SHAPE REPRESENTATION

INTRODUCTION

Examples of digital objects are as follows:

- Space curves such as the trajectory of a robot arm
- Solid objects such as an engine block
- Surfaces such as the hood of a car
- Volumetric objects such as a magnetic resonance imaging (MRI) scan

Each class of objects is represented using different methods. These methods are addressed by a relatively new discipline, called CAGD (computer-aided geometric design also known as geometric modeling or geometric design), situated between mathematics and computer science. We will outline its basic themes and show how they apply to the task of shape representation.

Several books exist on the topic of CAGD, and they should be consulted for more details: Farin (1), Faux and Pratt (2), Gallier (3), Hoschek and Lasser (4). An undergraduate text is Reference 5. An authoritative collection of papers is in Reference 6.

THE SHAPE OF TRIANGLES

One of the most basic geometric objects is a triangle. There are many ways to judge the shape of a triangle. All aim at measuring closeness to an equilateral triangle. One such measure is as follows: Of all three angles in a triangle, consider the smallest one. The larger this smallest angle is, the better the shape of the triangle.

Now consider a set of 2-D points $\mathbf{x}_i$. A common problem is to connect them all by (nonoverlapping) triangles, which forms a *triangulation*. The most commonly used is the *Delaunay triangulation*. Although there are many possible triangulations of a point set, the Delaunay triangulation produces triangles of optimal shape in the following sense. There are many possible triangulations of the point set. In each triangulation, find the minimum angle of all angles formed by the triangles in the triangulation. We observe that a small minimum angle, say 1 degree, flags a triangle as badly shaped. Thus, it seems reasonable to find, for all triangulations, the one in which the *largest* minimum angle occurs. This triangulation is unique: It is the Delaunay triangulation. Figure 1 gives an example. An efficient algorithm for finding the Delaunay triangulation was given by Lawson (7). The boundary points of the Delaunay triangulation form the *convex hull* of the point set. Often the boundary triangles are badly shaped—to avoid this, *contrained Delaunay triangulations* are employed; see Reference 8.

The Delaunay triangulation has another interesting property: Consider any triangle, and form its circumcircle. Then it is guaranteed that no other data point $\mathbf{x}_i$ is inside this circle. For a recent book on triangulations, see Reference 9.

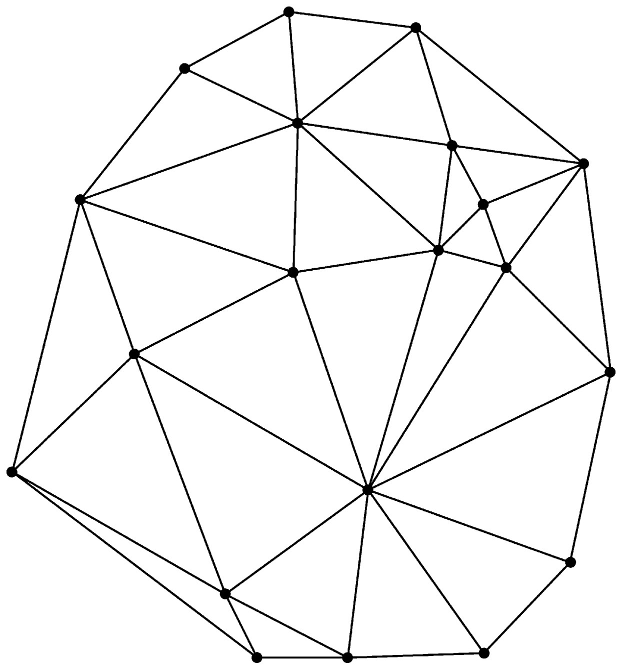

Figure 1. The Delaunay triangulation of a point set.

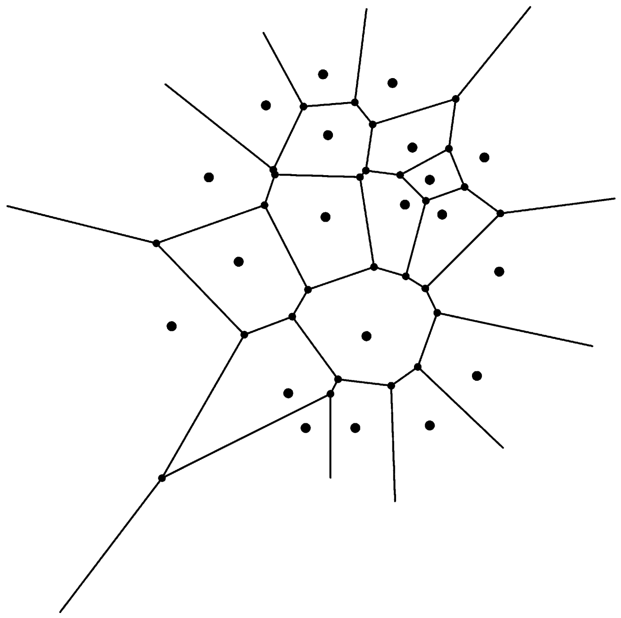

Figure 2. The Voronoi diagram corresponding to Fig. 1.

Another important structure associated with a set of points and their Delaunay triangulation is the *Voronoi diagram*. It is obtained by connecting the circumcenters of neighboring triangles by straight lines. If the triangles are on the convex hull of the point set, there will be infinite lines. See Fig. 2 for an illustration. For literature, see Reference (10).

Closely related to the Voronoi diagram is the *medial axis*. It represents shape by abstraction, namely by producing a skeleton of a closed polygon. If we approximate a closed polygon by a large set of sample points on it, we can compute their Voronoi diagram. By connecting all neigh-

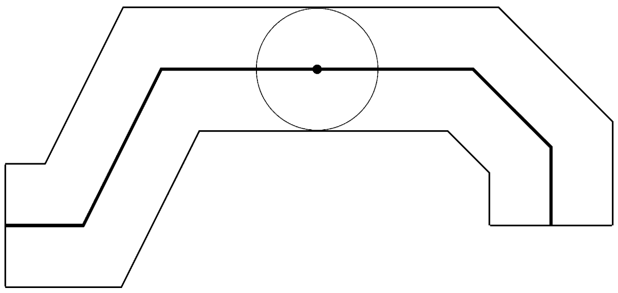

Figure 3. The medial axis of a closed polygon. One circle is shown that touches the polygon twice.

boring Voronoi vertices, we obtain the medial axis. A different (but equivalent) definition: Any interior point of a closed polygon is the center of a circle touching the polygon. Some interior points are centers of circles touching *two* points of the polygon. They form the medial axis. The medial axis originated in the life sciences (11) and is now used in many disciplines (12–14). For an illustration, see Fig. 3.

BÉZIER CURVES

Bézier curves originated in France in the early 1960s and were pioneered by P. de Casteljau (at Citroën) and P. Bézier (at Rénault). They are now ubiquitous in all of 2-D and 3-D modeling. They were invented to digitally model curves arising in the design of feature curves on car bodies, which departs from the classic method of using wooden template curves.

A Bézier curve of degree n is defined by

$$x(t) = \sum_{i=0}^{n} b_i B_i^n(t) \tag{1}$$

where the $B_i^n(t)$ are *Bernstein polynomials*.

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{2}$$

and the $\mathbf{b}_i$ are 2-D or 3-D *control points*.

Bézier curves have many important properties; we list some of them:

1. Endpoint interpolation: The curve passes through the polygon endpoints: $x(0) = b_0$ and $x(1) = b_n$.
2. Invariance under affine maps: In general, if an affine map is applied to the control polygon, then the curve is mapped by the same map. (We should mention that Bézier curves are *not* invariant under projective maps. If one wants this property, *rational* Bézier curves have to be used.)
3. Convex hull property: For $t \in [0,\ 1]$, the point $\mathbf{x}(t)$ is in the convex hull of the control polygon.
4. Linear precision: If the control points $\mathbf{b}_i$ are evenly spaced on the straight line between $\mathbf{b}_0$ and $\mathbf{b}_n$, then the Bézier curve is the linear interpolant between $\mathbf{b}_0$ and $\mathbf{b}_n$.
5. Variation diminution: No straight line (2-D) or plane (3-D) intersects the curve more often than the control polygon. This property accounts for the shape preservation property of Bézier curves.

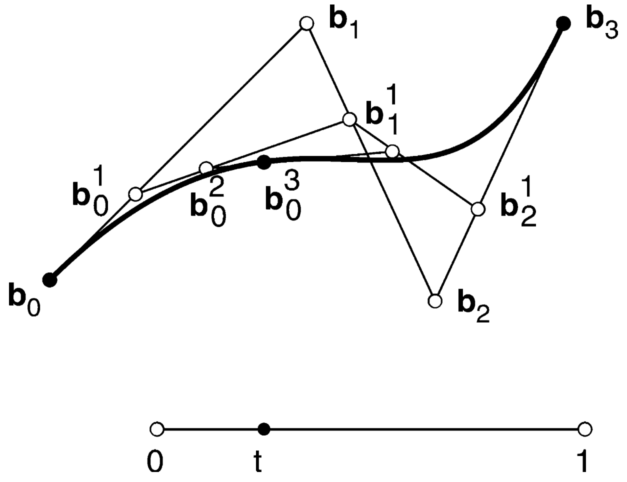

Figure 4. The de Casteljau algorithm for a cubic curve.

The derivative of a Bézier curve may again be written as a Bézier curve:

$$.x(t) = n \sum_{i=0}^{n-1} \Delta b_i B_i^{n-1}(t) \tag{3}$$

where $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$ are forward differences.

Bézier curves may also be evaluated using a recursive algorithm, the de Casteljau algoritflm.

de Casteljau algorithm:
Given: $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in IE^3$ and $t \in IR$,
set

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + t b_{i+1}^{r-1}(t) \quad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases} \tag{4}$$

and $\mathbf{b}_i^0(t) = \mathbf{b}_i$. Then $\mathbf{b}_0^n(t)$ is the point with parameter value t on the *Bézier curve* $\mathbf{b}^n$.

The polygon $\mathbf{P}$ formed by $\mathbf{b}_0, \ldots, \mathbf{b}_n$ is called the *Bézier polygon* or *control polygon* control polygon of the *Bézier curve* $\mathbf{b}^n$. Similarly, the polygon vertices $\mathbf{b}_i$ are called *control points* or *Bézier points*. As t varies between 0 and 1, each intermediate point $\mathbf{b}_i^r(t)$ traces out curves of degree r. They are illustrated in Fig. 4 for the case $n = 3$.

Figure 5 shows an example of a practical use of Bézier curves: About 70 points were computed to represent the shape of a propeller blade cross section and a degree five Bézier curve was fitted to them. Note that the data are somewhat noisy, so the fit is not very good.

B-SPLINE CURVES

Bézier curves are a great modeling tool, but they cannot cope with complex shapes very well. For those, the tool of choice includes *B-spline curves* of degree n. They have added flexibility by not using just one interval [0, 1], but rather a partition of an interval $[a,\ b]$ into subintervals. In this article, we restrict ourselves to the special nondecreasing knot sequence $a = u_0 = \ldots = u_{n-1},\ u_n, \ldots, u_{L+n-1},\ u_{L+n}, = \ldots = u_{L+2n-2} = b$. An example

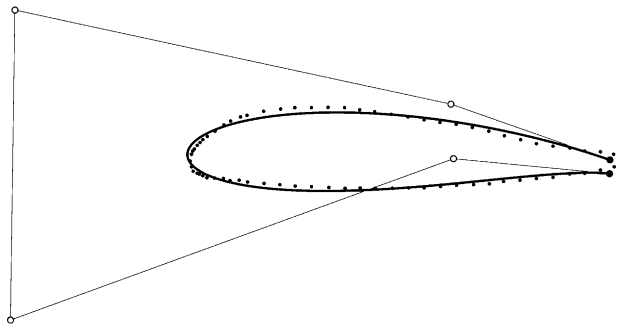

Figure 5. A quintic Bézier curve fit to data from a propeller blade.

for the cubic case $n = 3$ and $L = 2$: 0,0,0,1,1.5,2,2,2. We also demand that no more than n successive u_i are equal.

A B-spline curve is obtained by replacing the Bernstein polynomials B_i^n in Reference 15 by B-splines $N_i^u(u)$. They are *piecewise polynomial* functions and are typically defined recursively:

$$N_l^n(u) = \frac{u - u_{l-1}}{u_{l+n-1} - u_{l-1}} N_l^{n-1}(u) + \frac{u_{l+n} - u}{u_{l+n} - u_l} N_{l+1}^{n-1}(u) \tag{5}$$

The recursion is anchored by

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_{i-1} \le u < u_i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

A B-spline curve is then defined by

$$\mathrm{x}(u) = \sum_{j=0}^{L-n+1} \mathrm{d}_j N_j^n(u)$$

where the $\mathbf{d}_j$ are again called control points. A de Casteljau-like algorithm exists for their evaluation as well; it is known as the *de Boor* algorithm.

B-splines enjoy all the properties of Bézier curves; in addition, they have a *local control* property: If only one of the $\mathbf{d}_j$ is changed, then the curve will only change nearby and will remain unchanged everywhere else. This is due to the fact that each N_i^n has *local support*: $N_i^n(u) \equiv 0$ for $u \notin [u_{i-1}, u_{i+n}]$.

Bézier curves are a special case of B-spline curves. Over the special knot sequence $u_0 = \ldots = u_n = 0,\ u_{n+1}, \ldots, u_{2n+2} = 1$, the B-splines N_i^n are identical to the Bernstein polynomials B_i^n.

An advantage of B-spline curves over Bézier curves is that they may be used to form closed curves. For this, the knot sequence and the control points are extended to form periodic sequences. An example of a closed cubic B-spline curve is shown in Fig. 6.

B-spline curves may also be used for least-squares approximation in much the same way—iust replace the Bernstein basis by the B-spline one.

NURBS

Bézier curves or B-spline curves are capable of representing parabolas but not any other conic, such as ellipses or hyperbolas. This can be overcome by generalizing to *rational curves*. A rational B-spline curve (typically known

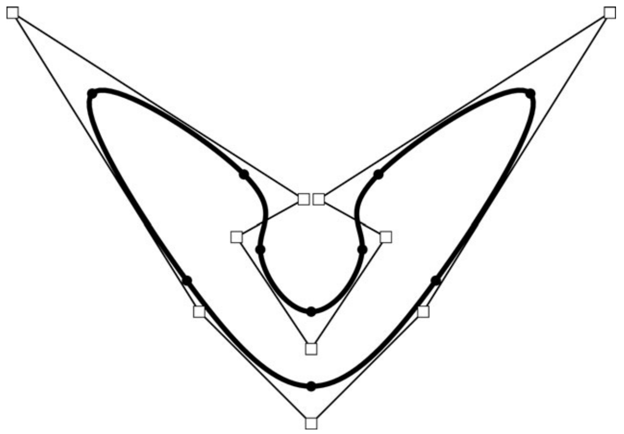

Figure 6. A closed cubic B-spline curve.

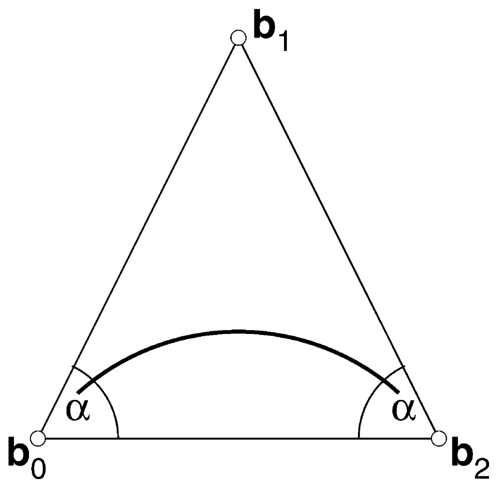

Figure 7. A segment of a circle in rational quadratic form.

as a NURB curve, for the non uniform rational B-spline curve) is given by

$$\mathbf{x}(u) = \frac{\sum_{j=0}^{L-n+1} w_j \mathbf{d}_j N_j^n(u)}{\sum_{j=0}^{L-n+1} w_j N_j^n(u)}$$

The scalar factors w_i are called *weights*. The effect of a large value for one w_i is that the curve will pull toward the control point $\mathbf{d}_i$.

Bézier curves are a special case of B-spline curves; the same is true for rational Bézier curves. We now show how to write a segment of a circle as a rational Bézier curve of degree two. Referring to Fig. 7, we see that $\mathbf{b}_0$ and $\mathbf{b}_2$ are the segment's endpoints and $\mathbf{b}_1$ is the intersection of the respective tangents. The circle segment is then written as

$$\mathbf{x}(t) = \frac{\mathbf{b}_0 B_0^2(t) + v\mathbf{b}_1 B_1^2(t) + \mathbf{b}_2 B_2^2(t)}{B_0^2(t) + vB_1^2(t) + B_2^2(t)}$$

where $v = \cos\alpha$, with α being the angle formed by $\mathbf{b}_0\mathbf{b}_1$ and $\mathbf{b}_0\mathbf{b}_2$.

For references, consult References 16–18

THE SHAPE OF CURVES

The main shape characteristic of curves is described by classic differential geometry—it is the concept of *curvature*. When a curve bends sharply, its curvature is high; where it is fiat, its curvature is low. Intuitively, curvature is then defined as follows: At a given point on the curve, find a circle that best fits the curve there. The inverse of this circle's radius is the curvature. Examples: A straight line has zero curvature; a circle has constant curvature.

Analytically, it can be shown that the curvature, denoted by $\kappa(t)$, is given by

$$\kappa(t)\frac{\|\dot{\mathbf{x}}\wedge\ddot{\mathbf{x}}\|}{\|\dot{\mathbf{x}}\|^3}$$

where dots denote first and second derivatives, respectively. A common way to judge the shape of a curve is by inspection of its *curvature plot*. This is the graph of $\kappa(t)$. If it consists of only a small number of monotone pieces, then the curve is said to be *fair*. Even if a curve appears acceptable judging from its display on a computer screen, it may badly fail a designer's intent. The curvature plot will detect any shape imperfections. An example is given in Fig. 11.

PARAMETERIC SURFACES

Curves axe important for 2-D processes such as font design. (For example, all letters in this book are designed as piecewise Bézier curves.) Yet most "real-life" applications, such as car design or scientific computing, require 3-D surfaces for modeling objects. Several kinds of surfaces exist; the most commonly used are tensor product B-spline surfaces, with tensor product Bézier surfaces as an important subset.

A Bézier surface is given by

$$\mathbf{x}(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{b}_{i,j}B_i^m(u)B_j^n(v) \tag{7}$$

This surface is a bipolynomial map of the 2-D unit square $0\le u,v\le 1$ (the domain) into the surfacés 3-D range. The *control points* $\mathbf{b}_{i,j}$ form the *control net* of the surface. The boundaries of the control net are Bézier polygons of the patch's boundary curves.

If we replace the Bernstein polynomials by B-splines, we obtain a B-spline surface:

$$\mathbf{x}(u)=\sum_{i=0}^{L-n+1}\sum_{j=0}^{M-m+1}\mathbf{d}_{i,j}N_i^n(u)N_j^m(v) \tag{8}$$

This surface is again defined by a set of control points, the *control net*, and by two sets of knot sequences, one each for the u-and v-direction. Bézier surfaces are *one* polynomial surface, whereas B-spline surfaces consist ot many such patches. An example of a B-splirre surface is shown in Fig. 8 .

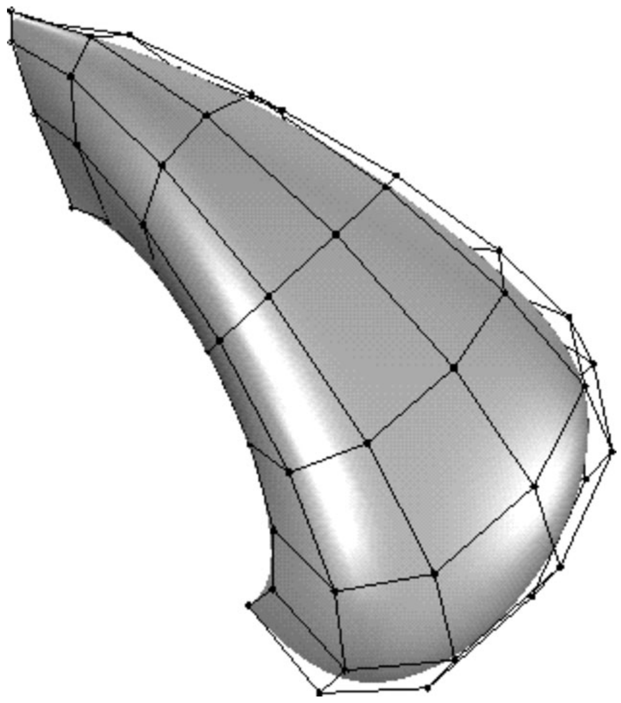
Figure 8. A B-spline surface and its control net.

SUBDIVISION SURFACES

Although B-spline and NURB surfaces are frequently employed in the area of CAD/CAM, the emerging field of computer animation has adopted *subdivision surfaces* as the shape representation of choice. These are defined by recursive algorithms that, starting from a coarse polygon, converge to a smooth limit surface. Here, we consider one of the more popular schemes: *Loop subdivision*, devised by C. Loop (19). Its input is a coarse triangle mesh as discussed and proceeds as follows.

1) Form edge points e_j^{i+1}: Assuming that v_1^i and v_2^i are the endpoints of an edge in the mesh and that v_3^i, and v_4^i are the remaining vertices of the two triangles sharing the edge, set

$$e_j^{i+1}=\frac{3}{8}(v_1^i+v_2^i)+\frac{1}{8}(v_3^i+v_4^i) \tag{9}$$

This process is easily visualized using a *mask*, shown in Fig. 9. The shown coefficients have to be multiplied by a factor of 1/8.

2) For each vertex $\mathbf{v}^i$ in the mesh, form a new vertex point v^{i+1}. Assuming $\mathbf{v}^i$ has n neighbors $v_1^i,\ldots,v_n^i$, it is computed as follows:

$$v^{i+1}=(1-n\alpha)v^i+\alpha\sum_{j=a}^{n}vv_j^i \tag{10}$$

where

$$\alpha=\frac{1}{n}(\frac{5}{8}-(\frac{3}{8}+\frac{1}{4}\cos\frac{2\pi}{2})^2) \tag{11}$$

for $n>3$ and $\alpha=\frac{3}{16}$ if n = 3.

3) Form new triangles.

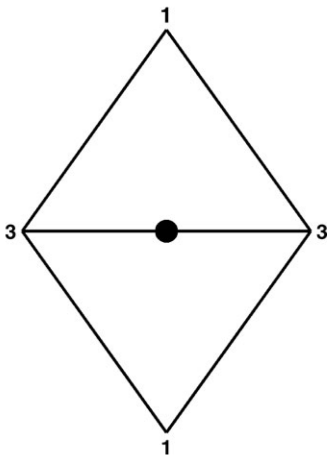

Figure 9. Loop subdivision: the edge point mask.

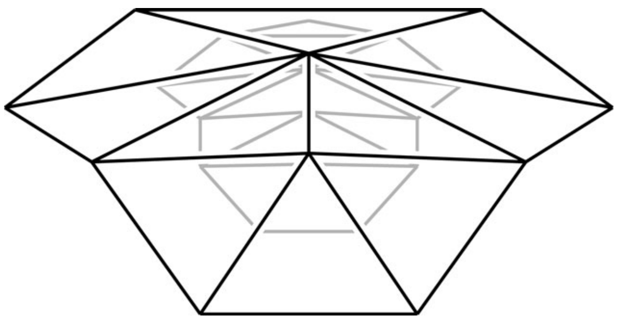

Figure 10. Loop subdivision: the original control net (black) and one level of subdivision (gray).

Figure 10 illustrates.

Now consider the mesh that is obtained after some number k of iterations. Select any vertex $\mathbf{v}$ of it. This vertex will converge to a point $\mathbf{v}^\infty$ on the limit surface. Using an eigenvalue analysis one can show that this limit point is given by

$$\mathrm{v}^\infty = \frac{3+8\alpha(n-1)}{3+8n\alpha} + \frac{8\alpha}{3+8n\alpha}\sum_{j=1}^{n} \mathrm{v}_j \tag{12}$$

where the $\mathbf{v}_j$ are the neighbors of $\mathbf{v}$ in the mesh obtained after k iterations. For the special case $k = 0$, equation 16 gives the limit points corresponding the original mesh vertices.

For more details on the Loop and other subdivision schemes, consult Reference 20.

THE SHAPE OF SURFACES

Once a surface is constructed, one needs to verify that it is acceptable. This verification needs to involve checking if the surface is within a prescribed tolerance to the given data points. Equally important for many applications, one needs to inspect its shape. One of the most critical type of surfaces occur in car body design: The slightest undulation in the surface of a hood, for example, will result in an unpleasant appearance of the car. These outer car body surfaces are referred to as "class A surfaces" in the trade. The term goes back to the CSD package, which was developed by Mercedes-Benz: "A" refers to "Aussennaut" or outer surface; these are the ones needing the highest level of perfection.

Thus, it is important to have tools that permit surface shape inspection before an expensive prototype is built. One method is to intersect the surface with a plane, resulting a curve. The curvature of this curve is plotted to aid a trained car designer to infer shape properties of the surface.

We show a less traditional application: The model of an American Indian vessel is inspected using a curvature plot of a cross section; see Fig. 11.

The vessel is not exactly a surface of revolution that is clearly revealed by the nonsymmetric curvature plot.

Another tool "paints" various curvatures[4] onto the surface, which was first suggested by R. Forrest (21). [These are Gaussian, mean, or absolute curvatures; see do Carmo (24)]. In this application, the curvature map is used as a *texture map* for displaying the surface. The result is a colored image of the surface that reveals much more shape information than a standard display. Again, we show the example of using curvatures in archeology: Figure 12 shows a vessel and its absolute curvatures.

Quantitative methods of reasoning about the shape of a vessel are becoming far more powerful than was possible when vessel shape was first given a mathematical treatment by G. Birkhoff (15). That book promoted the use of curvature for shape description, but it is only with modern CAGD techniques that we can realize the full potential of that approach.

Other developments are focusing on more involved aspects of the differential geometry of surfaces, the (probably) most promising one being the concept of *crest lines*. A crest line is, intuitively speaking, a line of extreme curvature on a surface; an example is the ridge line of a mountain. A crest line is defined as follows. Let $\kappa_{\max}(u, v)$ be the maximum principal curvature at a point $\mathbf{x}(u, v)$, and let $\mathbf{d}$ be the corresponding (domain) direction. The scalar-valued bivariate function $\kappa_{\max}(u, v)$ reports the extreme curvature at a given point. The extreme values of this function occur along curves; one such curve is defined by the zero contours of the directional derivative

$$D_{\mathrm{d}}\kappa_{\max}(u, v) = 0; \quad D^2_{\mathrm{d}}\kappa_{\max}(u, v) < 0. \tag{13}$$

All points satisfying equation 12 form the crest lines of a surface. They have recently been used in medical imaging and seem to be a useful concept in the shape analysis of surfaces arising in other areas of CAGD. For algorithms to find crest lines, see References 22 and 23. For an illustration, see Fig. 13.

TRIANGLE MESHES

In the 1990s, shape capture technology become widely available These are devices that can collect coordinates of points (also called *vertices*) on real objects. Several kinds of scanning devices exist:

- Laser scanners

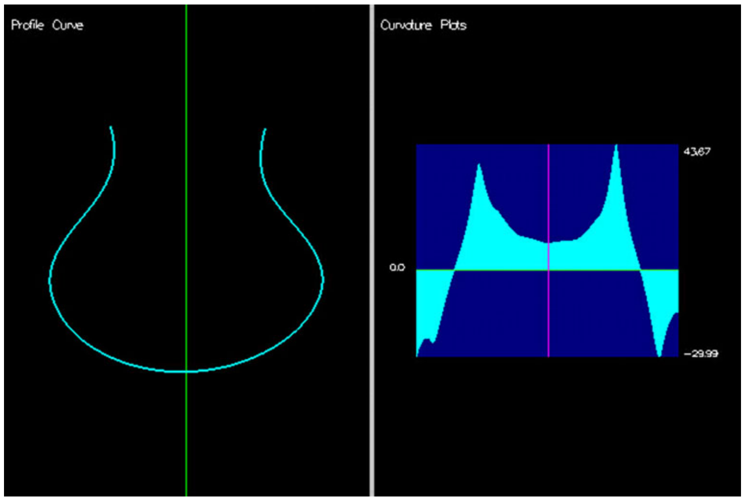

Figure 11. A vessel cross section and a curvature plot.

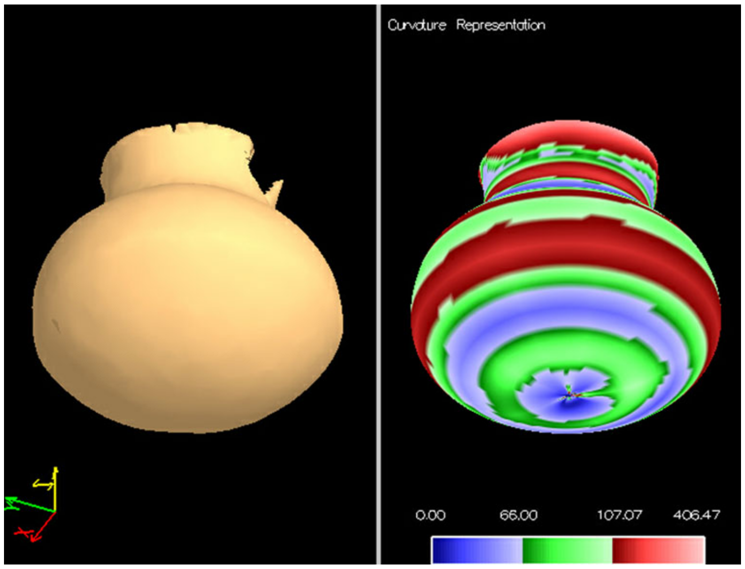

Figure 12. A vessel and its absolute curvatures.

- Stereo photogrammetry
- Touch probes
- Satellite remote sensors

The result of an object scan is a collection of x, y, z triples, ranging from several hundred to some billions. To make these point sets (called *point clouds*) manageable, they are typically endowed with a structure known as a *triangle mesh*. A triangle mesh is a data structure consisting of the list of all x, y, z triples plus a list of pointers. Each pointer (pointing to three vertices) represents one triangle. This additional connectivity information, while using up some space, helps speeding up many algorithms for working with meshes. Figure 14 shows an example of a triangle mesh.

Triangle meshes are not only obtained through digitizing. Many FEM codes deliver triangle meshes. Breaking a smooth surface down into many triangles for display also produces triangle meshes.

The local shape of a smooth surface is described using its curvatures. Triangle meshes are piecewise planar and not differentiable across triangle edges. To talk about curvatures, approximation methods must be used. If one is interested in the approximate curvatures around a vertex, a least-squares fit may be used to build a local approximating surface; degree 2 is usually sufficient. Then curvatures may be computed as in the previous section.

VOLUMETRIC DATA

Volume grids are 3-D arrays where each array element (called a *voxel*) holds one or more scalar values. There are devices whose output are volume grids, such as confocal laser microscopes or MRI scanners. Volume grids may also be the output of fluid flow computation.

When presented with a volume grid, a major task is to extract the shape information represented by it. For example, one might be interested in extracting all voxels of a

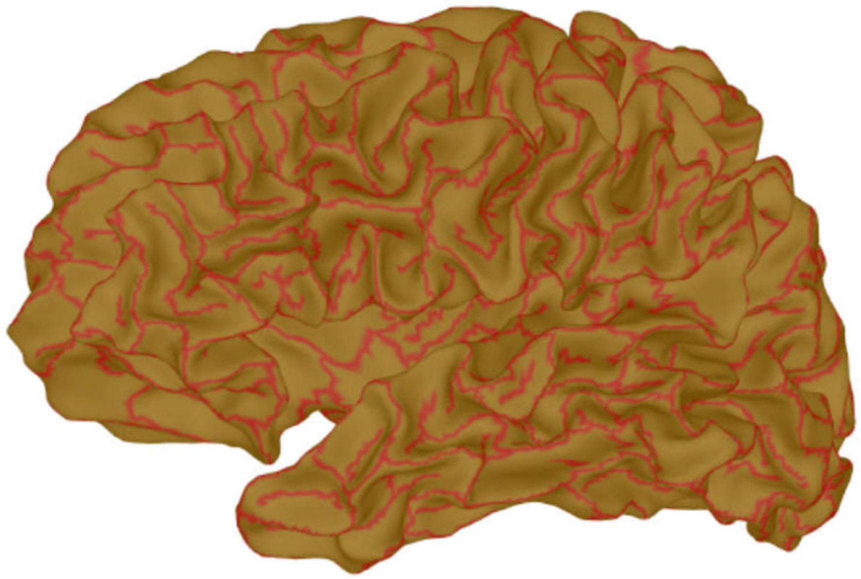

Figure 13. Crest line on a brain surface.

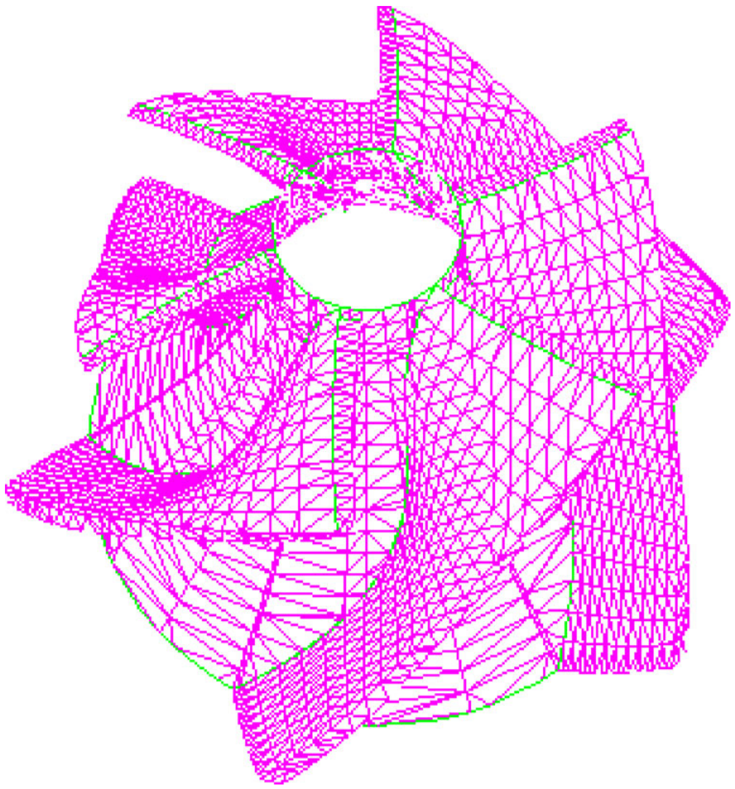

Figure 14. Part of a turbine modeled as a triangle mesh.

skull from an MRI data set. These voxels record the skull's density, a value not found anywhere else in the brain scan. The major algorithm for identifying those voxels is known as *marching cubes*; see Reference 25. It weaves a triangle mesh through those voxels where the known density level occurs. Figure 15 shows the result of the algorithm applied to a human brain scan. The image is a smooth rendering of a triangle mesh. For more information, see Reference 26.

BIBLIOGRAPHY

1. Farin, G. *Curves and Surfaces for Computer Aided Geometric Design*, 5th ed.; Morgan-Kaufamann: San Mateo, CA, 2001.
2. Faux, I.; Pratt, M. *Computational Geometry for Design and Manufacture*, Ellis Horwood: New York, 1979.
3. Gallier, J. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*; Morgan-Kaufmann: San Mateo, CA, 1998.

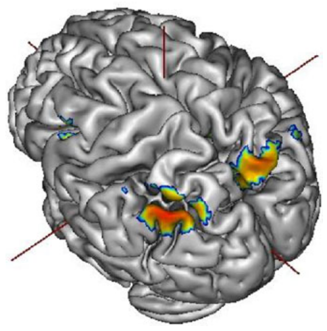

Figure 15. A brain surface obtained from the marching cubes algorithm.

4. Hoschek, J.; Lasser, D. *Grundlagen der Geometrischen Datenverarbeitung*; B.G. Teubner: Stuttgart, 1989. English translation: *Fundamentals of Computer Aided Geometric Design*; AK Peters: Boston, 1993.
5. Farin, G.; Hansford, D. *The Essentials of CAGD*; AK Peters: Boston, 2000.
6. Farin, G.; Hoschek, J.; Kim, M.S. Eds., *Handbook of 3D Modeling and Graphics*; Elsevoer: New York, 2002.
7. Lawson, C. Transforming Triangulations. *Discrete Math.* 1971, **3**, pp. 365–372.
8. Cline, A.; Renka, R. A Constrained Two-Dimensional Triangulation and the solution of Closest Node Problems in the Presence of Barriers. *SIAM J. Numer. Anal.* 1990, **27**, pp. 1305–1321.
9. Hjelle, O.; Daehlen, M. *Triangulations and Applications*; Springer-Verlag: New York, 2006.
10. Okabe, A.; Boots, B.; Sugihara, K. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*; John Wiley: Chichester, UK, 2000.
11. Blum, H. A Transformation For Extracting New Descriptors of Shape. In *Perception of Speech and Visual Form*; Wathen-Dunn, W., Ed.; pp. 362–380.
12. Bookstein, F. The Line Skeleton. *Comput. Graph. Image Process.* 1979, **11**, 123–137.
13. Lee, D.T. Medial Axis Transformation of a Planar Shape. *IEEE Trans. Pattern Anal. Mach. Intel.* 1982, **4**, pp 363–369.
14. Rosenfeld, A. Axial Representations of Shape. *Comput. Vision, Graph. Image Process.* 1986, **33**, pp 156–173.
15. Birkhoff, G. *Aesthetic Measure*, Harvard University Press: Cambridge, MA: 1933.
16. Boehm, W. On Cubics: A Survey. *Comput. Graph. Image Process.* 1982, **19**, pp 201–226.
17. Farin, G. *NURB Curves and Surfaces*, AK Peters: Boston, 1999.
18. Piegl, L.; Tiller, W. *The Book of NURBS*; Springer Verlag: New York, 1995.
19. Loop, C. A G^1 Triangular Spline Surface of Arbitary Topological Type. *Comput. Aided Geometric Design* 1994, **11**, pp 303–330.
20. Warren, J.; Weimer, H. *Subdivision Methods for Geometric Design*; Morgan-Kaufamann: San Mateo CA, 2003.
21. Forrest, A. On the Rendering of Surfaces, *Comput. Graph.* 1979, **13**, pp 253–259.
22. Stylianou, G; Fairn, G. Crest Lines for Surface Segmentation and Flattening. *IEEE Trans. Vis. Comput. Graph.* 2004, **10**, pp 536–544.
23. Thirion, J.P.; Gourdon, A. The 3d Marching Lines Algorithm. *Graphi. Models Image Process.* 1996, **58**, pp 503–509.
24. do Carmo, M. *Differential Geometry of Curves and Surfaces*; Prentice Hall: Englewood Cliffs, NJ, 2nd ed.; 1976.
25. Lorensen, W.; Cline, H. Marching Cubes: A High Resolution 3D surface Construction. *Comput. Graph.* 1987, **2**(4).
26. Chen, M.; Kaufman, A.; Ragel, R. *Volume Graphics*; Springer-Verlag: New York, 2000.

GERALD FARIN
Arizona State University