For those cases the plane coefficients are calculated using Newell's method for computing a plane equation [17].

Using Newell's method the equation of the plane is calculated computing the polygon normal and using a reference point to compute the last coefficient of the plane equation. The normals are calculated as:

$$n = \begin{bmatrix} \sum (y_{i+1} - y_i)(z_{i+1} + z_i) \\ \sum (z_{i+1} - z_i)(x_{i+1} + x_i) \\ \sum (x_{i+1} - x_i)(y_{i+1} + y_i) \end{bmatrix}$$

3.2.2 Statistical analysis

Once the plane that best fits the set of points is found the deviation error is calculated. This was first done by comparing the deviation obtained from evaluating the plane for a given component with the original value of that component (i.e. the z component was compared against the z value obtained from evaluating the plane equation). But since this error measure could give very different results according to different components and to different orientations a new error measure was used. The error is then calculated as the summation of the distances from each point $p_i \in L$ to the calculated plane:

$$Error = \sum_{\forall p_i \in L} distance(p_i - plane)$$

In order to normalize the error and make it valid for comparison between models the above calculation was modified as follows:

$$Error = \frac{\sum_{\forall p_i \in L} distance(p_i - plane)}{N \cdot R}$$

Where:

N = Number of sampled points.

R = radius of the control sphere for a given scale.

The first factor:

$$\frac{\sum_{\forall p_i \in L} distance(p_i - plane)}{N}$$

gives the average distance of a point to the plane. Knowing that a point could be at most R far from the plane (since the plane is constrained to pass through the center of the control sphere and all points that are being

considered must be inside the control sphere), we normalized the average distance by dividing by R . Hence, the error is the ratio of the average distance to the largest possible distance.

3.2.3 The curvature function

The curvature value for a given point and a given scale is calculated based on the goodness of fit (GF) of the plane that best fits the marked set of voxels. The GF is calculated according to the following formula:

$$GF = w1 \cdot \frac{numMarked}{OptPlane} + w2 \cdot statFitPlane$$

Where:

$w1$: weight factor. $w1 \in [0,1]$

$w2$: weight factor. $w2 \in [0,1]$

numMarked: number of marked voxels (voxels that are intersected by the model). *numMarked* is always greater than or equal to 0.

OptPlane: number of voxels on a plane passing through the middle of the control sphere. This value is always greater than 0 and is set by the working scale.

statFitPlane: estimates a correlation coefficient for the set of points and a given plane. *statFitPlane* is normalized so it can only have values between 0 and 1.

The factor *statFitPlane* is calculated as described in the Section 3.2.2. Substituting *statFitPlane* into the previous equation yields:

$$GF = w_1 \cdot \frac{\text{numMarked}}{\text{OptPlane}} + w_2 \cdot \frac{\sum_{\forall p_i \in L} \text{distance}(p_i - \text{plane})}{N \cdot R}$$

$$\text{Curvature} = 1 - GF$$

Hence, the value of *GF* is a weighted sum of the goodness of fit of a plane (based on a statistical analysis) and the ratio of the number of voxels intersected to the number of voxels that should have been marked if the surface was a plane that passed through the center of the control sphere. The former term was added in order to avoid symmetry effects and detect special cases (for example peaks).

4 IMPLEMENTATION

The algorithm was implemented in C language. The program reads different polygonal models and then estimates curvature at the vertices of the polygonal mesh at different scales.

In a first stage the program calculates curvature by moving a control sphere from one vertex to another and voxelizing the space this control sphere occupied. Later on this was optimized by implementing a voxelization of the whole space occupied by the given model. Then, the control sphere was used as a way to obtain the desired scale (bigger control sphere means larger scale). Further optimization involved using an octree for the mentioned space voxelization.

The main functions implemented were: voxelization of the model's space using an octree; sampling of the model to obtain curvature values; and calculation of the goodness of fit (1-curvature) at a given point on the model.

The program proceeds as follow:

- 1- Voxelization of the 3-D space: A first bounding box (cube) is calculated for the model and it is set to be the first node of the octree (root). The subdivision proceeds by checking if the current node is intersected by the model, if so, then this cube is subdivided into 8 sub-cubes. The new centers for the children cubes are calculated. If these new centers intersect the polygonal model and the level of subdivision is not the maximum one, keep recursively subdividing. If a cube does not intersect the model then stop subdividing that branch and that is a leaf node. All intersecting nodes are marked and the points of intersection are stored.

- 2- Sampling of the model: The vertices are used as samples. The control sphere is centered at each of the model's vertex and the best fit plane is found for each location.

- 3- Calculation of curvature for the different samples at different scales:

The curvature measure is computed as described in Section 3.2.3 for a given point p (sampled point) and a given scale (set by the radius of the control sphere R) by estimating the plane that best fits a given set of point. This set of points is the list L of all voxels that have been marked as intersecting the model and whose distance to p is smaller than or equal to R .

The values for the weight coefficients, w_1 and w_2 , are estimated by trial and error in such a way that a plane would give a perfect fit and a peak will not.

GF is computed for every sampled point (vertex of the model). The values of the goodness of fit (1-curvature) and the plane coefficients are stored.

In order to visualize the curvature changes, GF values are associated with a different intensity of color and different scales are associated with different colors. So after GF has been calculated for every vertex of the model, the model is display as a shaded solid and the color of each vertex is proportional to GF .

5 RESULTS

The program was tested for different models. Figure 5.1 through 5.4 show the goodness of fitness (GF) obtained for a cube, a cube with a subdivided face, a bottle and a sphere model. The abscissa represents the vertex we are dealing with and the ordinate the goodness of fit obtained for that point. The models were normalized so that they could be bounded by a unit cube. The radius of the control sphere was set to 0.1. For the models that are shown here w_1 was set to 0 and w_2 was set to 1. Hence, we can evaluate the accuracy of the plane fitting algorithm to detect curvature changes.

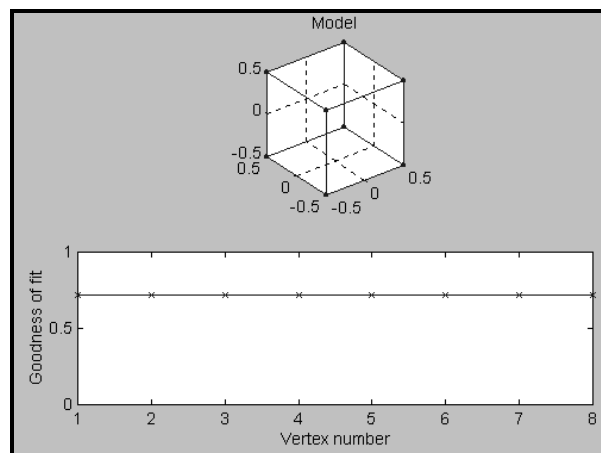


Figure 5.1 - Goodness of fit for the vertices of a cube model.

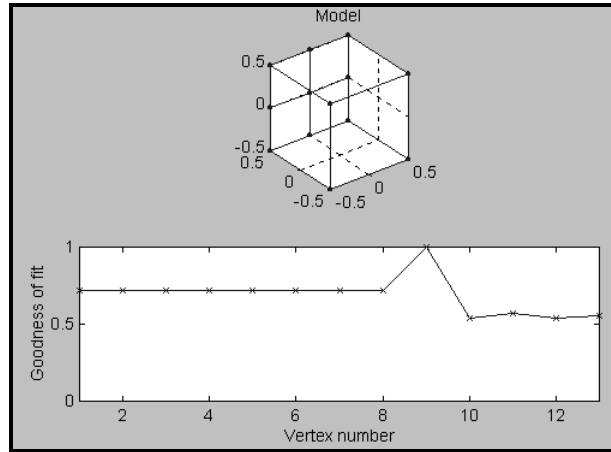


Figure 5.2 - Goodness of fit for the vertices of a subdivided cube model.

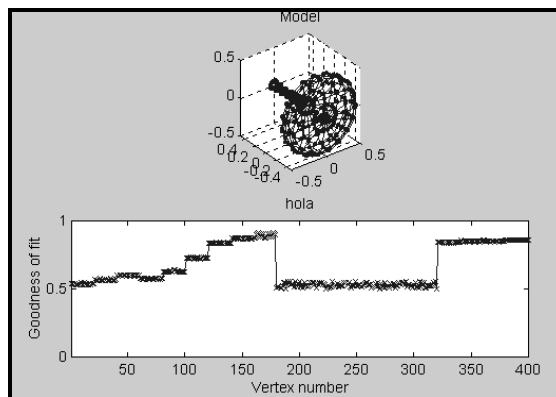


Figure 5.3 - Goodness of fit for the vertices of a bottle model.

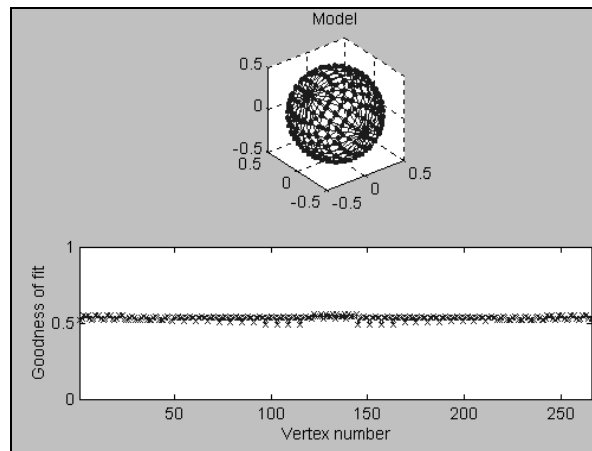


Figure 5.4 - Goodness of fit for the vertices of a sphere model.

The values obtained for the case of a cube are all constant. That is what we expected. For the cube with 5 extra vertices we can see that a perfect fit (null curvature) was obtained for the vertex lying in the middle of one face. The same curvature value was obtained for the vertices of the corners and similar curvature values were obtained for the 4 vertices that split the edges of one face.

For the sphere the goodness of fit of the planes varied within a narrow range. Here, we can notice that ideally a sphere should have constant curvature values. For polygonalized models this is not true, since the sphere is approximated by polygons. The fewer polygons used in the approximation the more changes in curvature we can find. For the model used, our results for the curvature measure can be interpreted as a model with constant curvature.

For the case of the bottle we can see areas of high goodness of fit (neck) and areas of smaller goodness of fit (sides). A region with high goodness of fit (low curvature) was obtained for the area of the base of the bottle. This is true since the base is almost planar.

Curvature was also computed for different scales. Table 5.1 shows the results of running the program for 3 different scales using the sphere model.

Scale	Mean <i>GF</i>	Std. Dev.
R=0.1	0.4780	0.0166
R = 0.01	0.9998	0.0005
R=0.001	1.00	0

Table 5.1 – Mean values of *GF* (1- curvature) at 3 different scales for the sphere model

We can see that when the radius of the control sphere *R* decreases *GF* increases. This is equivalent to say that curvature decreases. This means that for a very small scale the sphere model has a very low curvature (a person standing on the earth can perceive it as flat) and that as the scale increases the curvature does too.

6 DISCUSSION

Our new algorithm showed to reflect curvature changes at different scales. Thus it gives us an extrinsic measure of a polygonal model's shape. We can think of this measure as being an average normal to the point we have sampled. Thus, we can relate it to the Mean curvature that tells us how much our dihedral angle is for a given step.

The curvature measure presented here has several potential applications. It could be used in the LOD generation as well as LOD management, in model comparison to extract similarities and differences between models and also in the design and visualization area.

This way of measuring curvature would be very convenient for the simplification by areas, i.e. sub-regions at different scales and to identify regions whose curvature lies within a certain bound.

For LOD generation the new curvature measure could be combined with many of the algorithms described in Section 2. For example, it could be combined with Schroeder's algorithm [1] to contribute in the decision of which vertex to delete. This decision is based on the type of vertex and on the distance to an average plane (calculated using normals and areas of triangles). This could be substituted or complemented with our curvature measure and extended to consider different curvature scales.

Turk's [2] algorithm could be augmented with this curvature measure too. In this case, the curvature information would improve the algorithm by taking into consideration different curvature scales.

Rossignac and Borrell's [3] work already subdivides the space in a regular form (voxelizes the space). For this reason, including the curvature measure would involve very little extra work. In this case the curvature would guide the space subdivision, indicating whether to keep subdividing or not based on the curvature change. It could also be used to measure the difference in curvature before and after a vertex is collapsed. If the difference is larger than a given tolerance then this vertex should not be collapsed.

Elk's work [6] already takes into account different scales (resolutions). It uses a distance metric that could be combined with our curvature measure to take its simplification decisions.

Hoppe's work [7] measures curvature with a distance metric and reduces the model based on an optimization of the set of possible simplifications. One use of our curvature information could be in the optimization process. Curvature at a simplified vertex would be compared to that of the original vertex, and that at a larger scale.

Cohen's algorithm [8] could be extended to use the curvature measure. This algorithm uses the envelopes to guide in the simplification algorithm. Curvature would also be taken into account for the simplification decision.

Finally, Limstrom's work [9] could also be combined with this curvature measure. In this case the decision of which edge to collapse is based on trying to preserve volume and surface quantities. Curvature could be another property to preserve.

Another possible use of this curvature measure could be for comparison of different models. Since the model's space is voxelized we could arbitrarily sample this 3-D space and calculate curvature for the sampled voxels. We can then compare the values obtained for corresponding voxels of the two models being tested.

Translations and rotations of the models would affect this approach. A way to make it invariant to this kind of distortions could be by constructing an adjacency graph that relates curvature values and space location. A measure of the degree of similarity of the two models could be extracted by comparing these two graphs.

The curvature measure could also be useful in design problems. By visualizing curvature (i.e. associating different curvature values with different colors) shape information could be extracted and a better 3-D analysis could be achieved. Thus the effectiveness of the manufacture and analysis processes could be increased.

7 CONCLUSIONS AND FUTURE WORK

We have presented a new algorithm to measure curvature on a polygonal model at multiple scales. We have applied the algorithm to different models. The values obtained reflected the model's shape and the working scale. We have also discussed possible applications of this technique.

There are several directions to extend our work. A straightforward extension would be to combine the curvature measure presented here with some of the existing mesh decimation algorithms, and to compare the efficiency of the resulting LOD management algorithms with the existing ones.

Our curvature measure could also be used to compare polygonal models. We are currently working on generating a sampling scheme in order to perform these comparisons.

8 REFERENCES

1. Schroeder, W. & Zarge, J. & Lorensen, W. "Decimation of triangle meshes". SIGGRAPH '92. July 1992. (pp. 65-70)
2. Turk, G. "Re-tiling polygonal surfaces". SIGGRAPH '92. July 1992. (pp. 55-64)
3. Rossignac, J. & Borel, P. "Multi-resolution 3D Approximations for rendering". IBM Research RC17697. 1992.
4. Hinker, P. & Hansen, C. "Geometric Optimization". Proceedings Visualization '93. 1993. (pp. 189-195)
5. Hamann, B. "A data reduction scheme for triangulated surfaces". Computer Aided Geometry Design. 1994. (pp. 197-214)
6. Eck, M. & DeRose, T. & Duchamp, T. & Hoppe, H. & Lounsbery, M. "Multiresolution analysis of arbitrary meshes". SIGGRAPH '95. (pp. 173-182)
7. Hoppe, H. "Progressive Meshes". SIGGRAPH '96. August 1996. (pp. 99-108)
8. Cohen, J. & Varshney, A. & Manocha, D. & Turk, G. *et al.* "Simplification envelopes". SIGGRAPH '96. Aug. 1996. (pp. 119-128)
9. Lindstrom, P. & Turk, G. "Fast and Memory Efficient Polygonal Simplification". Georgia Institute of Technology. 1998.
10. Koenderik, J. "Solid shape". MIT Press. 1990.
11. Bloch, E. "A First course in geometric topology and differential geometry". Birkhouser. 1997.
12. Dill, J. "An Application of color Graphics to the display surface curvature". SIGGRAPH '81. 1981. (pp. 153-161)
13. Hamann, B. "Curvature Approximation for Triangulated Surfaces". Computing Suppl. 8. 1993. (pp. 139-153)

14. Elber, G. & Cohen, E. "Second Order Surface Analysis Using Hybrid Symbolic and Numeric Operators". University of Utah. 1997.
15. Interrante, V. "Illustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution". SIGGRAPH '97. 1997. (pp. 109-116)
16. Watson, B. "Level of Detail Generation and Management". Ph.D. proposal. Georgia Institute of Technology. 1997.
17. Tampieri, F. "Newell's Method for computing the plane equation of a polygon". Graphics Gems V. 1995. USA.