A Methodology for Piecewise Linear Approximation of Surfaces*

Luiz Velho[†], Luiz Henrique de Figueiredo[‡], and Jonas Gomes[†]

[†]IMPA – Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, Brazil
lvelho@visgraf.impa.br jonas@visgraf.impa.br

[‡]LNCC – Laboratório Nacional de Computação Científica
Rua Lauro Müller 455, 22290-160 Rio de Janeiro, RJ, Brazil
lhf@lncc.br

Abstract

We discuss the problem of adaptive polygonization of regular surfaces of the euclidean 3D space, and present effective algorithms for computing optimal polygonizations of surfaces described in parametric or implicit form.

Keywords: Surface approximation, polygonization, parametric surfaces, implicit surfaces, geometric modeling.

1 Introduction

The polygonization of surfaces is a classical problem in computer graphics and geometric modeling that has many practical applications. The problem is computing a piecewise linear approximation for a smooth surface described either in parametric or implicit form.

In this paper, we present a conceptual framework for the piecewise linear approximation of surfaces and also a methodology for computing good polygonal approximations while keeping the number of polygons low. Based on the general principles in this methodology, we describe two specific new algorithms for the adaptive polygonization of parametric and implicit surfaces.

1.1 Importance of the Problem

A polygonal approximation is the simplest form of surface description and therefore is the representation of choice in the implementation of a large number of algorithms. Moreover, existing graphics hardware and libraries (e.g., OpenGL) have special support for polygonal meshes, specially triangular meshes. Thus, despite the existence of more sophisticated standard forms for surface description (e.g., Bézier, B-splines, NURBS, etc.), there is always a need to represent surfaces in polygonal form.

1.2 Parametric and Implicit Surfaces

The piecewise linear approximation of surfaces can be posed as the problem of obtaining a polygonal approximation for a smooth 2-dimensional manifold embedded in the 3-dimensional euclidean space $\mathbf{R}^3$. This geometric object is usually called a *regular surface* [1].

By conveniently subdividing a regular surface S, we can write $S = \bigcup_{i=1}^{m} S_i$, where each S_i is a regular surface (with boundary), defined either parametrically or implicitly. In other words, for each i there exists a regular parameterized patch $f_i\colon U_i \subset \mathbf{R}^2 \to S_i \subset \mathbf{R}^3$, $f(U_i) = S_i$, or an "implicit patch" $g_i\colon \mathbf{R}^3 \supset V_i \supset S_i \to \mathbf{R}$, such that $g^{-1}(0) = S_i$.

Therefore, in this context, the problem of approximating a regular surface reduces to the problem of obtaining a polygonal approximation for an implicit or a parametric surface patch.

1.3 Motivation and Goals

Polygonization methods for parametric and implicit surfaces have been studied exhaustively in the literature, but no conceptual survey exists. Since conversion between parametric and implicit representations is difficult [2], or even impossible, it is important to devise good polygonization methods for both class of surfaces.

The polygonization problem is well understood, and the existence of many published methods show that the area is mature. Therefore, there is a need for a review of the state of the art, containing a theoretical

*Research supported by grants from the Brazilian Council for Scientific and Technological Development (CNPq) and Rio de Janeiro State Foundation for Research (FAPERJ).

discussion and an analysis of the basic mechanisms behind existing solutions. The goal in any review of this kind should be to look for a deeper understanding of the problem, one that helps to reveal new solutions. This paper contains both a conceptual review of the state of the art and new solutions for the polygonization problem.

1.4 Structure of the Paper

In Section 2, we discuss the polygonization problem as a mathematical problem that needs computational solutions. In Section 3, we extend this theoretical discussion to adaptive methods. In Section 4, we discuss previous methods in the light of this theoretical framework. Section 5 describes a new polygonization method for parametric surfaces, and Section 6 describes a new polygonization method for implicit surfaces. Finally, Section 7 contains a discussion of these two solutions.

2 Piecewise Linear Approximations

In this section, we formally define the polygonization problem and discuss some issues faced by computational solutions for this problem.

Algorithms for computing a polygonal approximation to a surface are known under the generic name of *piecewise linear methods*, or simply *PL methods*. The resulting piecewise linear surface is called a *polygonization* of the regular surface.

The polygonization problem can be compared to the problem of carving a precious stone: we have to carve planar faces with a minimum loss of material. Mathematically, this problem can be stated in the following way:

Polygonization Problem. *Given a surface M, find a polygonal surface $\widetilde{M}$, such that $\widetilde{M}$ has the same topology as M and $\widetilde{M}$ approximates the geometry of M, as close as desired.*

More precisely, we have:

- $\widetilde{M}$ is a piecewise linear manifold [3];
- There exists a homeomorphism $h\colon M \to \widetilde{M}$ (by definition, this guarantees that $\widetilde{M}$ and M have the same topology);
- For every point p in M, we have $d(p, h(p)) < \varepsilon$, where d is the ordinary euclidean distance in the ambient space $\mathbf{R}^3$, and $\varepsilon > 0$ is a pre-selected tolerance (this guarantees that the geometry of $\widetilde{M}$ approximates the geometry of M within the tolerance ε).

The error condition means that the polygonal surface is contained within a tubular neighborhood of radius ε of the regular surface, provided that ε is sufficiently small (see Figure 1).

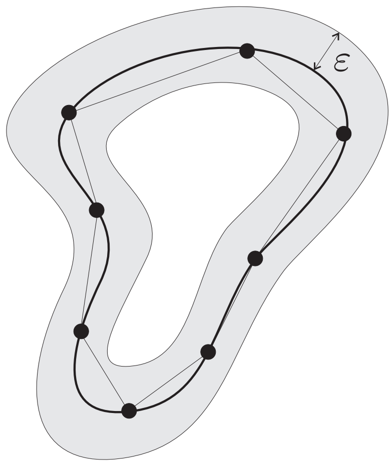


Figure 1: Polygonization and tubular neighborhood.

We say that a surface polygonization is *optimal* if it has the smallest number of polygons among all polygonizations that satisfy the same approximation error. Finding optimal polygonal approximation with the *exact* minimum number of polygons required for a given accuracy is a very difficult problem (probably NP-hard), and we must rely on good heuristics.

Trying to minimize the size of the approximation is important, but some applications have further requirements on the geometry of each polygon. For instance, long, thin polygons should be avoided in finite-element mesh generation and terrain modeling.

A related problem is *decimation*: reduce the number of polygons in an existing polygonal approximation while remaining within the chosen tolerance [4]. Optimal decimation seems to be just as difficult as optimal approximation.

2.1 Building a Polygonal Approximation

Polygonization methods incorporate two basic operations: *sampling* and *structuring*.

The sampling operation uses the surface geometry to generate a set of points on the surface. The structuring operation reconstructs a polygonal surface from the samples. This polygonal surface has the same topology of the original surface and approximates its geometry, as described above.

Sampling points on a parametric surface is a simple matter: just evaluate the parameterization function on a sample of the domain. As explained in Section 2.2, this domain sampling usually provides a natural structure for the samples on the surface.

Sampling points on an implicit surface is not so easy: conceptually, several non-linear equations in three variables must be solved. However, as explained in Section 2.2, this sampling can be done indirectly, by sampling the ambient space around the surface.

2.2 Polygonization Methods

The simplest strategy for polygonizing a parametric surface $f: U \subset \mathbf{R}^2 \to \mathbf{R}^3$ uses an intrinsic approach: Subdivide the domain U and use the function f to construct the polygonal approximation for the surface from the domain subdivision. The rationale behind this method is that, for a sufficiently fine triangulation of the domain, the surface f can be approximated by an affine surface $\tilde{f}$ that coincides with f on the vertices of the triangulation. The most usual case occurs when the domain U is a rectangular region of the plane $\mathbf{R}^2$, subdivided into a regular triangular mesh (i.e., all of the triangles in the mesh are congruent, but possibly in different orientations). In this case, we say that the polygonization is *uniform*. Figure 2 illustrates this for a parametric curve $f: U \subset \mathbf{R} \to \mathbf{R}^2$.

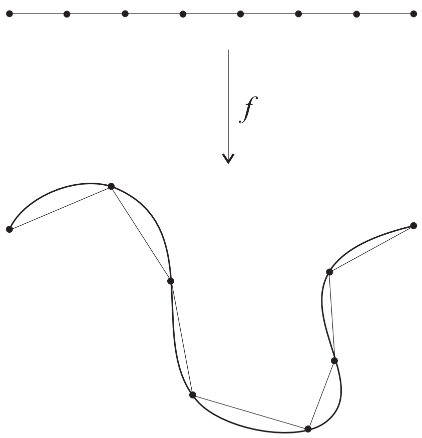


Figure 2: Uniform approximation of a parametric curve.

The simplest strategy for polygonizing an implicit surface $f: V \subset \mathbf{R}^3 \to \mathbf{R}$ uses an extrinsic approach: Subdivide the ambient space V into simplicial sub-cells $V = \cup V_i$, and compute a polygonal approximation to the surface inside each sub-cell V_i. The rationale behind this method is that, for sufficiently small spatial sub-cells, the solution of the equation $f = 0$ is approximated by the solution of the corresponding affine equation $\tilde{f} = 0$, where $\tilde{f}$ is an affine approximation to f inside the cell (there is a unique affine function $\tilde{f}$ that coincides with f on the vertices of the cell). The most common case occurs when the domain V is a cubic region of the space $\mathbf{R}^3$, subdivided into a regular cubic mesh (i.e., all cubes in the mesh are congruent). These cubes are then subdivided into tetrahedra, to obtain a simplicial decomposition of V. In this case, we say that the polygonization is *uniform*. Figure 3 illustrates this for an implicit curve $f: V \subset \mathbf{R}^2 \to \mathbf{R}$.

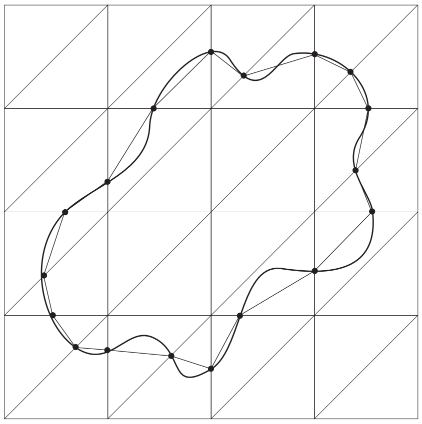


Figure 3: Uniform approximation of an implicit curve.

3 Adaptive Polygonization

Uniform polygonization methods are simple to understand and easy to implement. The connectivity of the uniform mesh provides the structure for a polygonal mesh approximating the surface. There are, nevertheless, some interesting optimizations to avoid computing f more than once at each vertex of the mesh. For example, one can see a regular mesh as a completely balanced quadtree, and then traverse it down to the lowest level, where the cells reside, while computing f exactly once at each vertex.

Based on the rationale behind the two methods described above, it is apparent that if a surface has high frequency details (large variations), then we need many small polygons to approximate it; on the other hand, if the surface has only low frequency details (small variations), then we should be able to obtain good approximations even with few, large polygons.

In general, the frequency details in a surface vary across the domain. Therefore, in uniform polygonizations, we are forced to use a large number of polygons everywhere in the mesh in order to obtain good approximations in the high-frequency regions.

A better alternative to uniform decomposition is *adaptive polygonization*, in which the sampling rate (and consequently the polygon size) varies across the domain according to the frequency variation of the surface.

In this paper, we are concerned with adaptive polygonizations of parametric and implicit surfaces. We shall use the curvature as a geometric measure of surface variation. Therefore, the adaptive polygonization methods we describe below sample the domain finely in regions where the curvature variation is high, and coarsely in regions where the curvature has a low rate of variation.

Adaptive methods try to produce an optimal polygonization, i.e., to obtain a good approximation of the surface within a prescribed accuracy, using the minimum number of polygons. Recall that finding the *exact* minimum number of polygons is very hard, and that we must rely on good heuristics. Thus, adaptive methods define practical criteria for optimization and heuristics for finding optimal polygonizations according to these criteria.

3.1 Adaptation Mechanisms

Adaptive polygonization algorithms are more complex than uniform algorithms because they must solve two interdependent problems:

- perform optimal sampling;
- ensure correct structuring.

Optimal sampling guarantees a faithful geometric approximation, and depends on the adaptation criteria (i.e., curvature). Correct structuring guarantees the global topological consistency of the polygonal mesh and depends on the structuring mechanism.

The main difficulty is that when changes are made *locally* to the sampling rate, the mesh topology may be affected *globally*. Therefore, it is necessary to carefully synchronize the solution of these two problems; otherwise, the polygonization may exhibit holes caused by wrong connectivity in the mesh (also called *topological inconsistency*). For example, different levels of subdivision in two adjacent cells can create a crack along the boundary between them (Figure 4). In this case, cracks show up in the surface and the piecewise approximation is not continuous. This problem is caused by a topologically inconsistent decomposition (Figure 5).

3.2 Classification Criteria

Adaptive polygonization algorithms can be classified according to the way that they implement the basic operations of sampling and structuring, while solving the approximation and consistency problems.

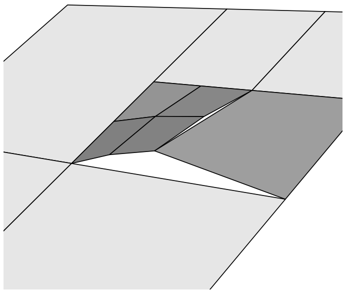


Figure 4: Surface cracks due to topological inconsistency in the quadtree decomposition shown in Figure 5 [5].

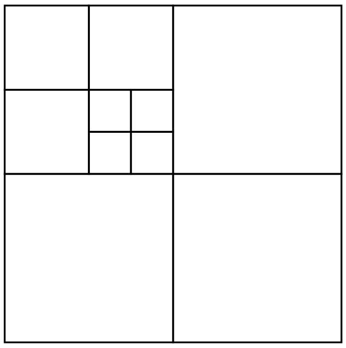


Figure 5: Topologically inconsistent quadtree domain decomposition.

There are two basic strategies to obtain a correct structuring of the cells: perform sampling and structuring in a single step, or in separate steps. When done in a single step, there are two main methods:

1. restricting a hierarchical subdivision; or
2. exploiting cell coherence.

The first approach uses a restricted spatial subdivision. The subdivision is maintained balanced with repeated refinement steps that modify a single cell at each step. Whenever a cell is divided, its neighbors are constrained to be at levels immediately above or below [5].

The second approach focuses on the common boundary elements of the subdivision. In the case of parametric surfaces, these elements are the edges of a 2D hierarchical mesh (quadtree); in the case of the implicit surface, these elements are the faces of a 3D hierarchical mesh (octree). The cells are subdivided independently, constraining the polygonization along common boundary elements, according to the adaptation criteria. In that way, a consistent decision can be made when splitting neighbor cells sharing the same boundary element [6].

When sampling and structuring are done in separate steps, the main issue is how to pass information between the two stages of the algorithm in order to guarantee topological consistency.

Another strategy is to use an algorithm that does not enforce correct structuring, but fixes the inconsistencies in a post-processing step. Typically, such post-processing implies the construction of a topological data structure representing the polygonal approximation, so that inconsistencies in neighboring faces can be found [7].

4 Previous Work

In this section, we give an overview of existing polygonization methods and compare them with a new methodology for adaptation based on the conceptual framework presented in Sections 2 and 3.

4.1 Algorithms for Parametric Surfaces

Previous algorithms for adaptive polygonal approximation of parametric surfaces subdivide patches recursively by splitting edges at their midpoints and creating internal edges to connect these sample points, until the surface patches are flat within a tolerance.

Three approaches have been used to obtain correct structuring, thus eliminating cracks: Clark in his pioneering work [8] uses cell coherence; vonHerzen and Barr [5] use restricted quadtrees; Tamminen and Jansen [7] perform post-processing.

The main deficiency of these algorithms is in the control of the adaptation process. Recursive subdivision is controlled by flatness tests for surface patches that only exist for special classes of surfaces [9-16], with a single exception to the use of interval arithmetic [17].

4.2 Algorithms for Implicit Surfaces

Previous adaptive polygonization algorithms for implicit surfaces employ a full 3D adaptive partition of space, and perform sampling and structuring in a single step. They recursively subdivide a large 3-dimensional cell complex in which the implicit surface is embedded, until the adaptation criteria are met; at that stage, polygons are generated for the cells intersecting the surface.

Correct structuring is achieved either by using a restricted octree [18, 19] or by imposing coherence across the faces of a tetrahedral subdivision [20].

The main deficiency of these algorithms is their high complexity. Strictly speaking, the 3-dimensional search is necessary for computing a correct solution in the case of surfaces with arbitrary topology. Unfortunately, such a requirement places an enormous burden in actual programs, both in terms of implementation and performance. This is mainly due to the 3D combinatorics of the problem, as well as the need of space and time resources. Recently, however, range analysis techniques [21] have been used successfully to avoid full enumeration of implicit surfaces in high-resolution meshes [22-25].

Another problem that needs to be faced when computing polygonal approximations for implicit surfaces is the presence of singularities. Although this problem could be present in parametric surfaces too, it is much harder for implicit surfaces, especially in a design environment [26]. As mentioned in the introduction, the discussion in this paper is restriced to regular surfaces, which do not have singularities.

4.3 A New Strategy for Adaptation

We propose a general methodology to solve the adaptive polygonization problem. Using this methodology we are able to develop two new algorithms for the approximation of parametric and implicit surfaces. These two polygonization methods will be described respectively in Sections 5 and 6. They share three main principles:

- start the adaptive decomposition from a coarse uniform decomposition of the domain;
- decouple sampling from structuring;
- exploit edge coherence to guarantee topological consistency by construction.

With this methodology, unlike in previous algorithms, the polygonization problem is solved in two separate steps that correspond to the operations of sampling and structuring.

The coupling of sampling and subdivision imposes restrictions on the structure of the decomposition. This requires complex algorithms and often results in sub-optimal polygonal meshes. By eliminating such a restriction we are able to develop simpler and more efficient methods.

Note that this strategy does not involve any post-processing. Instead, sampling and structuring are linked through edge coherence ensuring in this way the topological consistency of the mesh.

5 Polygonization of Parametric Surfaces

We now present an adaptive polygonization algorithm for parametric surfaces that uses complete edge sampling to avoid cracks, and area scanning to guide the subdivision [27]. This algorithm combines adaptive curve sampling with simplicial subdivision. This strategy allows better adaptation, trivially ensures global consistency, and produces meshes with a small number of polygons.

The basic algorithm is as follows:

1. [Initialization] Start with a coarse uniform simplicial decomposition of the parametric patch domain. If the domain is rectangular, this can be simply the subdivision of the domain along its diagonal into two triangular cells.

2. [Curve generation] Sample the edges of all cells in the initial decomposition adaptively to construct a polygonal approximation of the corresponding curves on the surface.

3. [Cell subdivision] Subdivide each cell by constructing internal edges, based on the number of points on the boundary edge curves. Repeat the sampling in step 2 for each new internal edge.

4. [Test for flatness] For each cell, test the corresponding surface patch for flatness.

5. [Recursion] Recursively subdivide cells whose patches are not flat. A flat patch is one whose edges are flat, according to the sampling in step 2.

Unlike previous methods for parametric surfaces, edge sampling is completely done at each subdivision step, while creating new edges in step 3. Further subdivisions respect this sampling. This is the key factor for optimal sampling and global consistency. Because edge curves are generated first, in a single operation, it is possible to find the minimum number of sample points that produces the desired approximation. The approximation criterion is given by the flatness test. Moreover, global consistency is automatically guaranteed, because edge curves are shared by adjacent cells.

Sections 5.1 and 5.2 below contain details of steps 2 and 3 of the algorithm.

5.1 Curve Generation

The edges in the domain decomposition are straight line segments, but the corresponding curves on the surface are not. The goal of the sampling in step 2 is to build an adaptive polygonal approximation of these curves.

There are many methods for adaptive sampling of parametric curves [15, 28-31]. In our implementation, we use *multiple random probing*, an extension of single random probing [31], because it is easy to implement and has many degrees of freedom to help achieve a good sampling [27]. This method implements a stochastic search for splitting the edge at the point farthest from the straight line segment joining the extremes of the curve.

This procedure can be seen as a heuristic for finding the critical points of the curve, e.g., the points of maximum curvature. Such a subdivision scheme is specially important in the early stages of the subdivision, when the edges are long and the surface can oscillate significantly from one extreme to the other.

The method tries to compute a polygon whose distance to the actual curve on the surface is at most the desired tolerance ε. As we shall see below, these polygonal approximations are the basis of a heuristic flatness test for surface patches.

5.2 Cell Subdivision

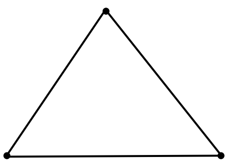
To subdivide each triangular cell, we classify its edges according to the number of sample points created during the curve generation step of the algorithm. A *simple edge* is composed of a single linear segment; it corresponds to a flat curve segment on the surface. A *complex edge* contains interior sample points, and therefore it is composed of several linear segments corresponding to a polygonal curve on the surface. The subdivision of the cells is based on the classification of each cell edge as simple or complex. There are four possible cases (Figure 6):

1. *three simple edges*: Recursive subdivision terminates and the procedure outputs one triangle, corresponding to a flat surface patch (Figure 6a).

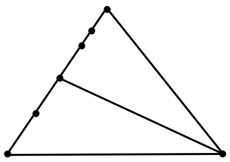
2. *two simple edges*: The cell is divided into two sub cells. A new internal edge is created by choosing an internal point in the complex edge and connecting it to the opposite vertex (Figure 6b).

3. *one simple edge*: The cell is divided into two sub cells. A new internal edge is created by connecting an internal point in one of the complex edges to the opposite vertex (Figure 6c).

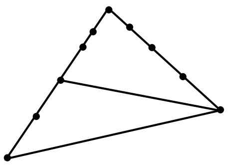
4. *no simple edge*: The cell is divided into four sub cells by generating new three internal edges connecting internal points between each pair of adjacent complex edges (Figure 6d).



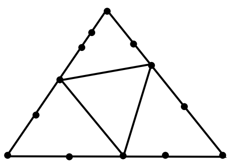
(a) three simple edges.



(b) two simple edges.



(c) one simple edge.



(d) no simple edge.

Figure 6: Four cases for cell subdivision in the path-based adaptive method: three simple edges (a); two simple edges (b); one simple edge(c); no simple edge(d).

Note that the flatness test in case 1 is heuristic: A triangular surface patch is considered flat when all the edges of the corresponding cell are flat. This test works well in practice, but, if necessary, it can be combined with further heuristic tests, such as testing the distance of the center of the triangular face to the point on the surface corresponding to this center. If this distance is not less than the tolerance ε, then the cell is declared not flat and can be further subdivided, say at the midpoints of its edges, as in case 4.

5.2.1 Creation of Internal Edges

In the cell subdivision process described above, the choice of the internal splitting edge is not unique. The algorithm uses the following criteria to make a choice.

For each potential splitting edge, we polygonize the corresponding curve on the surface using the algorithm described in Section 5.1. We choose the edge that corresponds to the curve with the smallest number of polygonal segments. This criterion minimizes the total number of segments in internal edges, and is a good heuristic for determining curves with low curvature variations.

These choices made by the algorithm in the subdivision step is a key factor for global optimal sampling. In fact, the choices provide the best subdivision of each patch: good accuracy of the approximation and small number of polygons.

5.3 Comments

One drawback of the current implementation of the method is that it is slow, due to the exhaustive search used to locate curves with low variation of the curvature inside each cell. A dynamic programming approach could probably be used to avoid recomputing candidates edges in later subdivisions. On the other hand, the cost of the method as described here may be appropriate for applications that do not change the surfaces, such as high-quality rendering.

5.4 Examples

We now show four examples of the method in action. For each example, we show the parameter domain decomposition and a 3-dimensional image of the polygonal approximation, rendered with Gouraud shading.

Cylinder. Figure 7 shows the cylinder given by:

$$x = \cos u, \quad y = \sin u, \quad z = v$$
$$u \in [0, 3.12], \quad v \in [0, 1].$$

The polygonization is built along lines that are parallel to the main axis of the cylinder. These lines correspond to paths of minimum curvature on the surface, which are, in this case, straight line segments. The other lines — the boundaries and the diagonal — are part of the initial triangulation.

Saddle. Figure 8 shows the "saddle" given by:

$$x = u, \quad y = v, \quad z = (uv)^3$$
$$u \in [0, 1], \quad v \in [0, 1].$$

Note how the polygonization adapts to the surface's geometry. The flat center is covered by only six triangles. The curved sides form a ruled structure in which the triangles are aligned transversally to the steepest directions.

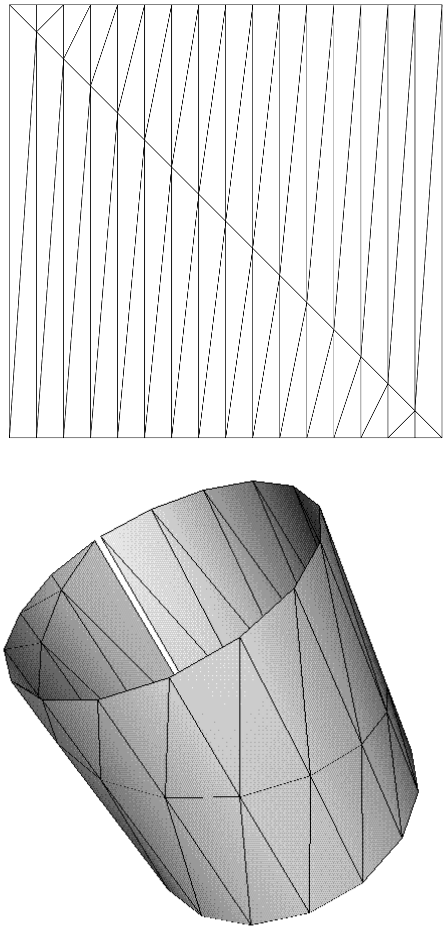


Figure 7: Domain decomposition (top) and polygonal approximation (bottom) for the cylinder given by $f(u,v) = (\cos u, \sin u, v)$, where $u \in [0, 3.12]$ and $v \in [0, 1]$.

Mountain. Figure 9 shows the "mountain" given by:

$$x = u, \quad y = v, \quad z = (\sin u \sin v)^4$$
$$u \in [1.5, 2.7], \quad v \in [0.75, 1.65].$$

Here we can see that the polygonization follows contour lines of the surface as in a topography map. These curves are level sets parallel to the base xy-plane. Besides their importance in geometric approximation, contour lines are perceptually insightful because they depict the surface's height variations.

Needles. Figure 10 shows the "needles" given by:

$$x = u, \quad y = v, \quad z = 0.8 \sin u \sin v$$
$$u \in [1.33, 11.33], \quad v \in [10.25, 21.25].$$

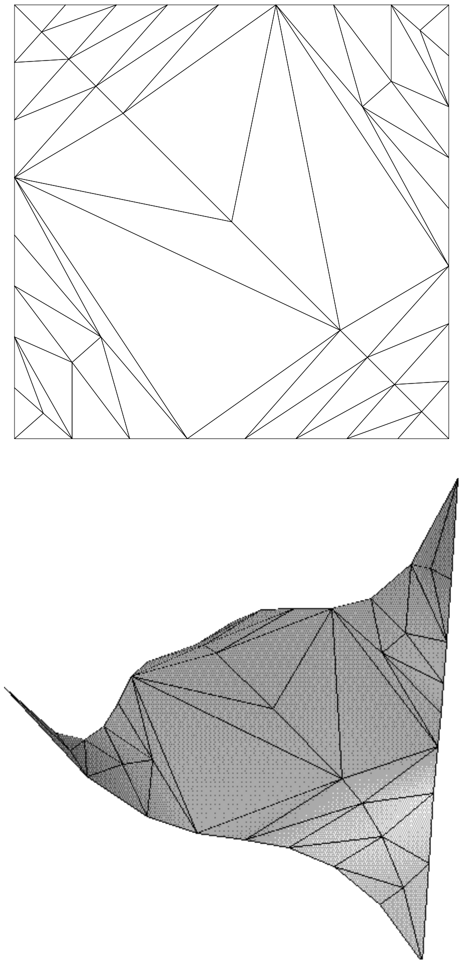


Figure 8: Domain decomposition (top) and polygonal approximation (bottom) for the saddle given by $f(u,v) = (u, v, (uv)^3)$, where $u \in [0,1]$ and $v \in [0,1]$.

This surface has both high frequency detail and sharp variations. Even though the algorithm started with just two triangles covering the entire parametric domain, it was able to capture all important features in the surface. Moreover, the polygons are uniformly distributed around those features.

6 Polygonization of Implicit Surfaces

In this section, we present an adaptive polygonization method for implicit surfaces [32]. The algorithm is easy to implement and is very efficient both in storage and execution time. For this reason, it provides a practical solution for real-world applications.

The basic algorithm uses both an extrinsic and intrinsic subdivision approach:

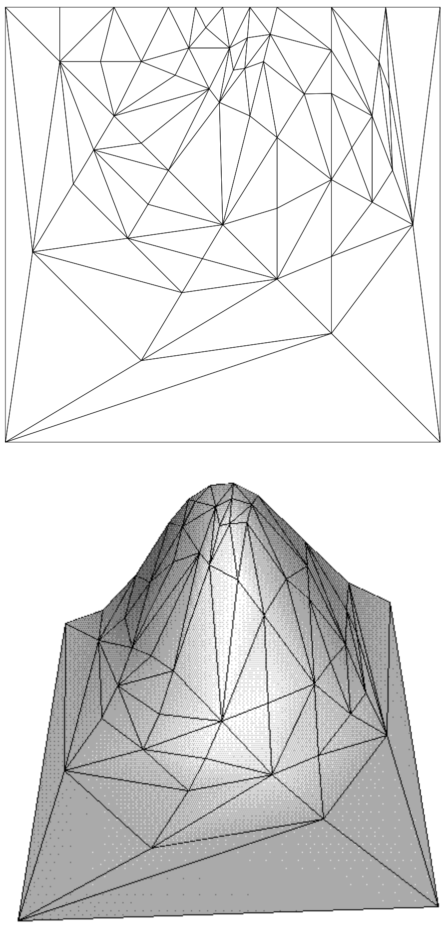


Figure 9: Domain decomposition (top) and polygonal approximation (bottom) for the mountain given by $f(u,v) = (u, v, (\sin u \sin v)^4)$, where $u \in [1.5, 2.7]$ and $v \in [0.75, 1.65]$.

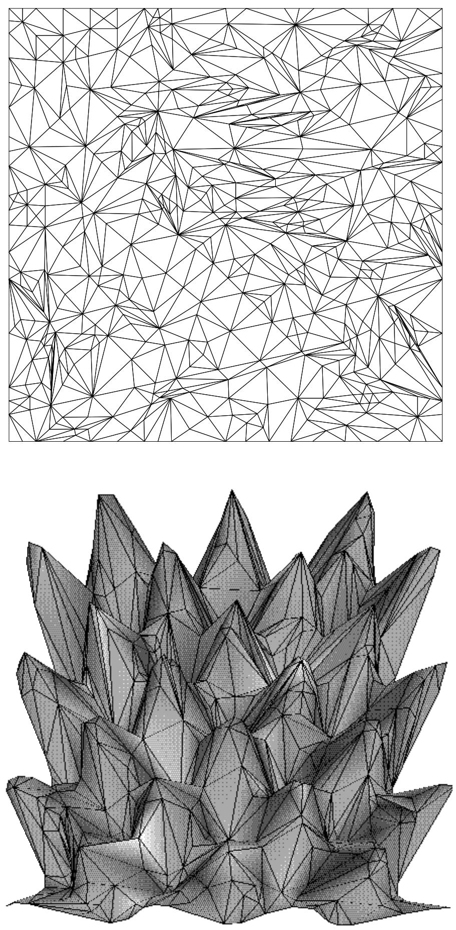


Figure 10: Domain decomposition (top) and polygonal approximation (bottom) for the needles given by $f(u,v) = (u, v, 0.8 \sin u \sin v)$, where $u \in [1.33, 11.33]$ and $v \in [10.25, 21.25]$.

1. [Initial Mesh] Start with a coarse uniform decomposition of the domain of the implicit function and compute a polygonal approximation to the surface (extrinsic step).

2. [Cell Subdivision] Refine the polygonal mesh from the previous step using edge coherence (intrinsic step).

3. [Test of flatness] For each cell, test the corresponding implicit surface for flatness.

4. [Recursion] Recursively subdivide cells whose corresponding implicit surface is not flat.

The key to simple adaptation is the separation between structuring and sampling. Structuring is done in step 1, where a coarse sampling is also performed. In the step 2, the structure of the initial mesh guides the adaptive sampling and, at the same time, provides the connectivity information for maintaining geometrical consistency.

The first step takes place in a 3-dimensional space (i.e., in the region delimited by the domain of the implicit function), while the second step is performed intrinsically (i.e., using the polygonal mesh approximating the implicit surface). This reduction in the dimensionality of the problem makes the computation very efficient.

The flatness test uses the implicit surface curvature. The curvature is estimated over the corresponding cell using the deviation of the surface normal from the normal of the polygon's support plane.

Below, we explain each step in detail.

6.1 Initial Mesh

This process corresponds to an uniform polygonization of the implicit surface, which is a well understood problem with many alternative solutions available in the graphics literature [33-36]. We employ a polygonization algorithm based on a simplicial space decomposition [37]. This is an elegant solution for the problem and it is a perfect match for our mesh adaptation method because of its simplicity and conciseness.

The uniform polygonization algorithm decomposes the bounding box of the implicit function domain using a simplicial cell complex. Our simplicial decomposition employs the classical Coxeter-Freudenthal space subdivision scheme [38], which is illustrated in Figure 11 for the 2D case. For the 3D case, see [37].

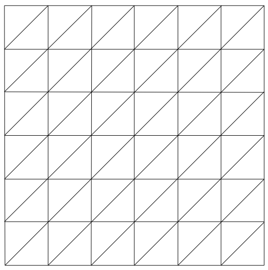


Figure 11: Coxeter-Freudenthal decomposition of the square in $\mathbf{R}^2$.

If the resolution of the space decomposition is adequate, a surface-cell intersection can be determined from the values of the implicit function f at the vertices of the cell. This is because the surface defined by $f(x, y, z) = 0$ intersects a cell only if the value of f changes from positive to negative within the cell, as indicated in Figure 12 for the 2D case.

When the surface intersects a cell, it is approximated by an affine mapping inside the cell, thus generating a new polygon that is added to the mesh. Polygon vertices are the intersection points of the implicit surface with the edges of the cell (see the dashed line in Figure 12).

6.2 Cell Subdivision

A triangle, with vertices v_0, v_1, v_2, is subdivided into four triangles by splitting its edges, $e_0 = v_0v_1$, $e_1 = v_1v_2$, and $e_2 = v_2v_0$, at their midpoints — m_0, m_1 and m_2 — and connecting the midpoints of adjacent edges. A diagram illustrating this scheme is depicted in Figure 13.

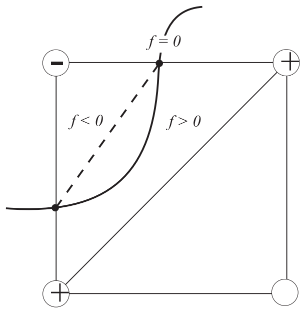


Figure 12: Intersection of a 2D cell with an implicit curve.

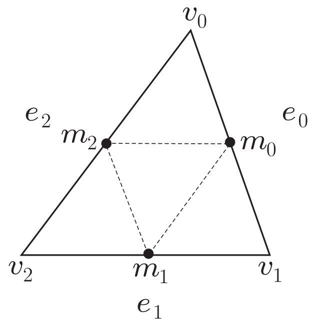


Figure 13: Subdivision scheme of a triangle.

The decision to subdivide the triangle is based on a classification of the surface curvature along its edges. If all edges are flat, then no subdivision is done, and the algorithm outputs that triangle. If one or more edges are not flat, then the current polygonal approximation the triangle must be subdivided. In this case, the edge midpoints are computed and the procedure is called recursively for each piece.

6.2.1 Edge Coherence

The most important part of the adaptation process, the edge coherence mechanism, is implemented in the computation of edge midpoints. If a triangle $v_0v_1v_2$ is not flat, then it must be subdivided. After computing the true midpoint m_i of the linear segment $e_i = v_iv_j$, we have two options to avoid cracks:

1. Always project the midpoint m_i onto the implicit surface and subdivide the adjacent polygon by connecting m_i to the opposite vertex. This has

the advantage of producing a PL-manifold structure, but would require a more complex scheme to guarantee the topological consistency of the mesh.

2. Project the midpoint m_i onto the surface only if the edge e_i is not flat. In this case, for flat edges, no cracks are created even if we do not split the adjacent polygon. This solution has the advantage of generating fewer polygons and is adequate if a PL structure is not required.

We use the second option in our implementation because of its simplicity and efficiency.

6.2.2 Projection

The projection of the midpoint m_i onto the implicit surface amounts to computing the point p_0 such that $f(p_0) = 0$ and, at the same time, minimizes $||p_0 - m_i||$ (this problem has a unique solution because the point m_i is inside a tubular neighborhood of the surface).

The solution of such an optimization problem can be computed using various numerical techniques, such as gradient descent. We employ a physics based method that is simple to implement and provides good accuracy control (see [39] for details).

6.3 Comments

This new hybrid extrinsic/intrinsic polygonization method for implicit surfaces has one limitation compared with methods that employ a full extrinsic scheme. If the initial sampling is too coarse in relation to the complexity of the implicit shape, the polygonization may not capture the correct topology of the surface. This is because structuring is done in 3D during the first pass. At that stage, the method commits to a topology and proceeds on to the second pass, which works in 2D and changes just the geometry. However, this is not a serious problem because there is usually enough information about the surface to determine the appropriate sampling rate for the initial step of the algorithm.

The performance gains are significant because the initial mesh computation, which is expensive at high resolution, needs to be executed only once – at a fixed, coarse resolution – while the intrinsic recursive subdivision is a fast operation.

6.4 Examples

We now show two examples of the method in action.

Blobby Object. The first example is a blobby object [40]. This type of model defines the implicit function as a density field generated from a point skeleton. The implicit surface is a level set of that field. In such a formulation, point sources can be combined either additively or subtractively in order to produce smooth blends.

We chose this implicit model because this is a common technique to construct implicit surface models in computer graphics modeling systems.

The test object is a spherical shape with a cavity on top. It was constructed using a skeleton with two points consisting of one strong positive source and one weaker negative source.

Figure 14 shows a polygonal approximation of this object. The mesh contains 2351 triangles.

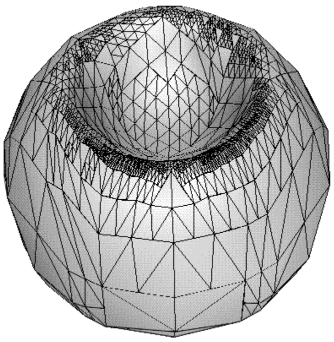


Figure 14: Polygonal approximation of blobby object.

Note that the density of the mesh increases mostly around the rim of the crater, where the subtracted material blend occurs. This area contains the sharpest variations in surface curvature.

Figure 15 shows a close up view of a portion of the triangle mesh with high polygon density.

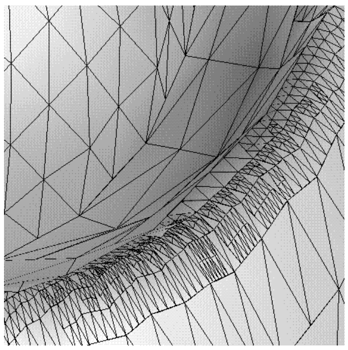


Figure 15: Detail of an area with high surface curvature.

Hypertexture Object. The second example is a hypertexture object [41]. This type of model is a procedural implicit shape, defined by functional composition of a base density function with density modulation functions.

In the example, we have a "noisy sphere" of radius 1, generated from a spherical field modulated band-limited noise. Figure 16 shows the polygonal approximation produced by our algorithm. The initial sampling employed a $3 \times 3 \times 3$ grid enclosing the object. Figure 17 shows a Gouraud-shaded rendering of the object. The mesh has 1058 triangles and was computed in 0.9 seconds in a SGI Indigo R4000 workstation.

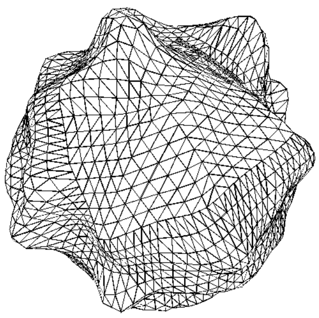


Figure 16: Polygonal approximation of the "Noisy Sphere".

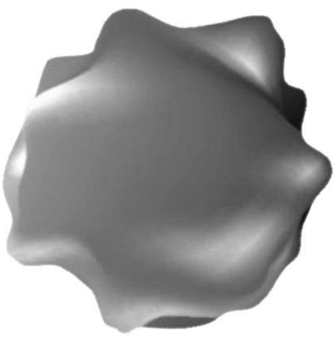


Figure 17: Gouraud-shaded "Noisy Sphere".

7 Conclusions

We have presented two new adaptive polygonization methods: one for parametric surfaces and one for implicit surfaces. Both methods are based on the same general principles: edge coherence; and separation of sampling and structuring,

The method for parametric surfaces combines recursive simplicial subdivision of the domain with point sampling along the corresponding curves on the surface. Cracks in the polygonal mesh are avoided by determining the optimal sampling rate along the edges of a cell before subdividing it. This method is suitable for surfaces with low variations, such as bicubic patches, as well as for surfaces with high variations, such as height fields. The method is also suitable for computing trimmed surfaces: just sample the trimming curves as described in Section 5.1 and start from a triangulation of the trimmed domain constrained to this sampling.

The method for implicit surfaces is a simplification of the method introduced by one of the authors [20]. The main difference is that the original method employed an extrinsic 3D adaptation, making the implementation more complex and less efficient. In the new method, we decompose the problem in two subproblems, so that most of the computation can be done intrinsically (2D). This makes the implementation simpler and more efficient. A disadvantage of this method is that to ensure a correct solution, the resolution of the sampling grid must be fine enough to discriminate the topology of each connected component of the object. In practice, a very coarse sampling grid suffices for most shapes of interest.

A similar polygonization method for implicit surfaces was developed independently by Thad Beier of Pacific Data Images (PDI) [42] and by Brian Wyvill [43].

We are currently working on a unified adaptive polygonization method based on edge coherence which computes a hierarchical polygonization, and handles both parametric and implicit surfaces in the same way.

Acknowledgements

The figures in Sections 5.4 and 6.4 were generated with Geomview [44].

References

[1] M. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[2] L. Velho and J. Gomes. Approximate conversion of parametric to implicit surfaces. *Computer Graphics Forum*, 15(5):327–337, 1996.

[3] C. P. Rourke and B. J. Sanderson. *Introduction to Piecewise Linear Topology.* Springer-Verlag, 1982.

[4] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):65–70, July 1992.

[5] B. Von Herzen and A. H. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):103–110, July 1987.

[6] L. Velho. Interactive modeling of soft objects. In *Proceedings of Ausgraph '90*, pages 227–230, 1990.

[7] M. Tamminen and F. W. Jansen. An integrity filter for recursive subdivision meshes. *Computers and Graphics*, 9(4):351–363, 1985.

[8] J. H. Clark. A fast algorithm for rendering parametric surfaces. In K. I. Joy, C. W. Grant, N. L. Max, and L. Hatfield, editors, *Tutorial: Computer Graphics: Image Synthesis*, pages 88–93. Computer Society Press, 1988.

[9] J. Lane and L. Carpenter. A generalized scan line algorithm for the computer display of parametrically defined surfaces. *Computer Graphics and Image Processing*, 11:290–297, 1979.

[10] J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):35–46, 1980.

[11] P. A. Koparkar and S. P. Mudur. Computational techniques for processing parametric surfaces. *Computer Vision, Graphics, and Image Processing*, 28(3):303–322, 1984.

[12] D. Filip. Adaptive subdivision algorithms for a set of Bézier triangles. *Computer Aided Design*, 18(2):74–78, 1986.

[13] R. D. Clay and H. P. Moreton. Efficient adaptive subdivision of Bézier surfaces. In *Proceedings of Eurographics '88*, pages 357–371, September 1988.

[14] D. R. Forsey and R. V. Klassen. An adaptive subdivision algorithm for crack prevention in the display of parametric surfaces. In *Proceedings of Graphics Interface '90*, pages 1–8, May 1990.

[15] M. Kosters. Curvature-dependent parametrization of curves and surfaces. *Computer Aided Design*, 23(8):569–578, 1991.

[16] J. W. Peterson. Tessellation of NURB surfaces. In P. Heckbert, editor, *Graphics Gems IV*, pages 286–320. Academic Press, 1994.

[17] S. P. Mudur and P. A. Koparkar. Interval methods for processing geometric objects. *IEEE Computer Graphics and Applications*, 4(2):7–17, 1984.

[18] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[19] M. Hall and J. Warren. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–43, 1990.

[20] L. Velho. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints. In *Proceedings of Eurographics '90*, pages 125–136, September 1990.

[21] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions.* Ellis Horwood Ltd., 1984.

[22] K. G. Suffern and E. D. Fackerell. Interval methods in computer graphics. *Computers & Graphics*, 15:331–340, 1991.

[23] J. M. Snyder. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):121–130, July 1992.

[24] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):131–138, July 1992.

[25] L. H. de Figueiredo and J. Stolfi. Adaptive enumeration of implicit surfaces with affine arithmetic. *Computer Graphics Forum*, 15(5):287–296, 1996.

[26] C. L. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-Patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.

[27] L. Velho and L. H. de Figueiredo. Optimal adaptive polygonal approximation of parametric surfaces. In *Proceedings of SIBGRAPI '96*, pages 127–133, October 1996.

[28] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[29] M. Crampin, R. Guifo, and G. A. Read. Linear approximation of curves with bounded curvature and a data reduction algorithm. *Computer Aided Design*, 17(6):257–261, 1985.

[30] R. E. Chandler. A recursive technique for rendering parametric curves. *Computers and Graphics*, 14(3/4):477–479, 1990.

[31] L. H. de Figueiredo. Adaptive sampling of parametric curves. In A. Paeth, editor, *Graphics Gems V*, pages 173–178. Academic Press, 1995.

[32] L. Velho. Adaptive polygonization made simple. In *Proceedings of SIBGRAPI '95*, pages 111–118, October 1995.

[33] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):163–169, August 1987.

[34] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[35] J. Bloomenthal. An implicit surface polygonizer. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, 1994.

[36] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis*, 22:322–346, 1985.

[37] J. Gomes and L. Velho. *Implicit Objects in Computer Graphics*. Number 53 in IMPA Monograph Series. IMPA, Rio de Janeiro, 1993.

[38] H. Coxeter. *Regular Polytopes*. Macmillan, New York, 1963.

[39] L. H. de Figueiredo and J. Gomes. Sampling implicit objects with physically-based particle systems. *Computers & Graphics*, 20(3):365–375, 1996.

[40] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[41] K. Perlin and E. Hoffert. Hypertexture. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):253–262, July 1989.

[42] T. Beier, 1990. (personal communication).

[43] B. Wyvill. Explicating implicit surfaces. In *Proceedings of Graphics Interface '94*, pages 165–173, May 1994.

[44] Geomview. Software written at the Geometry Center, University of Minnesota. Available at `http://www.geom.umn.edu/software/`.