Polygonization of Implicit Surfaces

Jules Bloomenthal

CSL-87-2 May 1987

Abstract: This paper discusses a numerical technique that approximates an implicit surface with a polygonal representation. The implicit function is adaptively sampled as it is surrounded by a spatial partitioning. The partitioning is represented by an octree, which may either converge to the surface or track it. A piecewise polygonal representation is derived from the octree.

The technique is insensitive to the complexity of the implicit function, allowing the designer great latitude. With a polygonal representation of the surface available, certain computational economies result. In particular, the roots to the function need not be solved each time the surface is rendered.

CR Categories and Subject Descriptors: I.3.5 **[Computer Graphics]:** Computational Geometry and Object Modeling - curve, surface, solid, and object representations.

Additional Keywords and Phrases: Octree, polygonization, sampling.

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

Introduction

In recent years, implicit surfaces have become increasingly important to computer graphics. In part, this is due to a developing sophistication in the modeling of three-dimensional objects. Early computer graphics models generally were constructed by hand or generated explicitly with relatively simple techniques. More complex models have become practical by specifying the surface as those points obeying some property. When the property is written as a function of the surface points, $f(P) = 0$, an implicit surface is defined. The function f may be defined procedurally or approximated by data sets.

Because implicit surfaces conveniently define volumes, they are used frequently in CSG-based solid modelers [Requicha, 1982]. To maintain consistency within these modelers, the conversion of parametric polynomial surfaces into implicit form has received some attention [Sederberg, 1986]. Implicit surfaces also are used in the blending of shapes that otherwise would intersect [Blinn, 1982; Hoffman, 1985; Middleditch, 1985]. Many complex forms are defined more readily as implicit surfaces. For example, explicit techniques for constructing smooth, two-way branches [Charrot, 1984; Bloomenthal 1985] are not extended easily to n-way branches. As this paper illustrates, such extensibility is simple to obtain with an implicit function.

Recently there has been interest in deriving polygonal representations from implicit surfaces [Allgower 1980; Wyvill, 1985]. Many graphics systems rely upon polygons, which generally are faster to render than other primitives. The polygonization of implicit surfaces allows the model to be stored conventionally and this permits subsequent renderings to be made without repeated solution of the implicit function. In addition, a polygonal representation facilitates a number of geometric operations such as analysis of a surface and the resting of one surface upon another.

The technique presented here is to surround the implicit surface with an octree, at whose corners the implicit function is sampled. From these samples, surface vertices are calculated and connected to form a polygonal representation. The octree partitioning of space isolates the definition of the implicit function from the creation of the polygonal representation; thus, the function may be defined without regard to the conversion process.

Although spatial partitioning is not new to implicit functions, its organization as an octree, the concurrent adaptive refinement of the octree, and the algorithm for polygonizing each octree node are presented as new work.

The Implicit Function and its Evaluation

An implicit surface is defined as the set of points P satisfying the implicit function $f(P) = 0$. Only for a relatively simple function does a closed form solution exist; generally, numerical techniques are required to determine the roots of the function. The function itself may be inspected simply by displaying planar samples of f, such as shown in Figure 1. If g is defined as $g = f\text{-}c$ then, in solving for *g = 0*, a non-zero *f* is found; *c* is often called the ‘‘contour level’’ of the surface. In Figure 1, the surface cross-section contains one, two, or three discrete closed curves, depending on the contour level.

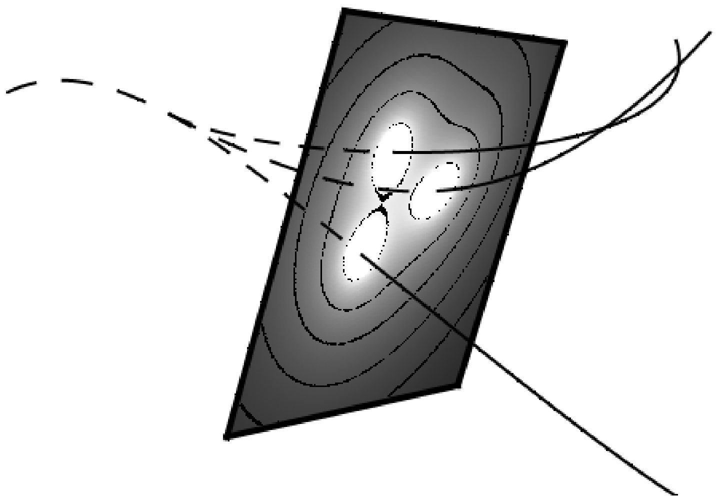


Figure 1: A slice measuring a function of distances to three space curves. Various contour levels are shown.

A series of planar slices, as shown in Figure 2, constitutes a voxel array, which may be rendered as a single image [Artzy, 1980]. A voxel array is readily procured for a number of real world objects, but often is computationally exorbitant ($O(n^3)$, *n* the size of the array) for synthetic ones. It becomes more economical if the spatial locations are distorted according to the surface [Thompson, 1985], but this greatly complicates the sampling process as well as presumes some a priori knowledge of the surface. A simpler approach, presented in the next section, is to adaptively sample within a fixed grid.

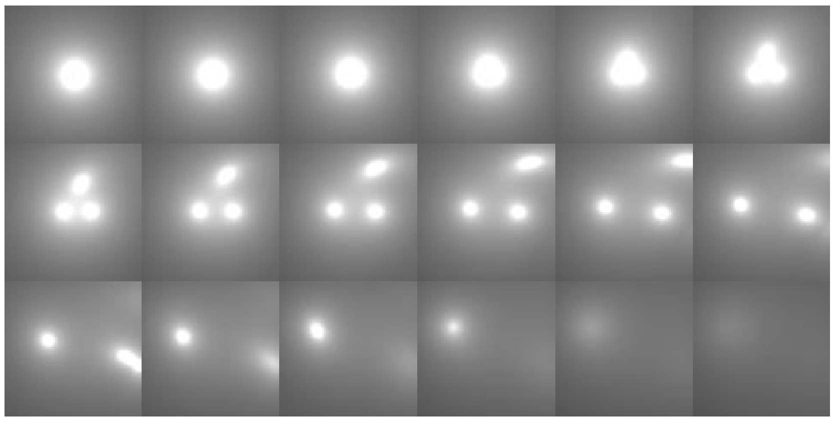


Figure 2: A series of slices, in reading order, constitutes a fixed grid sampling.

Spatial Partitionings

This paper considers two methods for sampling an implicit surface; each method partitions space into convex polyhedral cells whose union surrounds the surface. The first method converges to the surface, creating a hierarchical ‘‘converged partitioning;’’ the second tracks the surface, creating a ‘‘tracked partitioning.’’ In both cases, the function *f* is evaluated at each of the vertices, or corners, of the partitioning cell.

Only those cells that intersect the surface are retained in the partitioning; that is, each cell must contain a corner for which f evaluates negatively and a corner for which *f* evaluates positively. The surface intersects any edge that connects corners of opposite sign. Thus, the partitioning of space serves to limit the domain within which roots to the implicit function are searched.

If *f* is continuous, a root must exist along any edge connecting oppositely signed corners. If the definition of implicit surface is relaxed somewhat, to be the set of points that separate positive *f* from negative *f*, then one may converge arbitrarily close to the surface even if *f* is discontinuous. In cases where the partials to *f* are unknown or the function values are not relatively equidistant from the root, binary sectioning along the edge may converge to the root more quickly than other techniques, such as *regula falsi* [Dahlquist, 1974]. The convergence terminates when the distance between the converging points is less than some fraction of the length of the edge; the average of the two converging points is then considered to be a point lying on the implicit surface.

To form the polygonal representation, the surface vertices must be connected to their neighbors; this requires that the value of *f* at each corner be shared among adjacent cells. One technique to accomplish this is to store corner information in a hash table, indexed by the corner coordinates [Wyvill, 1986]. A more structured approach is for each cell to contain eight pointers to its corners. As new cells are created, they must point correctly to shared corners; for example, if the partitioning is hierarchical, a node of a child cell must point to the corner it shares with its parent.

Any cell or combination of cells that fills space may be used, but basic geometric properties, such as corner locations and face planes, are computed more simply if the cells are identical and similarly oriented. In three dimensions, the only such cell that fills space is the cube [Coxeter, 1963]; it enjoys a number of rotational symmetries, and divides into eight similarly oriented cubes. However, negatively signed corners of a cube cannot, in all cases, be separated from positive corners by a single plane. This will be an important consideration when discussing the development of the polygonal representation.

Converged Partitioning

The implicit surface may be represented as an octree, which is a hierarchical partitioning of space formed by the subdivision of cubes, beginning with a cube that bounds the surface [Meagher, 1982]. The octree converges to the surface by subdivision of those cubes that intersect the surface; this continues recursively for each descendant cube until a limiting depth of recursion is reached or other criteria are satisfied.

The faithfulness to the surface of the converging octree depends upon its depth of recursion. Although the details of polygonization are presented in a later section, some results are presented here to demonstrate the relationship between depth of recursion and faithfulness to the surface. As shown in Figure 3, the sphere, *f(P) = radius*-|*P-Center*|, is approximated rapidly with increasing depth.

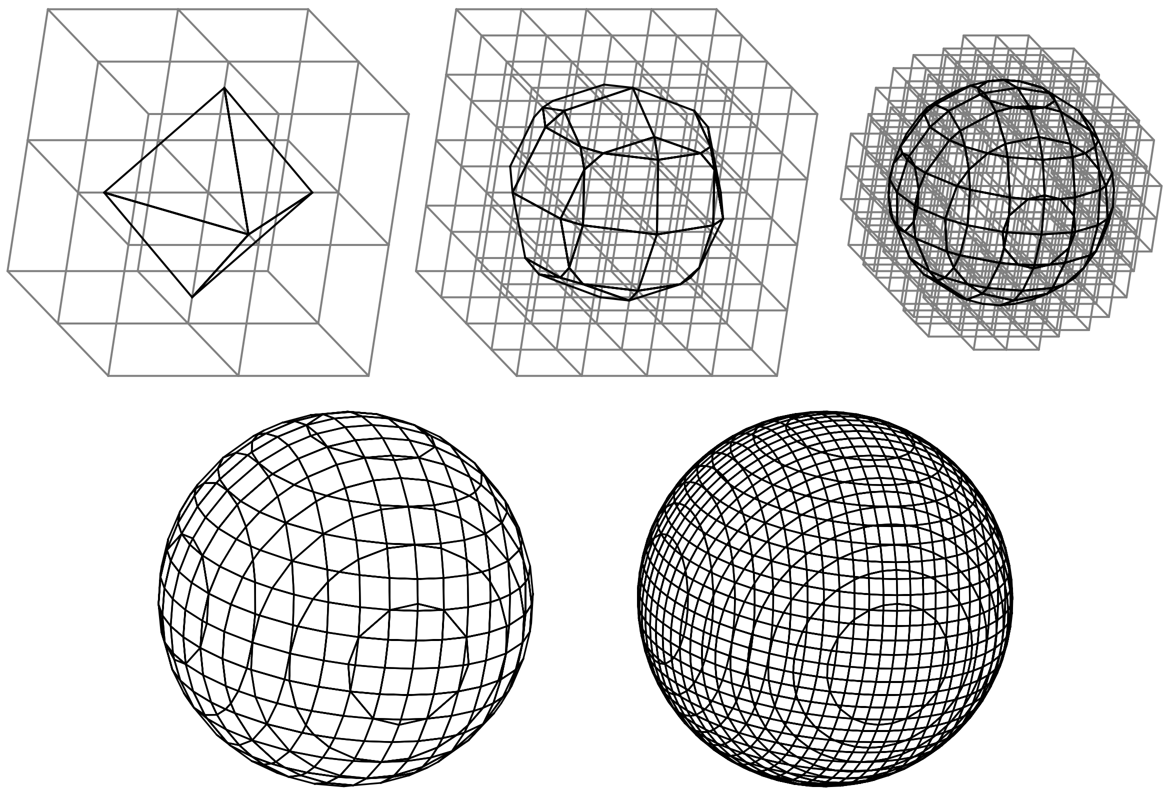


Figure 3: A sphere approximated from depths 1 through 5.

A disadvantage with the converged partitioning is that small surface detail may be missed by a large cube, resulting in a premature termination in the subdivision of the cube. Thus, a contiguous object may be broken into pieces. Another disadvantage is that, if the extent of an object is not correctly estimated, the object may be truncated, as shown in Figure 4. These problems may be mitigated by various ad hoc techniques; for example, the surface gradient at the cube corners may serve to estimate whether the surface penetrates the cube.

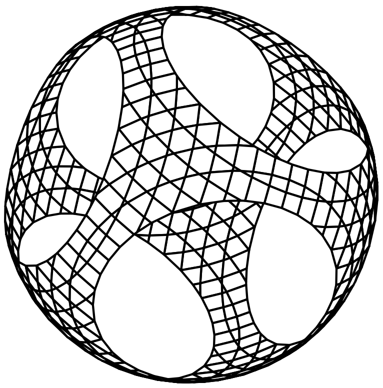


Figure 4: A sphere truncated by the root cube.

Tracked Partitioning

A different approach to spatial partitioning is to track the surface by cell propagation. An initial ‘‘seed cell’’ is established that intersects the surface and is small in comparison to surface detail. New cells propagate along established cell edges that intersect the surface. The cells may be stored as an adjacency graph, with care required that no cell is added redundantly to the graph. Unlike converged partitioning, tracked partitioning requires a seed cube for each disjoint object. The location of the seed cell can be specified by the user or determined by iteration. Figure 5 illustrates a tracked partitioning.

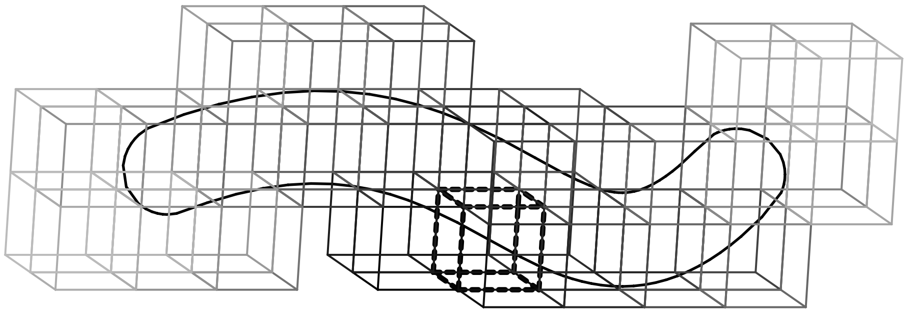


Figure 5: Surface tracking: darker cubes are older; the seed cube is dashed.

The resulting cubes may be subdivided to refine the polygonal representation. Unfortunately, an adjacency graph formed by the tracking process is not as amenable to adaptive subdivision as is the converged octree. This is remedied if an ‘‘aggregated octree,’’ rather than adjacency graph, is constructed during the surface tracking. As cells are added to the octree, new parent and root nodes are required, as illustrated in Figure 6. Octree neighbor finding techniques [Samet, 1984] are used to prevent any redundant measuring or filling of space.

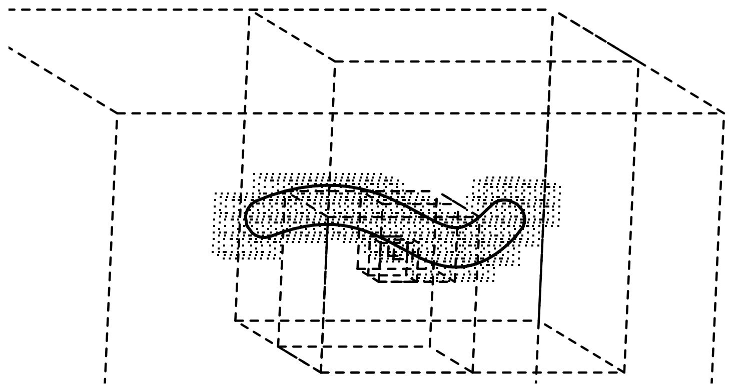


Figure 6: Creation of the aggregated octree; the expanding root nodes are dashed.

Polygonizing the Cell

A polygonal representation is a list of surface vertices and their interconnections. It may be derived from a cubic partitioning by processing those cubes intersecting the surface. If the partitioning is an octree, the processing is confined to terminal nodes of the tree. For each cube to be processed, the surface vertices are ordered, forming a convex polygon whose sides are each embedded in a cube face [Wyvill, 1986]; the process is local to each cube.

This paper introduces a simple algorithm, illustrated in Figure 7, to perform the three-dimensional ordering of surface vertices. It is similar to a cartographic technique that derives two-dimensional contours from elevation data [Cottafava, 1969]. The ordering begins with any surface vertex on the cube and proceeds towards the positive corner and then clockwise about the face to the right until another surface vertex is reached. This iterates until the polygon is closed. Although a table method is possible [Lorensen, 1987], the algorithm is especially useful for adaptive subdivision, discussed later.

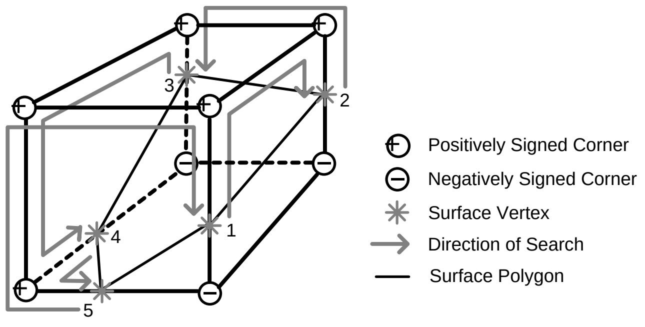


Figure 7: Algorithm to order vertices.

Unfortunately, as demonstrated in Figure 8, the polygonization may produce multiple polygons; it is ambiguous as to whether the surface multiply slices the cube or is a narrow cylinder that surrounds two opposing corners. No such ambiguity exists if a simplex is the partitioning cell. A simplex is the simplest linear decomposition of n-space, which, for three-space, is the tetrahedron. Any two vertices of a tetrahedron are connected by an edge and, thus, oppositely signed corners of a tetrahedron can be separated by a single plane. Although techniques exist for partitioning space with simplices [Allgower, 80; Dobkin, 86], adaptive subdivision is difficult because the tetrahedron subdivides into four tetrahedra and one octahedron.

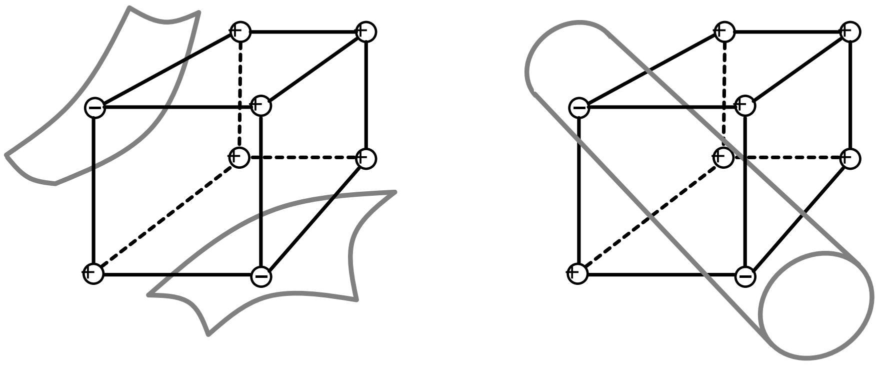


Figure 8: An ambiguous polygonization.

Although creating the union of the two possibilities in Figure 8 often suffices, a more reliable approach is to section the cube into twelve tetrahedra, requiring an additional sample at the center of the cube [Hunter, 1987]. Figure 9 illustrates the cube sections; the same vertex ordering algorithm applies to the tetrahedron.

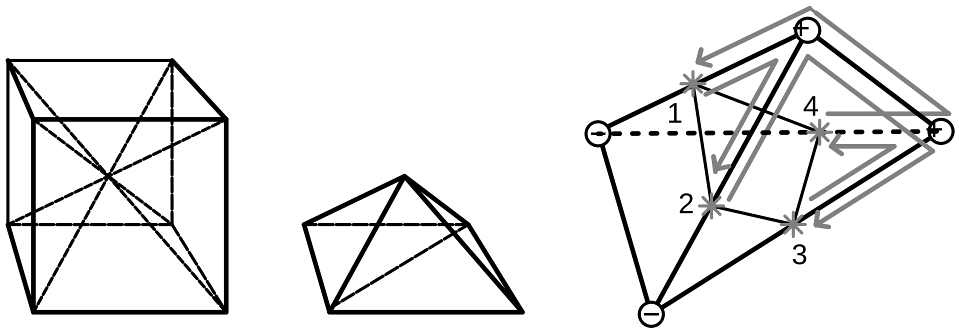


Figure 9: Tetrahedral sections of a cube. A cube is sectioned into six, five-cornered pyramids; each pyramid is divided into two tetrahedra.

The polygon is stored simply as a list of its vertices, or, to provide more topological information, as indices into a list of all vertices. To accomplish the latter, as the polygons are created, one cube at a time, any new surface vertex is assigned an index and the index is associated with the edge containing the vertex. If this edge is subsequently visited during the polygonization of a neighboring cube, the existing index is used, establishing the connectedness between polygons. Determining the existence of an index is possible because the corners of the edge are shared between adjacent cubes. The polygons resulting from this method may be decomposed into triangles, making for more uniform storage.

Connecting the vertices into polygons creates a linear surface representation. Higher order approximations, such as cubic patches, may be created from the vertices, if the corresponding surface normals are known [Crow, 1988]. The boundary curves for the patch are computed locally from two endpoints and two tangents; the tangents are derived from the surface normals. The higher order representation may alleviate problems, such as a lack of smoothness along silhouette edges, traditionally associated with images of polygons.

The polygonization method will always produce a closed surface given a root octree that completely encloses the solid. At times, however, the designer may wish to clip away parts of the solid for the purpose of viewing. Thus, it is sometimes useful to introduce a vertex acceptance criterion whereby clipped vertices are rejected; any polygon that consists solely of rejected vertices is itself rejected.

Surface Normals

At a point on the surface, the normalized gradient, ∇f, is the surface normal, presuming not all its partials are zero. With three more evaluations of the implicit function, the gradient is approximated:

$$\nabla f \approx ([f(P)-f(P_x)]/\Delta, [f(P)-f(P_y)]/\Delta, [f(P)-f(P_z)]/\Delta),$$

where $P_{x,y,z}$ represent P displaced by Δ along the respective axes.

In shading the surface to produce an image, accuracy of the surface normal is generally more important than the accuracy of the surface vertices. Because the normal vector computation is very sensitive to small fluctuations in the implicit function, it is important that the function be accurately evaluated and continuous (for discontinuous functions, an inferior approximation to the normal may be computed as a weighted average of the polygon normals surrounding the vertex). Standard references on numerical techniques caution that errors of the function are of much greater importance in numerical differentiation than in interpolation or integration [Dahlquist, 1974]; too *small* a Δ may be numerically unstable. The present implementation sets Δ to one-hundredth the size of a cube.

Adaptive Subdivision

The estimate of the surface may be improved by subdividing those cubes containing highly curved or intersecting surfaces. A similar approach using adaptive *quadtree* subdivision has improved the quality of rendered images; the criteria for subdivision were based upon object characteristics, such as tangency and curvature, chiefly as they apply in screen space [Von Herzen, 1987].

Using the polygon resulting from an octree node, criteria for subdivision of the node include:

whether any edge of the cube intersects the surface,
whether a maximum subdivision depth or a minimum cube size has been reached,
whether more than one polygon results from the cube,
the planarity of the polygon, and
the divergence of vertex normals from the normal at the polygon center.

Given the polygon vertices, P_i, their unit length normals, $\mathbf{N_i}$, and the unit length normal $\mathbf{N}$ at the polygon center, the planarity of the polygon can be estimated by:

$MAX(V_i . \mathbf{N}),\ i \in [1, nPoints]$ and V_i the unit length vector from P_i and P_{i+1},

and the divergence of the vertex normals can be estimated by:

$MIN(\mathbf{N_i} . \mathbf{N}),\ i \in [1, nPoints]$

The magnitude of the surface gradient, scaled by the cube size, is not an appropriate measure of curvature; the implicit function could be planar, for example, and still have a large gradient orthogonal to the plane. Another measure is the maximum deviation of a cube corner value from a linear function created as a least squares fit to the eight corner values [Heckbert, 1987].

Certain topological criteria warrant the subdivision of an adjacent cube. If the edge of a parent cube connects two equally signed corners and the midpoint is differently signed, as in Figure 10, left, then the three neighbors along that edge should be subdivided. For each face of a parent cube, if the four child corners that are midpoints of the four edges of the face all agree in sign but disagree with the center of the face, Figure 10, right, then the face neighbor should be subdivided. Without such subdivision, a hole will appear in the surface.

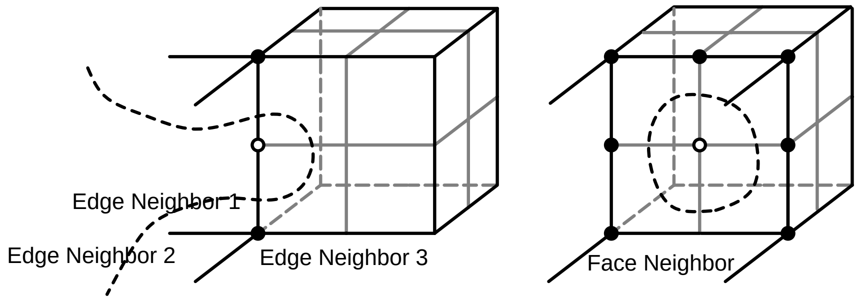


Figure 10: Conditions warranting subdivision of adjacent cubes. Left: midpoint of edge. Right: midpoint of face.

Adaptive subdivision requires a more robust polygonization scheme. As shown in Figure 11, left, a crack occurs along the shared face of two cubes, one of which is subdivided, the other not; a similar problem relative to quadtree methods was solved by restricting the quadtree and introducing additional edges [Von Herzen, 1987]. The solution presented here is less restrictive, but sensitive to any of the anomalies shown in Figure 10. The algorithm to order vertices, depicted in Figure 7, is modified as illustrated and described in Figure 11. This algorithm created the generalized cylinder of Figure 12 (shown orthographically, for the sake of clarity).

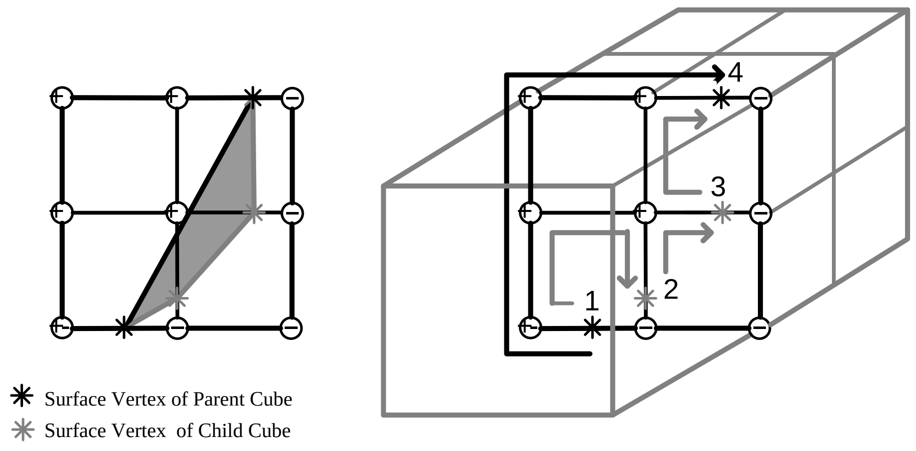

4
3
2
1
✳ Surface Vertex of Parent Cube
✳ Surface Vertex of Child Cube

Figure 11: Polygonization of adaptively subdivided cubes. Left: A crack would occur without modification to the cube polygonization algorithm. Right: The modification consists of tracking along the more highly divided face. When proceeding around a face, if the face neighbor is more highly divided, proceed about the neighbor's face, reversing direction at every surface vertex, until reaching the next surface vertex located on an edge belonging to the original face. Although shown for the face of a tetrahedron, the same principle will apply to the face of a tetrahedron.

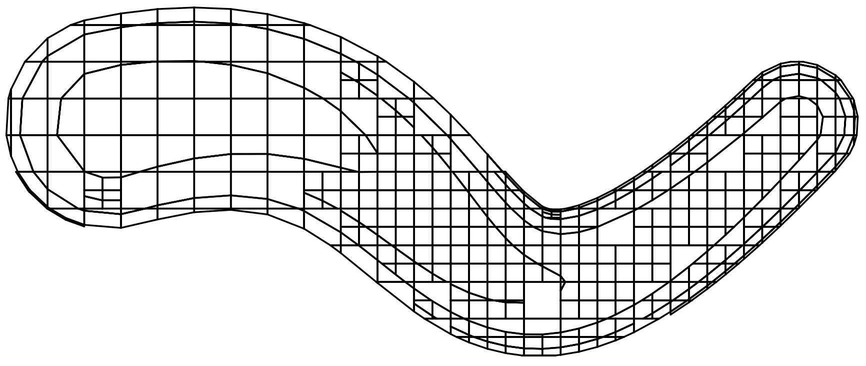


Figure 12: Adaptively subdivided generalized cylinder.

Adaptive subdivision is, in effect, performed during the creation of the converged octree. It may be applied subsequent to the creation of an aggregated octree, but it may also be concurrent with its creation, reducing the chance of truncation of surface detail. An algorithm for concurrently adapted tracked partitioning is below. Figure 13 demonstrates the improved results.

Algorithm for concurrently adapted tracked partitioning

```
1. A stack of cubes is initialized as empty; when added, a cube is flagged as being on the stack.
2. A cube is "considered" in the following manner:
       It is tested against the subdivision criteria described above.
       If it should subdivide, then subdivide the cube and consider its child cubes.
       Else:
           Test its parent against the coalescence criteria:
               Coalesce if subdivision is not indicated for the parent and the parent does
               not contain any of the anomalies shown in Figure 10.
           If it coalesces, then:
               Retain the cube and its siblings in the octree to avoid redundant testing.
               Flag the cube and its siblings so they won't propagate when read from the
               stack.
               Flag the parent as a terminal node; add it to the stack.
           Else, add it to the stack.
3. Create a seed cube, smaller than the surface detail and centered on a surface point:
       Make it the root octree node and consider it.
4. While cubes remain on the stack:
       Remove a cube from the stack.
       If the cube has no children or if its children have been coalesced, then for each of its faces
       that contains an edge intersecting the surface:
           If the cube already has an octree neighbor adjacent to the face, then:
               If the neighbor is not flagged as on the stack, consider the neighbor.
           Else:
               Create a neighbor cube and add it to the octree.
               If the neighbor descends from any node in the octree that is coalesced:
                   Then coalesce all descendants from that point.
                   Else, consider the neighbor.
5. Remove coalesced cubes from the octree.
6. For any terminal cubes containing a Figure 10 anomaly, subdivide and recurse.
```

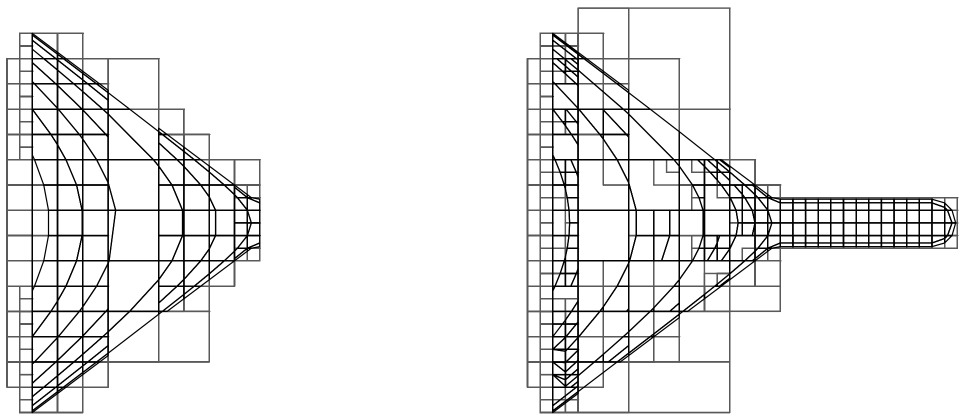


Figure 13: Subsequent subdivision cannot correct the truncation of a tracked partitioning (left); but tracking concurrent with subdivision properly follows the surface (right). For both, the partitioning proceeded from left to right.

Examples

In this section a small range of surfaces that may be polygonized with the above method is explored, and several characteristics of the method are illustrated. For example, a ‘‘blobby’’ molecule (i.e., equipotential surface) is created simply by summing the values of two spherical functions, as shown in Figure 14.

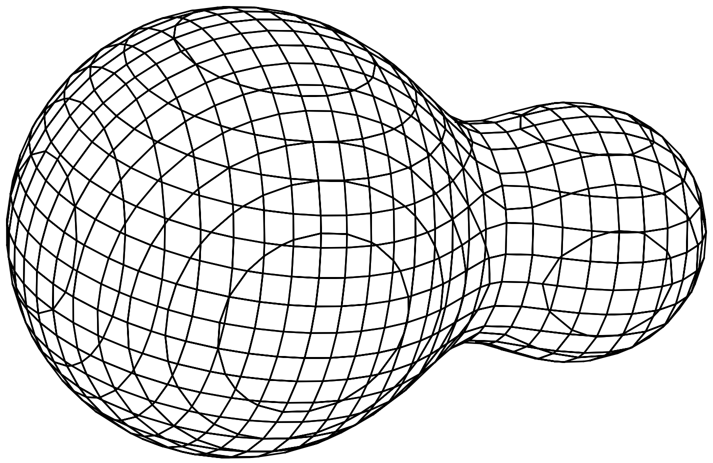


Figure 14: ***f(P) = g(P)+h(P)*, where *g(P)* = $radius_g^2/|Center_g-P|^2$, and similarly for *h*.**

The polygonization method is indifferent to the topology of a surface. For example, an equation of a torus produces Figure 15. Given an implicit surface that is closed and continuous, Euler’s formula may be used to compute the genus of the surface from the number of vertices, polygons, and edges. This does not apply if the root cube truncates the surface (Figure 4); such a condition exists if any surface vertex lies on the edge of the root cube.

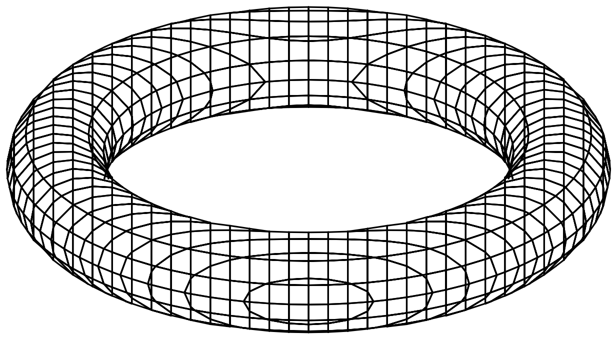


Figure 15: A torus.
$f(P) = x^4+y^4+z^4+R^4+r^4+2(x^2y^2+x^2z^2+y^2z^2-(R^2+r^2)(x^2+z^2)+(R^2-r^2)y^2-R^2r^2)$,
where *R* and *r* are the major and minor radii.

An offset surface may be defined implicitly as a set of points a fixed distance from another surface. Figure 16 displays a surface offset from a triangle.

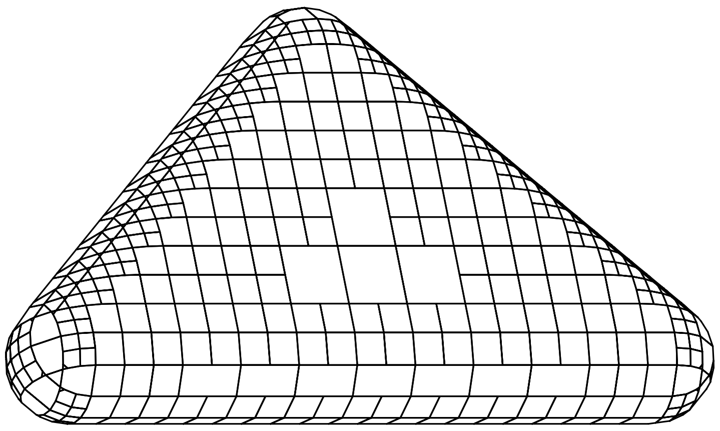


Figure 16: An adaptively sampled offset surface.

A number of observations regarding implicitly defined models follow a consideration of the generalized cylinder [Agin, 1972]. The generalized cylinder may be represented as an implicit function of distance to the nearest point on the cylinder's axis. Wyvill et al. modeled the axis as a set of discrete points, a large number being necessary to avoid a lumpy appearance [Wyvill, 1986]. Alternatively, the axis may be modeled by cubic splines and the nearest point determined algebraically.

The generalized cylinder produced by the implicit technique may, however, differ from the surface produced using explicit techniques. The explicit model is perhaps the most familiar and conventional; it explicitly provides its surface points and explicitly states their connectedness. Complex explicit shapes may be difficult to construct; for example, a generalized cylinder must compute reference frames, applying more at high curvature and at the hemispherical ends, as shown in Figure 17, left. Unfortunately, it is difficult to obey certain physical constraints, such as smoothness, that may be placed on the model. Branching or severe flexing, for example, can cause self-intersections and discontinuities, Figure 17, right.

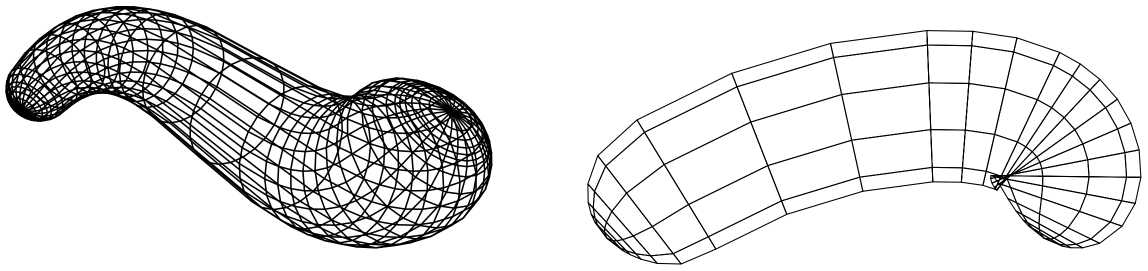


Figure 17: Two explicitly constructed, non-implicit surfaces. One is smooth (left), but the other is not (right).

The implicit model, Figure 18, avoids these problems by separating the model definition from the polygonization. If the proper function is used, sharply flexing surfaces will remain smooth and not interpenetrate, a distinct advantage in creating realistic models.

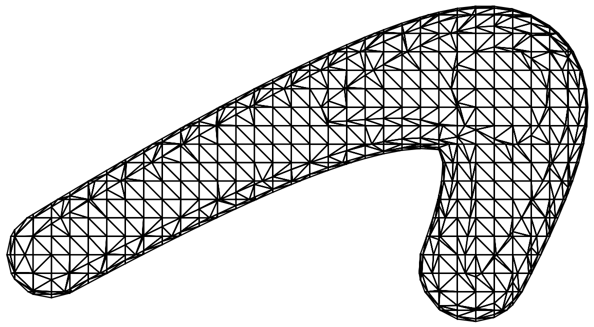
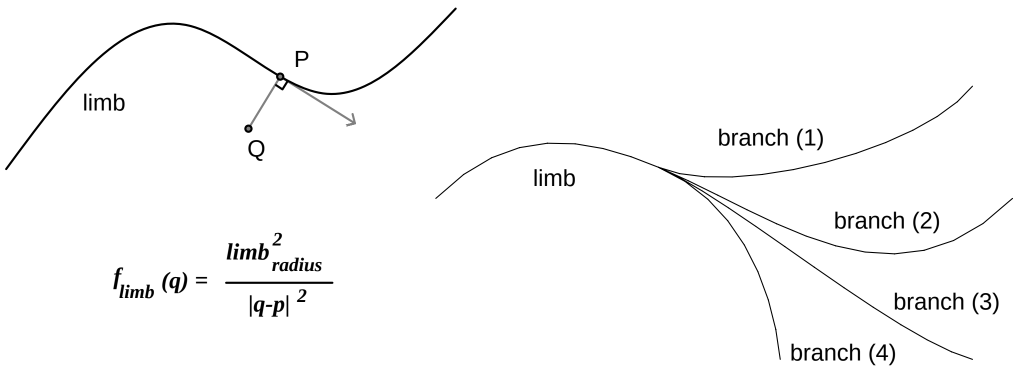


Figure 18: Implicit surfaces can avoid discontinuities. The spatial value is proportional to the distance to, and the curvature of, the closest point on the defining curve. Here, the polygons have been triangulated.

Generalized cylinders can demonstrate that implicit surfaces are indifferent to interesting topographies such as a branching surface. Unlike the explicit definition, the implicit definition of a branching generalized cylinder is easily extended to n-way branching, as illustrated in Figure 19. The resulting surface, shown in Figure 20, does not exhibit sharp angles; this smoothness is an attribute difficult to achieve with explicitly defined branching structures [Charrot, 1984; Bloomenthal, 1985].



$$f_{limb}(q) = \frac{limb^{2}_{radius}}{|q\text{-}p|^{2}}$$

$$f_{branches}(q) = Max\left(f_{limb}(q), \overset{nBranches}{\underset{i=1}{,}} f_{branches(i)}(q)\right)$$

Figure 19: n-way branch function.

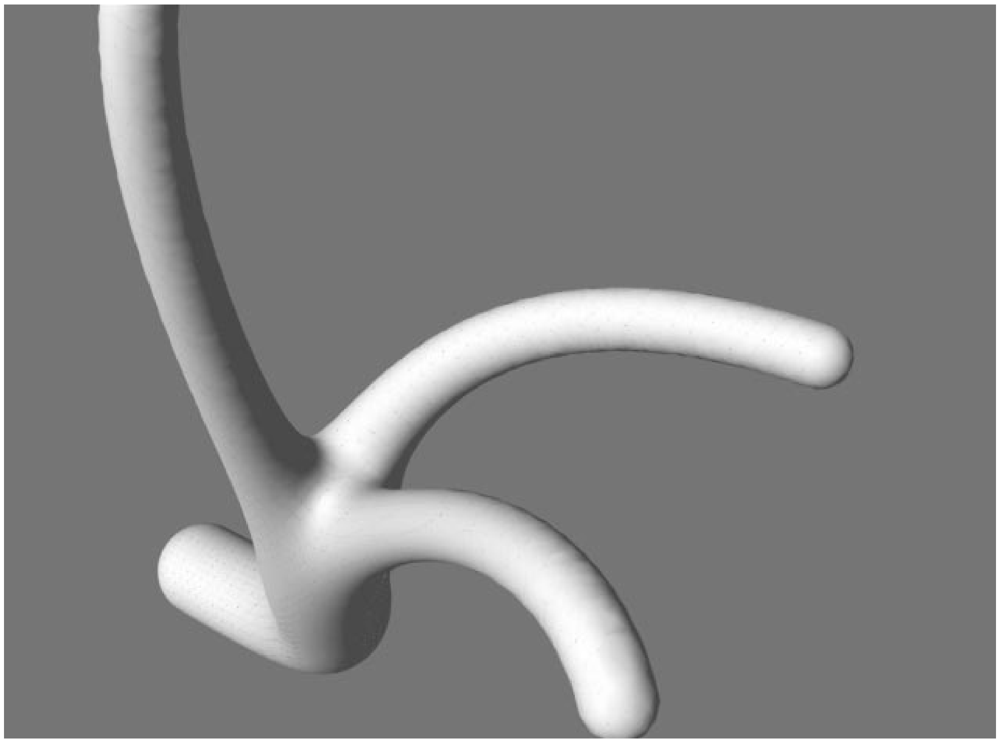


Figure 20: A tri-furcated ramiform.

Conclusions

This paper has presented a method for approximating an implicitly defined surface with a polygonal representation. The method is in three parts: a spatial function or procedure that defines the surface, an octree that converges to or tracks the surface, and a polygonization of individual octree cells.

The ability to derive a polygonal representation from an implicit surface has a number of consequences for the modeling and rendering processes. First, implicit surfaces may be rendered with polygonal techniques in addition to the ray tracing techniques usually used [Blinn, 1982; Hanrahan, 1987]. Second, the method isolates the polygonal representation from the complexities of the surface definition, thus improving the flexibility of a modeling environment. With small programming effort relatively complex shapes become possible. Last, the availability of polygonal data may have diverse animation or engineering applications.

Given a general set of octree operations, the basic polygonal representation method is implemented without difficulty. Adaptive subdivision requires a considerable additional effort. The algorithms were implemented on a Xerox Dorado processor, comparable in power and memory to a VAX 11/780. The creation of a non-adaptive octree and subsequent generation of more than 20,000 polygons for Figure 21 required about two hours. The single greatest factor relative to computation time is the complexity of the implicit function itself; it is important that the polygonization algorithm be implemented so that no point is sampled more than once.

A number of issues await future investigation. Because the method is numerical, the choice of the root cube (for the converging method) or the seed cube (for the tracking method) is important. If f is a polynomial, it is possible to estimate its extent in space for the purpose of creating an initial root cube [Arnon, 1984]; it is also possible to determine the maximum curvature of f for the purpose of determining the size of the seed cube [Sokolnikoff, 1964]. This paper has not discussed the deviation of the polygonal representation from the true surface, although in the case of simplices, error bounds have been developed [Allgower, 1987].

The octree is a convenient mechanism for adaptively storing information about the implicit surface; its partitioning of space provides a framework for connecting surface points into a polygonal representation. The convergence along a cube edge to find a surface vertex is accelerated because of the small size of the interval and its alignment with a major axis. This alignment and the regularity of the sampling lattice, however, may introduce undesirable aliasing artifacts for certain surfaces; these artifacts might be reduced by stochastic jitter of the cube corner locations [Cook, 1986].

A variety of implicit surfaces remain to be tested with the above techniques. Deformations, for example, might be implemented as functional compositions. Solid texture, increasingly popular as a step in the rendering process, might be transformed into a surface and given the properties of an object.

Many of the functions given in this paper depend upon the distance to a point or to a curve; it may be interesting to investigate functions of distance to other surfaces. Even with the limited tests used to illustrate the techniques presented in this paper, the resulting surfaces have structured, smooth, and natural qualities, which were the goals of this research.

Acknowledgments

To Eric Bier, for providing a root finder that confines itself to the unit interval. To Brian Tramontana and Bridget Scamporrino, for photographic assistance. To Nelson Max and Alain Fournier, for valuable discussions. To Kevin Hunter, for conversations regarding his own space-partitioning technique. To Debra Adams, Paul Haeberli, and Nancy Gill, for critical reading. To Pat Hanrahan and Ken Shoemake, for assistance with the writing. And especially to Paul Heckbert, for providing critical references, stressing the value of adaptive subdivision, and giving several critical readings of the paper.

And to Xerox, for its support of research.

References

Agin, G.J. Representation and description of curved objects. *Memo AIM-173*, Stanford Artificial Intelligence Report, October 1972.

Allgower, E.L. and Georg, K. Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *SIAM Review* **22**, 1 January 1980, 28-85.

Allgower, E.L. and Gnutzmann, S. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal of Numerical Analysis,* **24**, 2 April 1987, 452-469.

Arnon, D.S., Collins, G.E., and McCallum, S. Cylindrical Algebraic Decomposition. *SIAM Journal on Computing,* **13**, 4 November 1984, 865-889.

Artzy, E., Frieder, G., and Herman, G.T. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. Proceedings of SIGGRAPH'80 (Seattle, WA, July 14 18, 1980). In *Computer Graphics* **14**, 3 July 1980, 2-9.

Blinn, J.F. A generalization of algebraic surface drawing. *ACM Transactions on Graphics,* **1**, 3 July 1982, 235-256.

Bloomenthal, J. Modeling the mighty maple. Proceedings of SIGGRAPH'85 (San Francisco, California, July 22-26, 1985). In *Computer Graphics* **19**, 3 July 1985, 305-311.

Charrot, P. and Gregory, J. A pentagonal surface patch for computer aided geometric design.

Computer Aided Geometric Design, **1**, 1 July 1984, 87-94.

Cook, R.L. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, **5**, 1 January 1986, 51-72.

Cottafava, G. and Le Moli, G. Automatic contour map. *Communications of the ACM*, **12**, 7 July, 1969, 386-391.

Coxeter, H.S.M. *Regular Polytopes*. Macmillan, New York, 1963.

Crow, F. Patches from polygons: interpolating smooth surfaces over irregular data. in preparation.

Dahlquist, G. and Bjorck, A. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

Dobkin, D.P., Thurston W.P. and Wilks A.R. Robust contour tracing. *Technical Report CS-TR-054-86*, Computer Science Dept., Princeton University, September 1986.

Hanrahan, P. A survey of ray-surface intersection algorithms. Glassner, A. ed. Introduction to Ray Tracing, SIGGRAPH'87 course notes #13, Anaheim, CA, July 26-32, 1987.

Heckbert, P. Personal Communication, 1987.

Hoffman, C. and Hopcroft, J. The potential method for blending surfaces and corners. Technical Report TR 85-674 Computer Science Dept., Cornell University, 1985.

Hunter, K. Personal Communication regarding a 1983, unpublished report on the contouring of analytic functions, 1987.

Lorensen, W.E. and Cline, H.E. Marching cubes: a high resolution 3d surface construction algorithm. Proceedings of SIGGRAPH'87 Anaheim, CA, July 27-31, 1987. In *Computer Graphics* **21**, 4 July 1987, 163-170, to appear.

Meagher, D. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, **19**, 2 June 1982, 129-147.

Middleditch, A.E. and Sears, K.H. Blend surfaces for set theoretic volume modeling systems. Proceedings of SIGGRAPH'85 San Francisco, CA, July 22-26, 1985. In *Computer Graphics* **19**, 3 July 1985, 161-170.

Requicha, A.A.G. and Voelcker, H.B. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, **2**, 2 March 1982, 9-24.

Samet, H. The quadtree and related hierarchical data structures. *Computing Surveys*, **16**, 2 June 1984, 187-260.

Sederberg, T.W. and Goldman, R.N. Algebraic geometry for computer-aided geometric design. *IEEE Computer Graphics and Applications,* **6**, 6 June 1986, 52-59.

Sokolnikoff, I.S., *Tensor Analysis: Theory and Applications to Geometry and Mechanics of Continua*. J. Wiley and Sons, New York, 1964.

Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W. *Numerical Grid Generation: Foundations and Applications*. North-Holland, New York, 1985.

Von Herzen, B. and Barr, A. Accurate triangulations of deformed, intersecting surfaces. Proceedings of SIGGRAPH’87 Anaheim, CA, July 27-31, 1987. In *Computer Graphics 21*, 4 July 1987, 103-110.

Wyvill, G., McPheeters, C., and Wyvill, B. Data structure for soft objects. *Visual Computer*, **2**, 4 August 1986, 227-234.