FITTING SMOOTH SURFACES TO SCATTERED 3D DATA USING PIECEWISE QUADRATIC APPROXIMATION

O.M. van Kaick M.V.G. da Silva W.R. Schwartz H. Pedrini

Federal University of Paraná, Computer Science Department, 81531-990, Curitiba-PR, Brazil

ABSTRACT

The approximation of surfaces to scattered data is an important problem encountered in a variety of scientific applications, such as reverse engineering, computer vision, computer graphics, and terrain modeling. This paper describes an automatic method for constructing smooth surfaces defined as a network of curved triangular patches. The method starts with a coarse mesh approximating the surface through triangular elements covering the boundary of the domain, then iteratively adds new points from the data set until a specified error tolerance is achieved. The resulting surface over the triangular mesh is represented by piecewise polynomial patches possessing C^1 continuity. The method has been implemented and tested on a number of real data sets.

1. INTRODUCTION

The increasing power of computer systems, associated with efficient geometric and modeling algorithms, are now capable of producing models with high degree of detail. On the other hand, the availability and complexity of such models are growing rapidly due to the improved capabilities for collecting and distributing data, and due to the need for higher accuracy. Examples of applications involving large volumes of data include remote sensing, planetary exploration, computer vision, computer-aided design, and medical image analysis.

Careful design of models and algorithms is necessary in order to make tasks more computationally tractable. Maintaining redundant information or using trivial analysis procedures may demand computational requirements far from of current or anticipated hardware capabilities. One of the most critical research problems encountered in the analysis and visualization of large volumes of data is the development of methods for storing, manipulating, and rendering massive data sets efficiently. Unless data reduction or compression methods are used, extremely large data sets cannot be analyzed or visualized in real time.

A common method for approximating surfaces uses a regular grid structure, in which a set of sampled points is stored at regular intervals. The main disadvantage of this representation is its inherent spatial invariability, since the structure is not adaptive to the irregularity of the data. Alternatively, triangular meshes use an irreuglar tessellation for storing the data points, where the structure of the mesh can be adjusted to reflect the density of the data. Consequently, cells become larger where data are sparse, and smaller where data are dense.

Polygonal surfaces are frequently used to represent three-dimensional data sets in several scientific areas, due mainly to their simplicity and flexibility. They are supported by the vast majority of modeling and rendering packages, and polygonal surface data are widely available. Hardware support for polygon rendering is also becoming more popular. In the last few years, several polygonal surface simplification algorithms have been proposed in the literature to generate a surface containing fewer polygons. This is important for processing, visualizing, or transmitting larger surface data sets than the available capabilities of software, computers, and networks permit.

Although this piecewise linear approximation is simple in concept and generates compact, accurate surfaces, the generalization to a piecewise smooth representation is a natural and, in many cases, a necessary extension. Certain regions of interest may consist of smoothly curved areas that meet along sharp curves. Modeling such regions as piecewise linear surfaces (C^0-continuity surfaces) requires a large number of triangles, whereas a curved surface can provide a more accurate and compact model of the true surface. Smooth surfaces can also produce superior results for rendering purposes, reducing certain perceptual problems such as the appearance of Mach Bands along element boundaries.

On the basis of the considerations mentioned above, this paper describes an automatic procedure for construction of smooth surfaces defined as a network of curved triangular patches. The approximating surface is represented by a C^1 piecewise triangular Bézier surface. High-order elements are generally more expensive to evaluate, therefore, they should be used only where the error is high and the extra effort is justified.

Section 2 gives a brief review of some relevant surface approximation methods found in the literature. In Section 3, the proposed method is presented in details. Some experimental results are given in Section 4. Finally, some discussions and conclusions are summarized in Section 5.

2. PREVIOUS WORK

Piecewise-polynomial models are a common alternative to polygonal models. By using higher-order polynomials, smooth surfaces can be approximated more accurately than with planar polygons. A model composed of piecewise-polynomial elements discretizes the model into a fixed set of patches. For certain applications, however, a fixed set of surface elements may not be appropriate. Several adaptive methods have been developed, such as subdivision surfaces [1, 2, 3], hierarchical splines [4], and models composed of triangular patches [5, 6].

In 1974, Chaikin [7] introduced a method for generating a smooth curve from a control polygon by recursively cutting off the corners of the polygon. This is perhaps the first method of constructing smooth curves of arbitrary topological type. Catmull and Clark [1] and Doo and Sabin [2] generalized the idea to surfaces. In such schemes, a subdivision surface is defined by repeatedly refining an initial control mesh M^0 to produce a sequence of meshes M^1, M^2, ... that converge to a *limit* surface.

The Loop scheme [3] is probably the simplest subdivision method for triangular meshes. Each edge of the mesh is split into two, and new vertices are reconnected to form four new triangles. Vertices are rearranged by using an averaging step. Figure 1 shows this subdivision procedure.

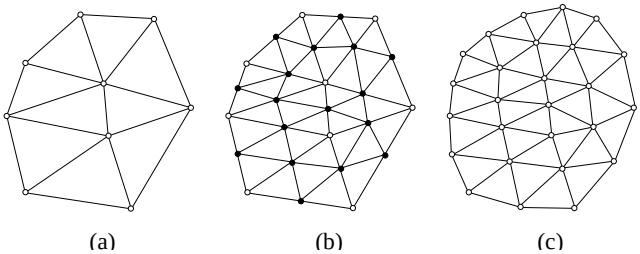

Fig. 1. Loop subdivision scheme. (a) initial mesh; (b) new mesh created by the splitting step (new vertices are shown as black dots); (c) mesh created by the averaging step.

The Butterfly scheme, proposed by Dyn *et al.* [8], is a subdivision scheme that recursively divides each triangular face of the control polygon into four triangular faces interpolating the old control points. Like other interpolating schemes, the subdivision step retains the existing vertices and splits each edge segment at its midpoint. The butterfly scheme is proven to achieve tangent plane continuity when applied to regular meshes. A modification of the Butterfly scheme was proposed by Zorin [9], which guarantees that the scheme produces C^1-continuous surfaces for arbitrary meshes.

3. SMOOTH SURFACE APPROXIMATION

The first step of our method is to generate a coarse piecewise linear approximation of the surface by creating triangular elements at the data points. The algorithm starts with an initial triangulation that covers the boundary of the domain, and iteratively adds new points from the data set until a specified error tolerance is achieved. The resulting surface is formed by C^0 continuous triangular patches.

The Delaunay triangulation is used to construct the mesh, generating the triangulation that maximizes the minimum angle of all triangles. This helps to reduce the occurrence of thin and long triangles since they can lead to undesirable behavior, affecting numerical stability and producing visual artifacts.

Once the initial triangulation has been generated, we can construct smooth surfaces over the triangular mesh using piecewise polynomial patches. Our interpolation method uses quadratic polynomials to construct C^1 surfaces, which is based on a scheme originally described by Powell and Sabin [10] and further developed by Cendes and Wong [11]. It divides each triangle into subtriangles and fits an approximating function over each subtriangle. Certain continuity conditions must be satisfied at every boundary between two patches in order for the entire surface to be smooth. Since the method requires only elevation and derivative values at the vertices of the triangulation, such approach is relatively simple to implement.

Before describing the interpolation method, some preliminary concepts are introduced.

3.1. Mathematical Preliminaries

The use of *barycentric coordinates* is a natural way of representing triangular patches, since this guarantees a symmetric influence of all three triangle corners. Let T be a planar triangle defined by the vertices V_1, V_2, V_3. Any point V in T can be expressed in terms of the barycentric coordinates (r, s, t) defined by $V = rV_1 + sV_2 + tV_3$, where $r + s + t = 1$ and $0 \leqslant r, s, t \leqslant 1$.

Bernstein polynomials of degree n over a triangle T can be defined in terms of barycentric coordinates (r, s, t) expressed as

$$B^n_{i,j,k}(r, s, t) = \frac{n!}{i!\ j!\ k!}\ r^i s^j t^k$$

which form a basis for all bivariate polynomials of degree n.

The parametric equation for a single triangular Bernstein-Bézier patch is

$$\mathbf{p}(r, s, t) = \sum_{\substack{i+j+k=n \\ i,j,k \geqslant 0}} \mathbf{b}_{i,j,k}\ B^n_{i,j,k}(r, s, t) \qquad (1)$$

where the coefficients $\mathbf{b}_{i,j,k}$ are called the Bézier control points of $\mathbf{p}(r, s, t)$.

Taking $n = 2$, Equation 1 gives

$$\mathbf{p}(r, s, t) = r^2\ \mathbf{b}_{2,0,0} + s^2\ \mathbf{b}_{0,2,0} + t^2\ \mathbf{b}_{0,0,2} + \\ + 2rs\ \mathbf{b}_{1,1,0} + 2rt\ \mathbf{b}_{1,0,1} + 2st\ \mathbf{b}_{0,1,1}$$

Figure 2 shows an example of a triangular patch and its corresponding Bernstein polynomials.

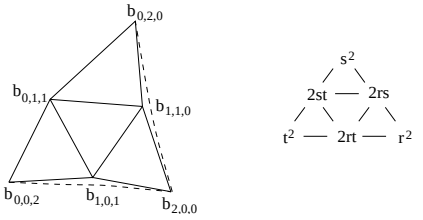

Fig. 2. Quadratic triangular Bernstein-Bézier patch and its corresponding Bernstein polynomials.

Farin [5] provides a comprehensive description of the conditions for derivative continuity on the common boundary between two adjacent triangular patches. To ensure C^1 continuity, the first derivatives of two adjacent patches $\mathbf{p}$ and $\mathbf{q}$ must join continuosly across the shared edge.

Our interpolation method computes the coefficients of the polynomial for each triangle based only on the elevation values and the estimated values of the first partial derivatives (tangent vectors) at the three vertices of the triangle. The derivative at a vertex is computed as the weighted average of tangents of the triangles adjacent to the vertex.

Given a data point (x_i, y_i), the n triangles surrounding this vertex are found. Then, the cross product of any pair of sides from each triangle is computed. This three-dimensional vector is perpendicular to the triangle, and its length is twice the area of the triangle for more details). For example, the cross product vector (x_p, y_p, z_p) for a triangle defined by the points $P_1(x_1, y_1, z_1)$,

$P_2(x_2, y_2, z_2)$, and $P_3(x_3, y_3, z_3)$ is given by

$$x_p = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1)$$
$$y_p = (z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1)$$
$$z_p = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)$$

Since the triangle vertices are ordered in counterclockwise direction in our implementation, values z_p are always positive. This agrees with the fact that these values represent a height component.

To estimate the derivative at a data point, the cross products are accumulated for all triangles involving that point, and the resulting normals at the vertex are averaged. The estimated gradient plane can be thought of as a spatial average of slopes and sizes of the surrounding triangles.

3.2. Quadratic Interpolant

The Powell and Sabin [10] scheme subdivides each triangular element into six subtriangles to generate smooth surfaces, as shown in Figure 3. The points P_2, P_4, and P_6 are the vertices of the triangle. The point P_o is the inscribed center of the triangle and the points Q_o, Q_1, and Q_2 are the inscribed centers for the adjacent triangles. The six subtriangles are formed by connecting the point P_0 with the triangle vertices and with the three points Q_i. Boundary triangles are subdivided into six by adding one point at an arbitrary location along each triangle boundary. In our algorithm, the midpoint of the edge on the boundary of the mesh is used.

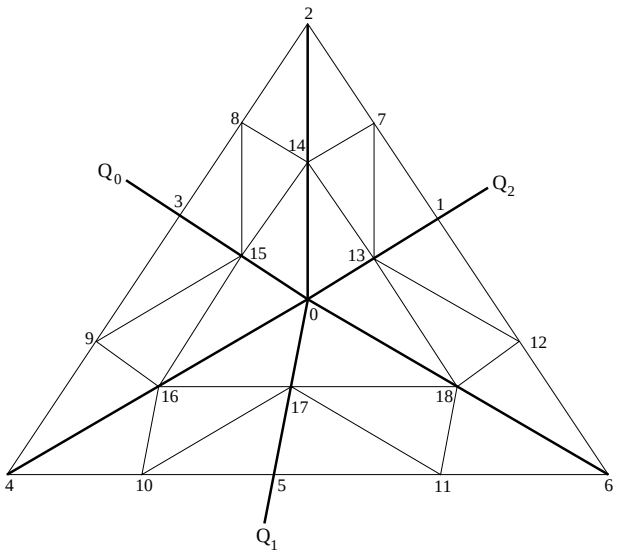

Fig. 3. Powell-Sabin subdivision.

The interpolation requires only function and derivatives values at each data vertex. The quadratic surface is calculated for each subtriangle by evaluating the six control points and the barycentric coordinates relative to the subtriangle. Then, for the subtriangle $P_0P_3P_4$, the C^1 quadratic surface can be represented as

$$\mathbf{p}(r,s,t) = r^2\ \mathbf{b}_0 + s^2\ \mathbf{b}_3 + t^2\ \mathbf{b}_4 + \\ + 2rs\ \mathbf{b}_{15} + 2rt\ \mathbf{b}_{16} + 2st\ \mathbf{b}_9$$

where (r, s, t) are the barycentric coordinates of a point (x, y) in the subtriangle and the six coefficients denote the data values at the control points.

In order to satisfy the coplanarity conditions in the Bézier triangle scheme, we need to specify data values and derivatives at the 19 control points. The vertex values at P_2, P_4, and P_6 are already specified. The derivative values at the vertices, denoted as z_i^x and z_i^y, $i = 2, 4, 6$, are estimated by using the technique described above.

Each of the six subtriangles i in Figure 3 is conveniently identified by the indices of the control points given by 0, $12+i$, $13+i$ mod 6, i, $6+i$, and $1+i \mod 6$, where $1 \le i \le 6$.

The 19 Bézier control points b_i, $0 \le i \le 18$, are defined as follows

$$b_0 = r\ b_{14} + s\ b_{16} + t\ b_{18} \qquad b_1 = \beta b_{12} + (1-\beta) b_7$$
$$b_2 = z_2 \qquad b_3 = \gamma b_8 + (1-\gamma) b_9$$
$$b_4 = z_4 \qquad b_5 = \alpha b_{10} + (1-\alpha) b_{11}$$
$$b_6 = z_6 \qquad b_7 = b_2 + c_3 z_1^x + d_3 z_1^y$$
$$b_8 = b_2 + c_4 z_1^x + d_4 z_1^y \qquad b_9 = b_4 + c_5 z_2^x + d_5 z_2^y$$
$$b_{10} = b_4 + c_6 z_2^x + d_6 z_2^y \qquad b_{11} = b_6 + c_7 z_3^x + d_7 z_3^y$$
$$b_{12} = b_6 + c_8 z_3^x + d_8 z_3^y \qquad b_{13} = \beta b_{18} + (1-\beta) b_{14}$$
$$b_{14} = b_2 + c_0 z_1^x + d_0 z_1^y \qquad b_{15} = \gamma b_{14} + (1-\gamma) b_{16}$$
$$b_{16} = b_4 + c_1 z_2^x + d_1 z_2^y \qquad b_{17} = \alpha b_{16} + (1-\alpha) b_{18}$$
$$b_{18} = b_6 + c_2 z_3^x + d_2 z_3^y$$

where the coefficients c_i, $0 \le i \le 8$, are defined as

$$c_0 = \frac{1}{2}(x_0 - x_2) \qquad c_1 = \frac{1}{2}(x_0 - x_4)$$
$$c_2 = \frac{1}{2}(x_0 - x_6) \qquad c_3 = \frac{\beta}{2}(x_6 - x_2)$$
$$c_4 = \frac{1-\gamma}{2}(x_4 - x_2) \qquad c_5 = \frac{\gamma}{2}(x_2 - x_4)$$
$$c_6 = \frac{1-\alpha}{2}(x_6 - x_4) \qquad c_7 = \frac{\alpha}{2}(x_4 - x_6)$$
$$c_8 = \frac{1-\beta}{2}(x_2 - x_6)$$

The coefficients d_i, $0 \le i \le 8$, are defined similarly by substituting y_j for x_j in the above equations. Finally, α, β, and γ represent the distance ratios defined as

$$\alpha = \frac{\overline{P_6P_5}}{\overline{P_6P_4}} \qquad \beta = \frac{\overline{P_2P_1}}{\overline{P_2P_6}} \qquad \gamma = \frac{\overline{P_4P_3}}{\overline{P_4P_2}}$$

where $\overline{P_iP_j}$ is the distance between nodes i and j.

4. EXPERIMENTAL RESULTS

Our method has been tested and evaluated on several synthesized and real objects in order to demonstrate its performance. The algorithms were implemented in C++ programming language on a PC Pentium III with 866 MHz and 1 Gbyte of main memory.

The linear interpolation method is based on an incremental Delaunay triangulation algorithm [12], whereas the quadratic method is based on the Powell-Sabin subdivision scheme described above.

Figure 4 shows three sets of images obtained by applying our linear and quadratic interpolation method. The same number of

data points are retained in both methods. Table 1 reports the average root mean square (RMS) error for each reconstructed object.

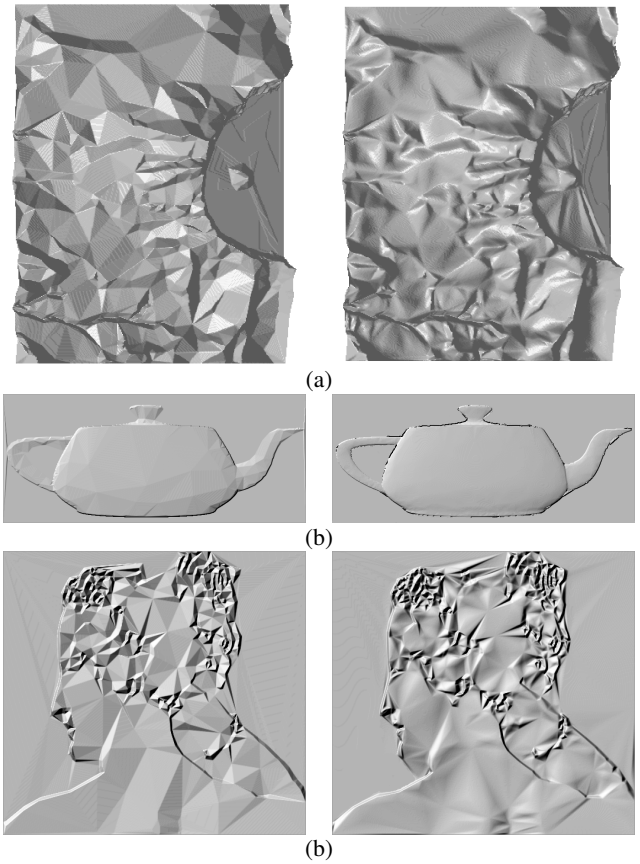

Fig. 4. Approximations of three data sets by the linear and quadratic method, respectively. (a) Crater Lake DEM (336×459); (b) Teapot (751×321); (c) Paolina (480×512).

Object	# of Points	Linear	Quadratic
Crater Lake	3,011	6.76	5.87
Teapot	4,651	3.21	2.89
Paolina	4,910	3.54	3.22

Table 1. Summary of results for reconstructed objects.

Besides resulting in smoother surfaces, the quadratic interpolation presented a lower overall error measurement, which is based on the maximum difference between the actual data values and the surface approximation. Therefore, the method can also be used to reduce the number of points required to construct a surface from scattered data within a specified approximation accuracy.

5. CONCLUSIONS

This paper describes a method for constructing smooth surfaces by approximating scattered 3D data points. The resulting surface over the triangular mesh is represented by piecewise polynomial patches possessing C^1 continuity. A Delaunay triangulation algorithm is used to produce a coarse mesh, which is refined until a specified error tolerance is achieved. Once this initial triangulation has been generated, smooth surfaces are constructed over the triangular mesh using piecewise polynomial patches.

Experimental results have demonstrated that our technique is effective for modeling dense sets of scattered 3D points, producing approximations with high quality and efficient data reduction.

As a future work, the use of more sophisticated methods for estimating derivatives at the vertices of each triangle may produce better results, since the calculation of the coefficients of the polynomial depend on precise values for derivatives.

6. REFERENCES

[1] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer-Aided Design*, vol. 10, no. 6, pp. 350–355, Nov. 1978.

[2] D. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points," *Computer-Aided Design*, vol. 10, no. 6, pp. 356–360, Nov. 1978.

[3] Charles Loop, "Smooth subdivision surfaces based on triangles," M.S. thesis, University of Utah, Department of Mathematics, 1987.

[4] David R. Forsey and Richard H. Bartels, "Surface fitting with hierarchical splines," *ACM Transactions on Graphics*, vol. 14, no. 2, pp. 134–161, Apr. 1995.

[5] Gerald Farin, *Curves and Surfaces for Computer-Aided Geometric Design - A Practical Guide*, Academic Press, Inc, 1992.

[6] Charles L. Lawson, "Software for C^1 surface interpolation," in *Mathematical Software III*, John R. Rice, Ed., pp. 161–194. Academic Press, New York, 1977.

[7] George Merril Chaikin, "An algorithm for high-speed curve generation," *Computer Graphics and Image Processing*, vol. 3, no. 4, pp. 346–349, Dec. 1974.

[8] Nira Dyn, David Levin, and J. A. Gregory, "A Butterfly subdivision scheme for surface interpolation with tension control," *ACM Transactions on Graphics*, vol. 9, no. 2, pp. 160–169, Apr. 1990.

[9] Denis Zorin, Peter Schröder, and Wim Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *SIGGRAPH'96 Conference Proceedings*, New Orleans, Louisiana, USA, Aug. 1996, Annual Conference Series, pp. 189–192.

[10] M. J. D. Powell and M. A. Sabin, "Piecewise quadratic approximations on triangles," *ACM Transactions on Mathematical Software*, vol. 3, no. 4, pp. 316–325, Dec. 1977.

[11] Zoltan J. Cendes and Steven H. Wong, "C^1 quadratic interpolation over arbitrary point sets," *IEEE Computer Graphics and Applications*, vol. 7, no. 11, pp. 8–16, Nov. 1987.

[12] Hélio Pedrini, "An improved refinement and decimation method for adaptive terrain surface approximation," in *Proceedings of the 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Plzen, Czech Republic, 2001, pp. 103–109.