A Unified Approach for Hierarchical Adaptive Tesselation of Surfaces

Luiz Velho, Luiz Henrique de Figueiredo, and Jonas Gomes
Visgraf Laboratory, IMPA–Instituto de Matemática Pura e Aplicada

This paper introduces a unified and general tesselation algorithm for parametric and implicit surfaces. The algorithm produces a hierarchical mesh that is adapted to the surface geometry and has a multiresolution and progressive structure. This representation can be exploited with advantages in several applications.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.6 [**Computer Graphics**]: Methodology and Techniques.; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design (CAD)

Additional Key Words and Phrases: Geometric modeling, surface approximation, polygonization, parametric surfaces, implicit surfaces, multiresolution representations, adapted meshes.

1. INTRODUCTION

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

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

1.1 Motivation

The main drawback of the polygonal representation is that it generally requires a large number of polygons to faithfully describe the geometry of complex curved surfaces. In part, this is due to the piecewise linear nature of the polygonal mesh,

Address: Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, RJ, Brazil. lvelho,lhf,jonas@visgraf.impa.br.

but it is also due to the use of uniform meshes.

Fortunately, the effectiveness of polygonal representations can be much improved if adapted and hierarchical meshes are used instead of uniform meshes. *Adapted meshes* conform to the surface geometry and ideally have the smallest number of polygons among all polygonizations with the same level of approximation. *Hierarchical meshes* organize nested polygonizations with increasing number of polygons into a multiresolution structure that approximates the surface from coarse to fine levels of detail.

The combination of adapted and hierarchical polygonal meshes is a very effective form of piecewise linear surface representation. For this reason, there is great interest in algorithms for computing such representations.

1.2 Previous Work

Previous algorithms for adaptive polygonization are based either on a restricted hierarchical decomposition [Von Herzen and Barr 1987; Bloomenthal 1988; Hall and Warren 1990; Vlassopoulos 1990; Schmidt 1993] or on recursive subdivision with cell coherence [Clark 1988; Beier 1990; Velho 1990; Wyvill 1994; Velho 1996]. The first approach has the advantage of producing a combinatorial manifold structure [Rourke and Sanderson 1982], but the algorithms are complex and do not always generate a small number of polygons. The second approach uses simpler algorithms and produces fewer polygons, but it does not generate a combinatorial manifold structure, which is necessary in several applications. Moreover, these methods are dependent on the surface description, requiring separate algorithms for parametric and implicit surfaces. Some algorithms rely on tests that only exist for special classes of surfaces [Lane and Carpenter 1979; Lane and Riesenfeld 1980; Filip 1986; Clay and Moreton 1988; Forsey and Klassen 1990; Kosters 1991; Peterson 1994].

1.3 Contribution

In this paper, we introduce a new surface tesselation algorithm that improves previous solutions in several ways. Our method is general and works with any standard form of surface description, parametric or implicit. The method generates polygonizations that are adapted to surface geometry. The resulting meshes possess a combinatorial manifold structure and approximate the surface with a small number of polygons. The algorithm is efficient, simple to implement, and easily parallelizable. Moreover, it provides a hierarchical representation that incorporates the main characteristics of multiresolution [Eck et al. 1995] and progressive meshes [Hoppe 1996].

1.4 Organization of the Paper

The organization of the remainder of the paper is as follows: Section 2 explains the mechanisms behind polygonization algorithms, and the basic strategies to construct adapted meshes; Section 3 gives an overview of our polygonization method; Section 4 considers the different alternatives to generate the initial mesh to be refined; Section 5 describes a scheme to compute a multiresolution representation of curved segments on the surface; Section 6 presents a mesh refinement strategy based on multiresolution edges; Section 7 discusses how to exploit the hierarchical structure of the polygonization and relates it to other multiresolution schemes; Section 8 shows several examples of using our polygonization algorithm to approximate parametric and implicit surfaces; and finally Section 9 concludes with final remarks and future work.

2. BACKGROUND

Polygonization methods must implement two basic operations: sampling and structuring [de Figueiredo 1992; Gomes and Velho 1993]. *Sampling* is the computation of a set of points on the surface. *Structuring* is linking those points into a mesh. Thus, sampling deals with geometry, whereas structuring deals with topology. Algorithms can be classified according to how they implement these two basic operations.

2.1 Uniform Polygonization

The simplest polygonization algorithms are based on uniform decompositions. They employ a cell complex [Glaser 1970] made of elements of the same size. The function that describes the surface geometry is evaluated at node points of a regular grid underlying the decomposition, and the samples obtained are assembled using adjacency relations from the cell complex. Algorithms of this type are straightforward to implement, but produce polygonal meshes that are not adapted to the surface. Typically, uniform polygonizations have either too few or too many polygons. This solution is acceptable only for shapes with regular features, whose surface curvature is almost constant. Even then, the polygons may vary widely in size, as in the familiar case of a sphere parametrized with the usual spherical coordinates.

2.2 Adaptive Polygonization

A better solution is to construct an adapted polygonization, in which the sampling rate varies spatially according to local surface complexity. In principle, one can produce the minimum number of polygons required to approximate the surface within a desired precision. However, this problem is probably NP-hard, and so heuristics are unavoidable. Algorithms for adaptive polygonization typically subdivide an initial uniform mesh recursively until one or more adaptation criteria are met; at that stage, the final polygonal mesh is generated. The recursive nature of the process makes it possible to produce a multiresolution polygonization: at each recursion level, a polygonal mesh is generated along with the parent-child correspondences in the subdivision hierarchy.

2.3 Adaptation Mechanisms

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

—perform optimal sampling everywhere;

—ensure correct structuring across subdivision levels.

Optimal sampling guarantees a faithful geometric approximation, and depends on the adaptation criteria. Correct structuring guarantees the topological consistency of the mesh and depends on the subdivision mechanism.

The main difficulty in adaptive methods is that, when changes are made *locally* to the sampling rate, the mesh topology may be affected *globally*. Therefore, it is necessary to synchronize the solution of these two problems; otherwise, the polygonization may exhibit holes ("cracks") caused by incorrect connectivity [Clark 1988]. For example, different levels of subdivision in two adjacent cells can result in a crack along their common boundary.

2.4 Maintaining Consistency

As mentioned in Section 1.2, there are two approaches for the synchronization of sampling and structuring: with a restricted decomposition; or through cell coherence.

In restricted decompositions, the hierarchical subdivision is maintained balanced under refinement iterations that modify a single cell at time. Whenever a cell is divided, its neighbors are constrained to be at most one level above or below it. In the end, a combinatorial manifold structure may be derived from the hierarchy in a post-processing step that typically uses templates [Von Herzen and Barr 1987].

The cell coherence approach focuses on the boundary elements of the decomposition. Cells are subdivided independently, constraining the polygonization along common boundary elements according to the sampling criteria. In that way, a consistent decision is made when splitting neighbor cells sharing the same boundary element. A manifold structure cannot be enforced because cells are subdivided independently of each other.

Previous adaptive polygonization algorithms perform sampling and structuring in a single step. The coupling of sampling and structuring imposes restrictions on the resulting decomposition. This requires complex algorithms and often results in sub-optimal polygonal meshes. By eliminating this coupling, we are able to develop a simpler and more efficient algorithm.

3. OVERVIEW OF THE METHOD

We now give an overview of our adaptive polygonization method and its main operations. The algorithm combines hierarchical curve sampling with simplicial subdivision. This strategy allows better adaptation, ensures global mesh consistency by construction, and produces a multiresolution structure with a small number of polygons at each level.

3.1 Operations

The algorithm is composed of three independent operations:

Base Mesh Generation: Generate a simplicial uniform decomposition of the surface. The input is a surface description; the output is a mesh of triangular cells. The base mesh should capture the correct surface topology, but it needs only be a rough geometric approximation (See Section 4).

Edge Sampling: Adaptively sample an edge of the mesh to construct a hierarchical approximation of the corresponding curve on the surface. Each edge is sampled exactly once. The input is the edge endpoints; the output is a multiresolution curve representation (See Section 5).

Cell Structuring: Subdivide a cell based on the number of points on its boundary edge curves, by creating sampled internal edges and structuring new cells. The input is a triangular cell; the output is a list of triangular subcells (See Section 6).

Note that global knowledge about the surface is necessary only for Base Mesh Generation. Edge Sampling just requires information about local surface geometry, whereas Cell Structuring deals just with the local topology of surface patches. This is exploited by the algorithm to partition the problem in a simple and effective way.

3.2 The Algorithm

The basic algorithm is as follows:

1 [INITIALIZATION]
Start with a coarse decomposition of the surface:
1.1 Generate the base mesh;
1.2 Sample the edges of all cells in the base mesh.

2 [REFINEMENT]
For each cell, test the corresponding surface patch for flatness.
If the patch is not flat, then recursively subdivide the cell:
2.1 Structure new cells by constructing internal edges;
2.2 Sample all internal edges.

The procedures implementing structuring and sampling operations communicate through a well-defined interface: the *edge* (Section 5) and *cell* (Section 6) data structures.

3.3 Analysis

The key to simple adaptation is the separation of structuring and sampling. These two operations are linked through the *edge* data structure. Unlike previous methods, edge sampling is completely done when new edges are created: in the initialization phase (for edges of the initial mesh); and in the refinement phase at each subdivision step (for internal edges of a cell). Further subdivisions respect this sampling. *Edges are never resampled.* The use of edge coherence is the basis for adaptive sampling and topological consistency. Because edge curves are generated first, in a single operation, it is possible to find the minimum number of sample points that produces an approximation within the desired tolerance. Moreover, global consistency is automatically guaranteed, because edge curves are shared by adjacent cells. Therefore, *cracks are avoided by construction*, not corrected a posteriori [Von Herzen and Barr 1987].

Observe that sampling is performed in one-dimensional space (i.e., along curves on the surface). This reduction in the dimensionality of the problem makes the geometric computation more efficient, transforming the concern with topology into a trivial matter. On the other hand, by concentrating all geometric computation on edge sampling, cell structuring becomes much simpler, because for this operation one needs to deal only with topological issues.

4. GENERATING THE BASE MESH

The generation of the base mesh is the only operation of the algorithm that needs complete knowledge about the surface. As we shall see in Section 8, this kind of knowledge is desirable, since it can be exploited for generating a topological structure that best suits specific application constraints.

The initial mesh has to capture just the correct surface topology, without any concerns about geometric adaptation. This is because global structuring is done during the initialization phase, while generating the base mesh. At that stage, the method commits to a topology and proceeds on to the refinement phase, which just modifies local structure and geometry.

The base mesh can be uniform and very coarse. Its edges correspond to curves on the surface, which are going to be adaptively sampled. The techniques used to create this initial mesh depend on surface type. The next subsections describe some particular solutions for parametric and implicit surfaces.

4.1 Base Meshes for Parametric Surfaces

Generating the base mesh for parametric surfaces $f: U \subset \mathbf{R}^2 \to \mathbf{R}^3$ consists essentially in triangulating a 2D region. This is a trivial task in most cases, specially when the parametric domain U is a rectangle. In this case, the parametrization function f is evaluated at the corners of the rectangle, and the base mesh is generated by subdividing the domain along one of its diagonals into two triangles. Even this coarsest mesh is adequate in many cases.

The simplicity of this task makes it possible to incorporate additional constraints that could help solving application-dependent problems. For instance, one can impose a segmentation of the domain based on features defined by the application. Some examples are discontinuity lines and trimming curves (Section 8). Note that the edges of the base mesh can be curves even in the parametric domain. The only restriction is that all the points of an edge have to be visible by the opposite vertices of the triangles sharing that edge [Preparata and Shamos 1985].

4.2 Base Meshes for Implicit Surfaces

Generating the base mesh for implicit surfaces $f: V \subset \mathbf{R}^3 \to \mathbf{R}$ is a more difficult task. There is no direct way to generate points on the surface $f^{-1}(0)$, and for this reason, both mesh topology and geometry must be computed indirectly by sampling the 3D region V in which the surface is contained.

This problem can be solved using an uniform polygonization method [Allgower and Schmidt 1985; Lorensen and Cline 1987; Wyvill et al. 1986; Gomes and Velho 1993; Bloomenthal et al. 1997], as discussed in Section 2. Such methods employ a regular grid to decompose V into 3D cells, which are then intersected with the surface to generate triangles of the base mesh. The only requirements imposed by our algorithm is that the sampling grid has to be fine enough to capture the correct surface topology, and the edges of the base mesh are contained inside a tubular neighborhood of the implicit surface. A method for computing a polygonal mesh that captures the correct topology of an implicit surface is given in [Stander and Hart 1997] (see also [Wilhelms and Gelder 1990; Pasko et al. 1988]). The tubular neighborhood requirement, needed for correct edge sampling, is usually not difficult to enforce and verify.

In practice, a very coarse sampling grid suffices for most implicit shapes of interest. Knowledge about the surface should be used to determine the appropriate uniform sampling rate. For a compact surface of genus 0 (i.e., a shape homeomorphic to the sphere), the grid size should be smaller than the diameter of the largest sphere inscribed in the shape. For example, a $2 \times 2 \times 2$ grid in the case of a sphere.

An alternative method for generating the base mesh uses a polyhedron that is known to have the same topology and similar proportions as the implicit surface. This method is well suited for simple shapes, and works as follows: First, the mesh topology is created by triangulating the faces of the polyhedron. Then, the mesh geometry is determined by projecting the vertices of the mesh onto the implicit surface. This is done by positioning and scaling the polyhedron such that it is near the surface, and then, for each vertex, computing the point closest to it on the surface (as described in Section 5). For example, a tetrahedron could be used in the case of a sphere [Christensen 1978].

5. EDGE SAMPLING

The edges in the cell decomposition correspond to curves on the surface. The goal of edge sampling is to build an adapted, multiresolution approximation of these 3D curves. The edge data structure and its associated procedures are one of the most important components of our method. Other components rely on this information to make all important decisions in the algorithm.

5.1 The Edge Data Structure

The edge data structure stores a multiresolution representation of the curve segment on the surface that connects the endpoints of an edge. It is a hierarchical structure composed of two parts: a *base segment* and a *subdivision tree*.

The base segment structure contains: the edge endpoints p and q; a pointer to the root of its subdivision tree S; and the total error e incurred by the curve approximation. Here it is, in pseudo C:

```
struct Edge
   Point      p, q;
   TreeNode  *S;
   Real       e;
```

The subdivision tree structure is a binary tree whose nodes contain: the *split point* t of the parent curve segment (see Section 5.3); the vector d from the segment midpoint m to the split point t; pointers L and R to the left and right subtrees, which represent the two parts of the curve split at t; and the partial error e of this branch of the subdivision tree, defined as $\max(e_L, e_R)$, where e_L and e_R are the partial errors of the left and right subtrees, respectively. Here it is, in pseudo C:

```
struct TreeNode
   Point      t;
   Vector     d;
   TreeNode  *L, *R;
   Real       e;
```

The diagram in Figure 1 shows the geometric elements of an edge. Note that the split point t is redundant information, since it can be derived from d and the parent edge segment. Nevertheless, it is kept in the data structure for efficiency reasons.

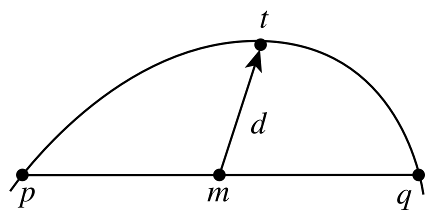


Fig. 1. Geometric elements of an edge.

Figure 2 depicts a graph illustrating the hierarchical structure of an edge. Note that these data structures make it very easy to perform most operations on edges. For example, determining whether the edge is a straight line segment or not amounts simply to testing the pointer S.

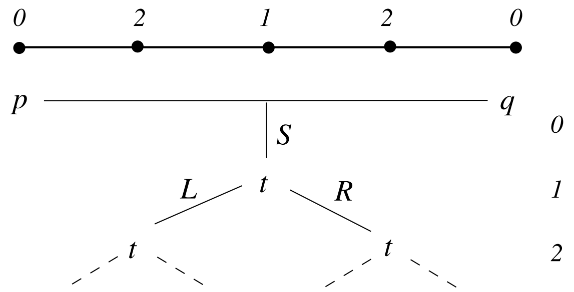


Fig. 2. Hierarchical structure of an edge.

5.2 Constructing an Edge

The edge representation is constructed using a combination of hierarchical sampling and simplification. This scheme generates a multiresolution, adapted, piecewise-linear approximation of the corresponding curve on the surface.

The hierarchical sampling procedure recursively subdivides the edge pq at its midpoint m, and finds a new sample, the split point t on the surface (Section 5.3). It also computes the difference between these two points, the vector $d = t - m$. Recursion terminates when the (user-defined) maximum sampling density is reached. The result is a dense hierarchical uniform sampling of the curve, which is passed on to the simplification procedure. Figure 3 illustrates this process showing the set of sample points on a curve and the corresponding edge subdivision tree.

The simplification procedure adapts the edge subdivision tree to the curve geometry, eliminating branches that can be approximated well by linear segments. The tree is visited in depth-first order, and a node is removed when the node siblings have already been removed and the length of the vector d is less than a user-defined tolerance ε.

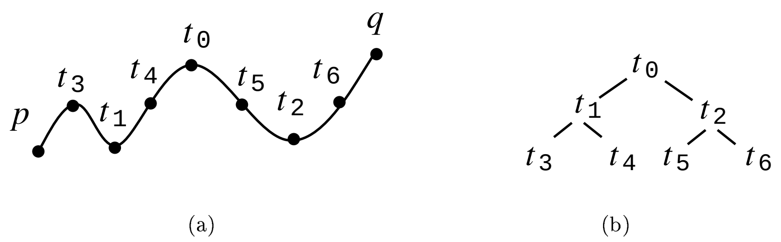


Fig. 3. Hierarchical uniform sampling of a curve: sample points (a) and full binary subdivision tree (b).

Figure 4 shows the effect of simplification on the curve in Figure 3. Note that the hierarchical structure of the samples is maintained.

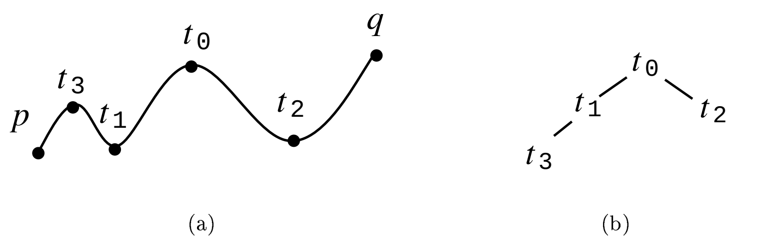


Fig. 4. Hierarchical adaptive sampling of a curve: sample points (a) and subdivision tree (b).

This simplification strategy guarantees that the polygonal curve approximation is contained inside a tubular neighborhood of the surface (provided that the user-defined uniform sampling rate is appropriate to avoid aliasing). Figure 5 shows a schematic drawing of the curve approximation and some related measures.

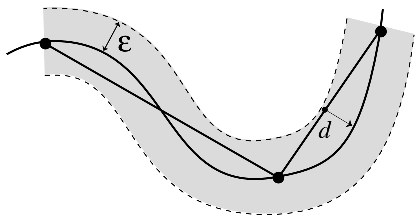


Fig. 5. Curve approximation and tubular neighborhood.

An estimate of the total accumulated approximation error at each branch of the subdivision tree is computed during the edge construction process. For a leaf node, it is simply the norm $\|d\|$ of the vector $d = t - m$. For an internal node, it is given by $\max(\|d\|, e_L, e_R)$. [1] The computation is propagated from the bottom upwards the tree and corresponds to the error incurred by replacing that part of the curve with a linear segment. We remark that the hierarchical uniform sampling density is set to twice the maximum edge resolution. In that way, the approximation error estimate is meaningful even at the finest level of resolution. This information is important for the local optimization done in the structuring operation (Section 6), and can be also used for progressive display of the mesh (Section 7).

5.3 Sampling Points on Parametric and Implicit Surfaces

The only knowledge required about surface geometry for the edge sampling operation is how to compute the split point of a curve segment on the surface. Therefore, we can encapsulate all surface-dependent code in a single procedure that, given the curve endpoints p and q, returns the split point t of the curve. Ideally, the split point should be the curve midpoint, with respect to arc length, so that the trees are nicely balanced. However, finding the exact midpoint is usually too hard and not worth the computational cost. In practice, any point on the curve that is near the midpoint is adequate.

In the case of parametric surfaces, $f\colon U \subset \mathbf{R}^2 \to \mathbf{R}^3$, this procedure is very simple to implement. First, the (u, v) coordinates of the edge midpoint in parameter space are computed: $(u_t, v_t) = (\frac{u_p+u_q}{2}, \frac{v_p+v_q}{2})$. Then, the 3D coordinates of the corresponding point t on the curve are obtained by evaluating the parametrization function f at (u_t, v_t), that is, $t = f(u_t, v_t)$.

In the case of implicit surfaces $f\colon V \subset \mathbf{R}^3 \to \mathbf{R}$, the procedure is more complex. We need to find the point on the implicit surface $f^{-1}(0)$ that is closest to the edge midpoint m. The problem amounts to computing the point t such that $f(t) = 0$ and, at the same time, minimizes the distance $\|t - m\|$. This is an optimization problem that can be solved using a numerical technique, such as gradient descent. However, we employ a physically-based method that is simple to implement and provides good accuracy control [de Figueiredo and Gomes 1996]. In its simplest version, the method is

$$
\begin{array}{l}
\text{while } |f(x)| > \varepsilon \\
\quad y \leftarrow f(x) \\
\quad x \leftarrow x - \delta\, \mathrm{sign}(y) \nabla f(x) \\
\quad \text{if } \mathrm{sign}(y) \neq \mathrm{sign}(f(x)) \text{ then } \delta \leftarrow \delta/2
\end{array}
$$

In words, the point x moves towards the surface by taking small steps δ in the direction of the gradient, halving the step size every time it crosses the surface. The point x stops when it is close enough to the surface, as measured by the value of f at x. A similar method was proposed in [Bloomenthal and Wyvill 1990].

5.4 Using Additional Information

The schemes described above for sampling curves on surfaces are the best one can do without additional information on f, i.e., by treating f as a black-box. If detailed information about f is known, then much better tests are possible, and the

[1] A more accurate but expensive estimate is to compute the one-sided Hausdorf distance from the edge midpoint m to the set of sample points on the curve.

possibility of aliasing can be drastically reduced, if not eliminated. Such tests exist for special classes of surfaces, such as Bézier surfaces [Lane and Carpenter 1979; Lane and Riesenfeld 1980; Filip 1986; Clay and Moreton 1988; Forsey and Klassen 1990; Kosters 1991; Peterson 1994]. For more general surfaces, one can use interval arithmetic [Mudur and Koparkar 1984], if the function f is given by a formula or even by an algorithm.

5.5 Relation with Wavelets

Our hierarchical sampling method is closely related to wavelets and other multiresolution schemes.

The hierarchical sampling procedure is analogous to the wavelet transform, in the sense that it builds a decomposition of the edge curve into *averages* — the midpoints m of linear segments, and *differences* — the vector d from m to the point t on the curve. These averages correspond to scaling function coefficients and the differences to wavelet coefficients [Schröder and Sweldens 1996].

The simplification procedure, on the other hand, is analogous to a wavelet-based lossy compression, in the sense that it removes redundant information present in those samples that could be predicted approximately using linear interpolation. This is accomplished by eliminating wavelet coefficients of low magnitude, i.e., when $\|d\| < \varepsilon$.

6. CELL STRUCTURING

The cell structuring operation performs a simplicial decomposition. It recursively subdivides a triangular cell into triangular subcells, using information from its edges. For this purpose, new internal edges are created and adaptively sampled (Section 5). The computation is completely independent of surface description (implicit or parametric), since all geometry is contained in the edge data structure and global topology is captured by the base mesh. Because the surface is a manifold, the local topology is encoded in the cell data structure itself.

6.1 Classification of Edges

Cell subdivision is determined by an analysis of the complexity of external edge curves. We classify such edges according to the number of sample points created by the edge sampling operation. 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 corresponds to a polygonal curve on the surface composed of several linear segments. The subdivision of the cells is based on the classification of each cell edge as simple or complex.

6.2 Subdivision Templates

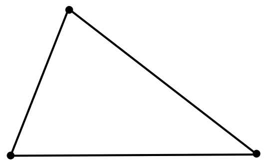
Structuring employs a set of subdivision templates that are generated by combining the types of all three edges of the cell. There are four possible cases (Figure 6):

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

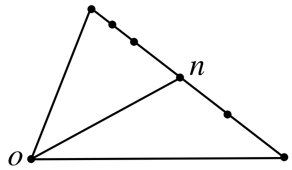
(b) *two simple edges*: The cell is divided into two sub cells. A new internal edge is created by choosing the split point n of the complex edge and connecting it to the opposite vertex o (Figure 6b).

(c) *one simple edge*: The cell is divided into three sub cells. Two new internal edges are created: one connecting the split points n_1 and n_2 of the complex edges, and another connecting one of these split points to the corresponding opposite vertex o_1 or o_2 (Figure 6c).

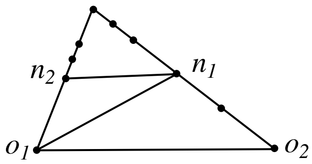
(d) *no simple edge*: The cell is divided into four sub cells by generating three new internal edges connecting the split point n_j of one of the edges, to its opposite vertex o_j, and to the split points of the two other edges (Figure 6d).



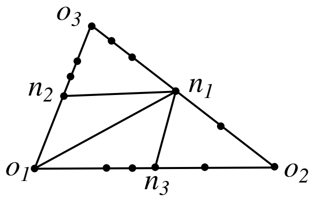
(a) three simple edges.



(b) two simple edges.



(c) one simple edge.



(d) no simple edge.

Fig. 6. Templates for cell subdivision.

6.3 Subdivision and Adaptation

Note that in cases (c) and (d) there is a choice for partitioning the cell. In case (c), the quadrilateral $n_1 n_2 o_1 o_2$ can be subdivided into the triangles $n_1 n_2 o_1$ and $n_1 o_1 o_2$, or into the triangles $n_1n_2o_2$ and $o_2n_2o_1$. In case (d), the triangle $o_1o_2o_3$ can be subdivided into the triangles: $n_1o_3n_2$, $n_1n_2o_1$, $n_1o_1n_3$, and $n_1n_3o_2$; or $n_2o_1n_3$, $n_2n_3o_2$, $n_2o_2n_1$, and $n_2n_1o_3$; or $n_3o_2n_1$, $n_3n_1o_3$, $n_3o_3n_2$, and $n_3n_2o_1$.

To decide which subdivision to make, a local optimization criterion is used. The selected configuration is based on the geometry of the edges, as well as on the aspect ratio of the subcells. In order to do this, all possible edges are generated, and we use the option with smallest approximation error ε, as defined in Section 5, and with aspect ratio closest to 1 (e.g. equilateral triangulation). This strategy corresponds to choosing the best direction to subdivide a cell, and can greatly improve the quality of the final mesh. The effectiveness of this procedure is due to the hierarchical nature of the refinement process. Local decisions at coarse levels of the hierarchy will have a strong influence in the way the surface is subdivided at finer levels. Therefore, they guide the algorithm to find a good global solution. This approach is similar in spirit to multigrid methods.

6.4 Topological Consistency Issues

An important remark related to topological consistency is that the cell structuring mechanism relies on the following assumptions: 1) edges are shared by adjacent cells; 2) cells are coherently oriented. Note that this can be naturally built into the generation of the base mesh, as well as into the process of cell subdivision.

If these two conditions are true, then the topological structure generated by the templates in the previous section is guaranteed to be globally consistent. Indeed, because edges are shared by adjacent cells and are always subdivided at their split points, the algorithm induces a true simplicial complex of increasingly finer resolutions at each recursion level of the process.

We remark that, although decompositions in 6c and 6d could be factored into a sequence of single edge subdivisions as in template 6b, templates 6c and 6d cannot be reduced to the recursive application of template 6b. The reason is that, in order to maintain topological consistency of the mesh, all complex edges must be subdivided at each refinement level.

To simplify the creation and manipulation of the topological data structure, cells and edges have labels derived from point *ids*, which are unique in the system.

6.5 Cell Sampling to Prevent Aliasing

It is well known that, when sampling is not correct, aliasing occurs. This problem is already addressed in the other two operations of our method: base mesh generation and edge sampling, which are responsible for computing sample points on the surface. Therefore, if the base mesh captures the surface topology and edge sampling detects all variations in surface geometry along the corresponding curves, then the hierarchical subdivision process we are describing here will produce a correct sampling on the whole surface. Note that, even though a 2D surface is only being sampled along 1D curves, the surface is sampled along a set of directions that adaptively cover its entire two-dimensional domain.

One possibility remains for the scheme described above to be affected by sampling problems: when all three edges of a cell are simple, but the surface variation inside the cell is still too large. This is handled by performing area sampling within the cell. It amounts to a geometric test in case (a) of the cell subdivision (Figure 6): we compute one additional sample point p, representative of the surface variation in the interior of the cell and measure the distance d from p to its projection on the support plane of the triangular face. If d is greater than ε (the same test of Section 5), then, instead of stopping recursion, the cell is further subdivided by generating three new internal edges from each vertex of the cell to point p (Figure 7). If these new edges are not all simple, then we will have identified a region of high surface variation inside the cell, even though there is no variation on its border.

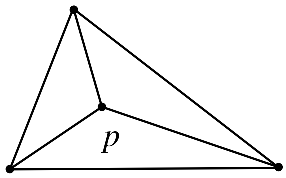
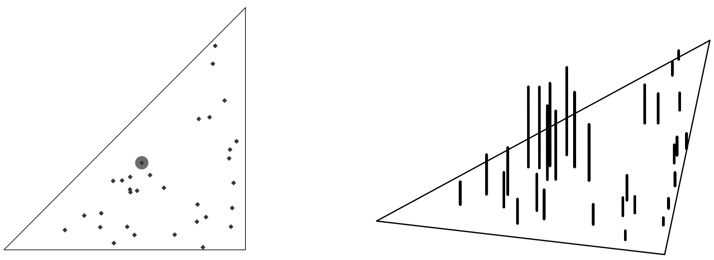


Fig. 7. Extra subdivision of the template in Fig.6a.

The interior point p is chosen using *multiple random probing* [Velho and de Figueiredo 1996], an extension of single random probing [de Figueiredo 1995], and similar to stochastic sampling. Multiple random probing generates many samples at random locations inside the cell and returns the one with largest deviation from the base plane of the cell.

Figure 8a depicts a typical random pattern used to probe a triangular patch. Figure 8b shows the distances from the sample points on the surface to their projections on the base triangle. The sample point most distant from the base triangle –indicated by a large gray dot in Figure 8a– is the representative sample returned by the procedure.



(a) (b)

Fig. 8. Multiple random probing of a triangular surface patch: sample pattern (a) and distances from sample points to base triangle.

There are many ways to proceed this sampling: in our current implementation we employ a pattern of uniformly distributed random points; another effective sample pattern is based on the Poisson disk distribution [Cook 1986].

The user controls the sampling density by specifying the number of random samples per unit area of the base triangle. This parameter is intuitive and closely related to the curve sampling resolution.

We emphasize that multiple random probing is effectively estimating the surface variation over a patch through stochastic sampling. This is the best one can do without more knowledge about the surface. Again, if additional information, such as Lipschitz bounds, is available, then better tests can be devised for the case of template 6a, as mentioned in Section 5.4.

7. MULTIRESOLUTION

Besides generating adapted simplicial meshes that approximate a surface to any desired accuracy, our polygonization algorithm also produces a hierarchical mesh structure that represents the surface at multiple levels of resolution. Such a representation combines the characteristics of multiresolution and progressive meshes, and has many important practical applications.

7.1 Producing a Hierarchical Structure

The algorithm, as presented in the previous sections, produces a triangle mesh that has only one resolution level and is contained inside a prescribed tubular neighborhood of the surface [Cohen et al. 1996]. With simple modifications, this basic algorithm can produce two kinds of hierarchical structures:

—*multiresolution mesh*: a sequence of nested triangle meshes, represented by the recursive subdivision structure, together with the corresponding vertex coordinates and edge refinement vectors [Eck et al. 1995].

—*progressive mesh*: a base triangle mesh and a sequence of vertex split transformations [Hoppe 1996].

Generating a multiresolution mesh is straightforward. The algorithm outputs all intermediate cells at every recursion level in addition to terminal leaf cells. This amounts to a simple change in the cell structuring operation (Section 6). When a cell is subdivided, that cell and the *ids* of its children are written to the output file.

Generating a progressive mesh is slightly more complicated. The algorithm outputs the base mesh and, instead of subdividing cells, it generates mesh transformations caused by the introduction of new vertices. To understand how to do this, first note that a vertex split corresponds to an edge that is subdivided (just imagine that one of its endpoints, v_s, splits in two, v_s and v_t, as shown in Figure 9).

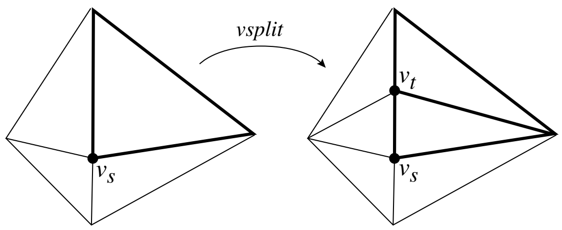


Fig. 9. Vertex split transformation.

This is exactly the case of template (b) in the cell structuring operation shown in Figure 6. Then, observe that templates (c) and (d) can be decomposed respectively into a sequence of two and three vertex split transformations. Pairs of triangles sharing an edge that is subdivided are grouped to form a vertex split transformation. A final remark is that, since the algorithm computes the approximation error of each edge segment, one has all necessary information for sorting the transformations into a progressive sequence within two levels of refinement.

7.2 Exploiting the Hierarchical Structure

The hierarchical structure generated by the algorithm is suitable for virtually all applications of multiresolution and progressive meshes [De Floriani and Puppo 1995; Certain et al. 1996; Hoppe 1996]. Moreover, it is more powerful and flexible because it combines the advantages of *both* these data structures into a single representation.

Examples of modeling and visualization applications that can use our hierarchical mesh structure are:

—tolerance analysis;

—multiresolution editing;

—level-of-detail rendering and compression;

—progressive transmission and display.

7.3 Comparison with Multiresolution and Progressive Meshes

As mentioned previously, our hierarchical adaptive polygonization algorithm is closely related to other approximation methods, such as the wavelet transform, and to simplification methods, such as mesh decimation [Schroeder et al. 1992; Li 1995] and mesh optimization [Hoppe et al. 1993].

The main difference is that all these methods, in one way or another, start with a high resolution mesh, and employ either multiresolution analysis, or some optimization algorithm, in order to construct a simpler adapted mesh (possibly with different levels of detail). In our algorithm, a full resolution mesh is never constructed; instead, only curves that cut the surface in several directions are sampled at the highest resolution.

In comparison with multiresolution meshes, our method has the advantage of allowing meshes with arbitrary connectivity, whereas multiresolution meshes enforce a fixed subdivision connectivity. This allows better approximations and adaptation. On the other hand, multiresolution meshes can use higher-order basis functions [Zorin and Schroder 1996], whereas our method currently only works with linear interpolation.

In comparison with progressive meshes, our method has the advantage of producing a mesh that is both progressive *and* hierarchical. We believe that this fact can be exploited in novel ways in such applications as surface editing and rendering. On the other hand, we do not handle topological changes, such as in [Popović and Hoppe 1997].

8. EXAMPLES

In this section, we show some examples of our algorithm in action.

8.1 Parametric Surface

The first example is a parametric surface having the shape of a "saddle", given by the function

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

Figure 10 shows the polygonization generated with an approximation error of $\varepsilon = 10^{-4}$. The base mesh is the simplest simplicial decomposition: a subdivision of the rectangular domain along its diagonal into two triangles.

Note how the polygonization adapts to the surface's geometry. The flat center is covered by only eight triangles. The curved sides form a ruled structure in which the triangles are aligned transversally to the steepest directions. This adapted mesh has 176 triangles. For comparison, a uniform mesh with the same approximation error would have 2048 triangles.

8.2 Implicit Surface

The next example is an offset implicit surface, defined by a distance function from a polygonal skeleton:

$$f(x, y, z) = \mathrm{dist}((x, y, z), S) - r,$$

where S is the horizontal unit square $[0,1] \times [0,1] \times \{0\} \subset \mathbf{R}^3$, and $r = 0.25$.

Figure 11 shows the polygonization for an approximation error of $\varepsilon = 10^{-5}$. The base mesh has 120 triangles, and was constructed using a $4 \times 4 \times 4$ Coxeter-Freudenthal uniform decomposition of the surface's bounding box [Gomes and Velho 1993]. The final mesh has 1824 triangles, whereas a uniform mesh with the same approximation error would have 122880 triangles.

Observe that, although constrained to producing a manifold structure, the algorithm is able to make a perfect transition between the large polygons on the top, the long polygons on the sides and the small polygons on the corners.

8.3 Comparison between Parametric and Implicit Surfaces

The third example is a torus, a surface of genus 1 that has both parametric and implicit descriptions. This surface was chosen for two reasons: first, it demonstrates that the algorithm can handle surfaces of arbitrary topological type; second, it is used to compare the polygonizations of parametric and implicit surfaces.

In the following comparison, two polygonal approximations of the same torus were generated from its parametric and implicit description. The approximation error was the same in both cases: $\varepsilon = 10^{-3}$.

Figure 12 shows the torus, defined parametrically by

$$x = \cos u(r + \cos v), \quad y = \sin u(r + \cos v), \quad z = \sin v$$
$$u, v \in [0, 2\pi], \quad r = 1.6.$$

On the left side of Figure 12, we show the adapted decomposition of the parameter domain. On the right side, we show the polygonal approximation, which contains 516 triangles. The base mesh was simply the subdivision of the rectangle $[0, 2\pi] \times [0, 2\pi]$ along its diagonal into two triangles.

Note that the algorithm has structured the parameter domain into a restricted hierarchy with three layers. This structure is not explicitly enforced in the subdivision: it is a natural consequence of the adaptation to the curvature of the torus. Note also that, although the triangulation in the parameter space is not at all uniform, the triangulation on the surface is approximately uniform, i.e., all triangles are approximately congruent.

Figure 13 shows the same torus, now defined implicitly by

$$f(x,y,z) = (x^2 + y^2 + z^2 - r^2 - 1)^2 - 4r^2(1 - z^2)$$
$$x, y \in [-3, 3], \quad z \in [-1, 1], \quad r = 1.6.$$

On the left side of Figure 13, we have an orthogonal projection of the base mesh together with the 3D grid; and on the right side, the polygonal approximation, which contains 1324 triangles. The base mesh has 200 triangles, and was constructed using a Coxeter-Freudenthal decomposition on a $4 \times 4 \times 2$ grid. This is smallest sampling grid that can capture the correct topology of the torus.

Note that, in spite of the influence of the base mesh, which is not uniform, the final polygonization is quite uniform and adapts to the surface.

This example shows that the algorithm produces consistent results using either the parametric or implicit description of a surface.

8.4 Trimming and Segmentation with Parametric Surfaces

The next example is a trimmed parametric surface. It demonstrates the capability of the algorithm to handle constraints in the parameter domain that are incorporated into the base mesh.

We use a tensor product Bézier patch with control points:

$$\begin{bmatrix} (-1,0,0) & (0,3,0) & (3,3,0) & (4,0,0) \\ (1,0,1) & (1,-3,1) & (3,3,1) & (2,0,1) \\ (0,0,2) & (0,3,2) & (3,3,2) & (3,0,2) \\ (-1,0,4) & (0,3,3) & (3,-3,3) & (4,2,3) \end{bmatrix}$$

Figure 14a shows the parameter domain with the trimming curve. Figure 14b shows the polygonal approximation. Figures 14c and 14d show the initial and final decomposition of the parameter domain, respectively. The polygonization has 261 triangles and was computed for $\varepsilon = 10^{-3}$.

8.5 Parametric Surface from Sampled Data

The next example is a cylindrical model for a digitized bust of Spock. The original Cyberware data had 87040 points, structured into a regular cylindrical mesh of 174080 triangles with a bounding box of $180 \times 120 \times 180$. We used this mesh as a black-box for a piecewise-linear parametric surface.

Figure 15 shows the polygonization generated with an approximation error of $\varepsilon = 0.3$. This approximation has 11840 vertices and 15921 triangles. This example shows that the method is also useful for data reduction. That is accomplished by effectively resampling the input dense uniform mesh, and creating a more compact adaptive mesh which approximates the surface within a prescribed tolerance.

Figure 16 shows the corresponding domain decomposition. The facial details are clearly visible, because the regions of high curvature are sampled more densely than the rest of the surface.

The base mesh is the simplest simplicial decomposition: a subdivision of the rectangular domain along its diagonal into two triangles.

8.6 High-frequency Implicit Surface

The next example shows that the method works well for highly complicated geometries. Figure 17 shows a Gouraud-shaded rendering of a fractal rock, obtained by modulating a blobby sphere of radius 1 [Blinn 1982] with Perlin's $1/f$ noise [Perlin and Hoffert 1989]. Figure 18 shows the polygonization used in this rendering. The approximation error was $\varepsilon = 10^{-3}$. The base mesh was simply a tetrahedron and the total number of triangles in the polygonal approximation is 5976.

8.7 Gaussian Spike

The next example is a Gaussian spike on a planar surface. It demonstrates that our method can handle isolated surface variations inside a cell. This Monge surface is defined parametrically by

$$\begin{aligned}
&x = u, \quad y = v, \quad z = 4e^{-\frac{u^2+v^2}{2\sigma^2}} \\
&u \in [-3, 2.5], \quad v \in [-1, 4.5], \quad \sigma = 0.125.
\end{aligned}$$

The base mesh consists of two triangles, both having simple (flat) edges. Therefore, the Gaussian spike is detected by multiple random probing inside the cells. We have used a density of 16 sample points per unit area.

Figure 19 shows the polygonal approximation, as well as a Gouraud shaded rendering of it. The mesh has 140 triangles for a error tolerance of $\varepsilon = 10^{-3}$. Figure 20 shows the adapted domain decomposition and a detail of the region corresponding to the Gaussian spike.

8.8 Parametric Surfaces over Warped Domains

The next example shows that the method works well even in the presence of bad parametrizations. Figure 21a shows a uniform polygonal approximation of a torus over warped domain, and Figure 21b shows the uniform domain decomposition of the parametric domain:

$$\begin{aligned}
&x = \cos u'(r + \cos v'), \quad y = \sin u'(r + \cos v'), \quad z = \sin v' \\
&u' = 2\pi u^\lambda, \quad v' = 2\pi v^\lambda \\
&u, v \in [0, 1], \quad r = 1.6, \quad \lambda = 2.8.
\end{aligned}$$

We remark that, to make this example possible, we have used the unit square as the base domain, which does not have the topology of the torus. Thus, our surface is really a rectangular sheet deformed to lie on the surface of the torus.

Figure 22a shows an adaptive polygonal approximation of the same torus, and Figure 22b shows the corresponding decomposition of the parametric domain. Note how the faces are much more uniform now, and follow the geometry of the surface, not the warped geometry of the parametrization. Also note how the domain decomposition adapts by the inverse of the exponential warping function, effectively linearizing the parametrization.

8.9 Multiresolution Model

The last example shows the result of using our algorithm to produce a hierarchical polygonization of an implicit surface. The example is a "blobby" shape. This type of model is defined implicitly by a density field function generated from a point skeleton [Blinn 1982]. The implicit surface is a contour level of that field. In this framework, point sources can be combined additively or subtractively in order to produce smooth blends. The main reason for chosing the "blobby" model is that it allows good control of the local surface curvature through the blending parameters.

We constructed a spherical shape with a cavity on top using a two point skeleton consisting of one strong positive source at $(0,0,0)$ with radius 0.8, and one weaker negative source at $(0.29,0,0.48)$ with radius 0.24. The "blobbiness" parameter is the same for both sources, $b = -6.6$. Figure 23 shows a sequence of polygonal approximations of this shape. It depicts from top to bottom, respectively, the base mesh, coarse, medium and high resolution polygonizations (levels 3, 4 and 5 of the hierarchy). The corresponding polygonal meshes are composed of 20, 340, 1130, and 4580 triangles. On the right, Figure 23 shows visualizations of the object at increasingly closer distances from the camera. The images were rendered at the appropriate level of detail using the meshes on the left.

9. CONCLUSIONS

We have presented a new polygonization algorithm for general surfaces. This algorithm computes a piecewise-linear approximation that is contained within any desired tubular neighborhood of the surface (provided the base mesh and the edge sampling rate are chosen adequately by the user). It also produces a hierarchical mesh that is adapted to the surface geometry and has a multiresolution progressive structure. We have used this same framework in a method for computing hierarchical triangle strips, which are important for efficient rendering [Velho et al. 1999].

9.1 Summary

The method works with parametric, implicit, or any other kind of surface description that provides procedures for: 1) the generation of a coarse triangular mesh; 2) the computation of a point on the surface.

In the case of parametric descriptions, the method supports an arbitrary simplicial segmentation of the domain, which may even contain curved edges.

The algorithm has a simple recursive nature, which makes the code quite simple. It decomposes the 2D sampling and structuring problem into two operations: 1D sampling and 2D structuring, thus making the implementation very efficient in terms of speed and memory.

The basic algorithm and its associated surface representation can be applied with advantage in several applications of geometric modeling, computer graphics and image processing.

9.2 Future Work

We are currently working on various applications of the representation produced by our algorithm. These include the development of a progressive viewer, and a mesh editor. Future work goes in several directions:

—Optimizing the algorithm for the surfaces commonly used in CAGD, e.g., NURBS;

—Converting digital elevation models (DEM) to compact triangular irregular networks (TIN), for representation and compression of terrain models;

—Image representation and compression;

—Exploiting the local adaptation control allowed by the algorithm in the context of space-variant refinement, such as in the schemes employed in flight simulators;

—Further investigating the domain segmentation capabilities of the algorithm. We conjecture that it can be used with realistic rendering methods, such as radiosity;

—Extending our representation to allow higher order interpolation schemes, based on wavelet theory and finite elements.

ACKNOWLEDGMENTS

The figures in Section 8 were generated with Geomview [Levy et al. 1991]. The data for the Spock bust was kindly provided by Hughes Hoppe. We thank the anonymous reviewers for their invaluable comments and suggestions that have greatly improved the presentation of this paper.

The authors are partially supported by research grants from the Brazilian Council for Scientific and Technological Development (CNPq) and Rio de Janeiro Research Foundation (FAPERJ). L. H. de Figueiredo is on leave from LNCC–Laboratório Nacional de Computação Científica, Petrópolis, Brazil.

REFERENCES

Allgower, E. L. and Schmidt, P. H. 1985. An algorithm for piecewise linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis 22*, 2, 322–346.

Beier, T. 1990. Practical uses for implicit surfaces in animation. in Modeling and Animating with Implicit Surfaces. ACM SIGGRAPH Course Notes.

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

Bloomenthal, J. 1988. Polygonization of implicit surfaces. *Computer Aided Geometric Design 5*, 4, 341–355.

Bloomenthal, J., Bajaj, C., Blinn, J., Cani-Gascuel, M.-P., Rockwood, A., Wyvill, B., and Wyvill, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco.

Bloomenthal, J. and Wyvill, B. 1990. Interactive techniques for implicit modeling. *Computer Graphics 24*, 2 (March), 109–116. (1990 Symposium on Interactive 3D Graphics).

Certain, A., Popovic, J., Duchamp, T., Salesin, D., Stuetzle, W., and DeRose, T. 1996. Interactive multi-resolution surface viewing. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 91–98. Addison Wesley.

Christensen, A. H. J. 1978. A note on geodesic polyhedra: Triangulation and contouring of spheres. *Computers and Graphics 3*, 163–165.

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

Clay, R. D. and Moreton, H. P. 1988. Efficient adaptive subdivision of Bézier surfaces. In D. A. Duce and P. Jancene Eds., *Eurographics '88* (Sept. 1988), pp. 357–371. North-Holland.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., AND WRIGHT, W. 1996. Simplification envelopes. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 119–128. Addison Wesley.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Transactions on Graphics 5*, 1 (Jan.), 51–72.

DE FIGUEIREDO, L. H. 1992. *Computational Morphology of Implicit Curves.* Ph. D. thesis, IMPA–Instituto de Matemática Pura e Aplicada.

DE FIGUEIREDO, L. H. 1995. Adaptive sampling of parametric curves. In A. PAETH Ed., *Graphics Gems V*, pp. 173–178. Academic Press.

DE FIGUEIREDO, L. H. AND GOMES, J. 1996. Sampling implicit objects with physically-based particle systems. *Computers & Graphics 20*, 3, 365–375.

DE FLORIANI, L. AND PUPPO, E. 1995. Hierarchical triangulation for multiresolution surface description geometric design. *ACM Transactions on Graphics 14*, 4 (Oct.), 363–411.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95 Conference Proceedings*, Annual Conference Series (Aug. 1995), pp. 173–182. Addison Wesley.

FILIP, D. 1986. Adaptive subdivision algorithms for a set of Bézier triangles. *Computer Aided Design 18*, 2, 74–78.

FORSEY, D. R. AND KLASSEN, R. V. 1990. An adaptive subdivision algorithm for crack prevention in the display of parametric surfaces. In *Proceedings of Graphics Interface '90* (May 1990), pp. 1–8.

GLASER, L. C. 1970. *Geometrical Combinatorial Topology*, Volume I. Van Nostrand Reinhold Company.

GOMES, J. AND VELHO, L. 1993. *Implicit Objects in Computer Graphics.* Number 53 in IMPA Monograph Series. IMPA, Rio de Janeiro.

HALL, M. AND WARREN, J. 1990. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics and Applications 10*, 6, 33–43.

HOPPE, H. 1996. Progressive meshes. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 99–108. Addison Wesley.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. *Computer Graphics 27*, 19–26. (SIGGRAPH '93 Proceedings).

KOSTERS, M. 1991. Curvature-dependent parametrization of curves and surfaces. *Computer Aided Design 23*, 8, 569–578.

LANE, J. AND CARPENTER, L. 1979. A generalized scan line algorithm for the computer display of parametrically defined surfaces. *Computer Graphics and Image Processing 11*, 290–297.

LANE, J. AND RIESENFELD, R. 1980. 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.

LEVY, S., MUNZNER, T., AND PHILLIPS, M. 1991. Geomview. Software written at the Geometry Center, University of Minnesota. Available at http://www.geom.umn.edu/software/.

LI, S. Z. 1995. Adaptive sampling and mesh generation. *Computer Aided Design 27*, 3, 235–240.

LORENSEN, W. E. AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (Aug.), 163–169. (SIGGRAPH '87 Proceedings).

MUDUR, S. P. AND KOPARKAR, P. A. 1984. Interval methods for processing geometric objects. *IEEE Computer Graphics and Applications 4*, 2, 7–17.

PASKO, A. A., PILYUGIN, V. V., AND POKROVSKIY, V. N. 1988. Geometric modeling in the analysis of trivariate functions. *Computers and Graphics 12*, 3/4, 457–465.

PERLIN, K. AND HOFFERT, E. M. 1989. Hypertexture. *Computer Graphics 23*, 253–262. (SIGGRAPH '89 Proceedings).

PETERSON, J. W. 1994. Tessellation of NURB surfaces. In P. HECKBERT Ed., *Graphics Gems IV*, pp. 286–320. Academic Press.

Popović, J. and Hoppe, H. 1997. Progressive simplicial complexes. In T. Whitted Ed., *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series (Aug. 1997), pp. 217–224. ACM SIGGRAPH: Addison Wesley.

Preparata, F. P. and Shamos, M. I. 1985. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY.

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

Schmidt, M. F. 1993. Cutting cubes – visualizing implicit surfaces by adaptive polygonization. *The Visual Computer 10*, 2, 101–115.

Schröder, P. and Sweldens, W. 1996. Building your own wavelets at home. in Wavelets in Computer Graphics. ACM SIGGRAPH Course Notes.

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

Stander, B. T. and Hart, J. C. 1997. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In T. Whitted Ed., *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series (Aug. 1997), pp. 279–286. ACM SIGGRAPH: Addison Wesley.

Velho, L. 1990. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints. In *Proceedings of Eurographics 90* (Sept. 1990), pp. 125–136. Elsevier Science Publisher.

Velho, L. 1996. Simple and efficient polygonization of implicit surfaces. *Journal of Graphical Tools 1*, 2, 5–24.

Velho, L. and de Figueiredo, L. H. 1996. Optimal adaptive polygonal approximation of parametric surfaces. In *Proceedings of SIBGRAPI '96 (Brazilian Symposium on Computer Graphics and Image Processing)* (October 1996), pp. 127–133. http://www.visgraf.impa.br/sibgrapi96/trabs/abst/a04.html.

Velho, L., de Figueiredo, L. H., and Gomes, J. 1999. Hierarchical generalized triangle strips. *The Visual Computer 15*, 1, 21–35.

Vlassopoulos, V. 1990. Adaptive polygonalization of parametric surfaces. *The Visual Computer 6*, 5, 291–8.

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

Wilhelms, J. and Gelder, A. V. November 1990. Topological considerations in isosurface generation extended abstract. *Computer Graphics (San Diego Workshop on Volume Visualization) 24*, 5, 79–86.

Wyvill, B. 1994. Explicating implicit surfaces. In *Proceedings of Graphics Interface '94* (May 1994), pp. 165–173.

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

Zorin, D. and Schroder, P. 1996. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 189–192. Addison Wesley.

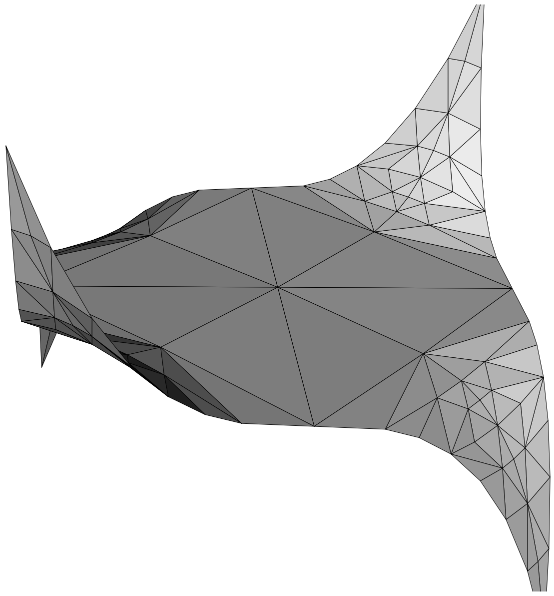


Fig. 10. Parametric surface.

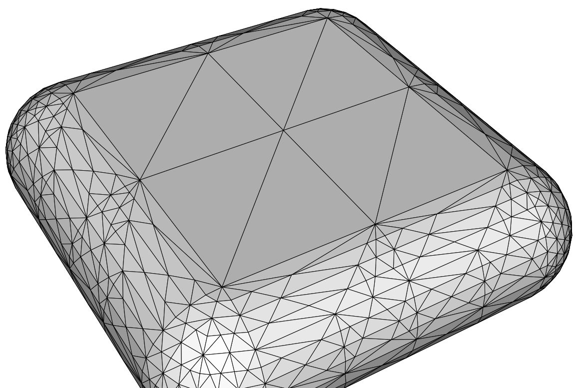


Fig. 11. Offset surface, an example of implicit surface.

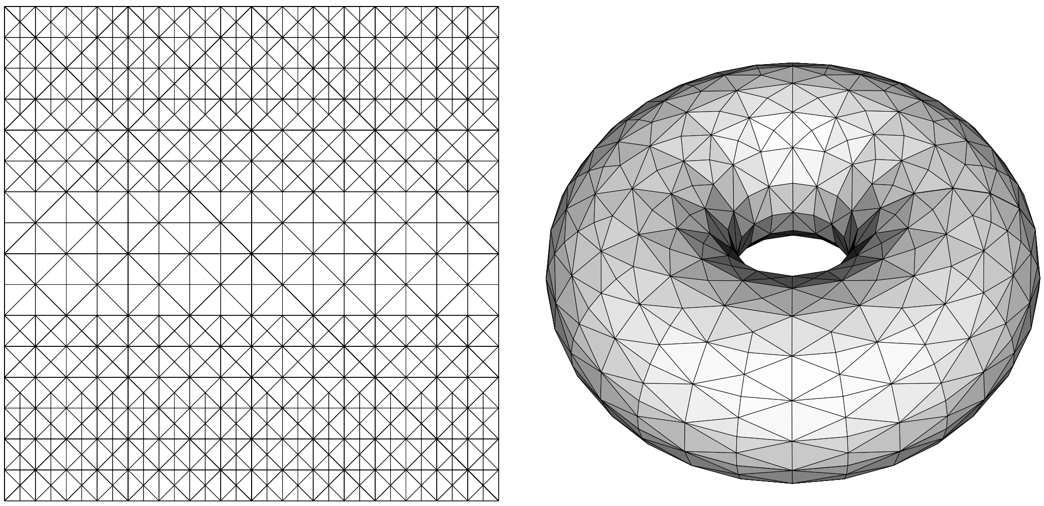


Fig. 12. Parametric torus: parameter domain decomposition (left) and polygonization (right).

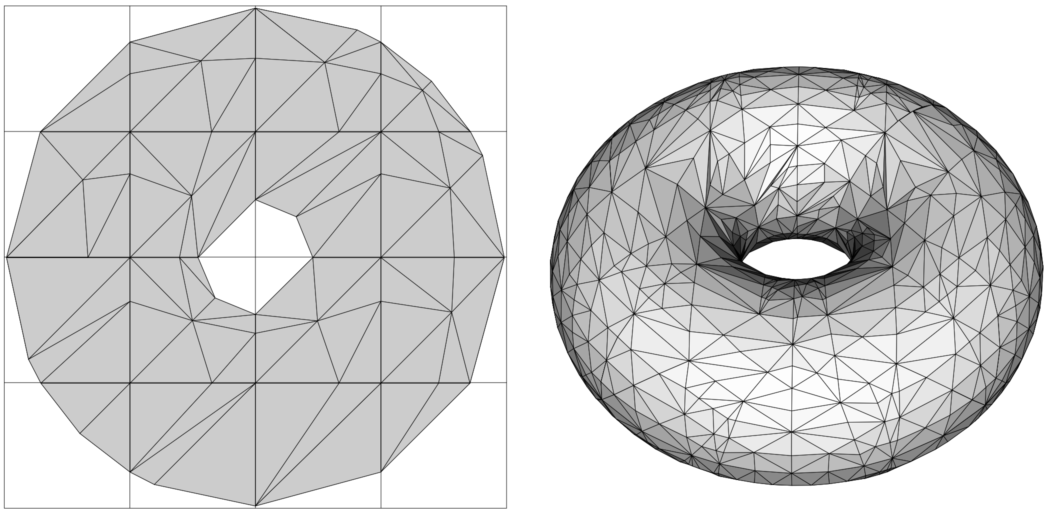
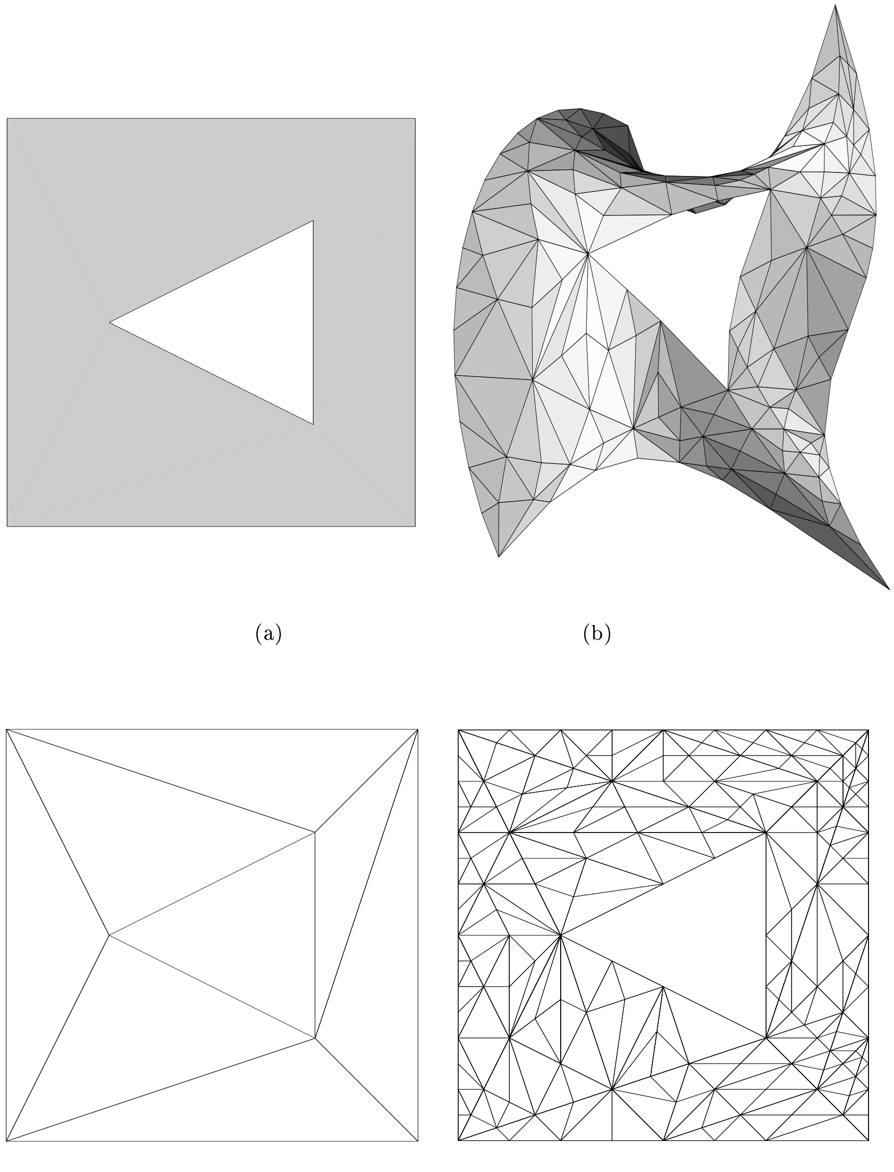


Fig. 13. Implicit torus: base mesh with 3D grid (left) and polygonization (right).



(a) (b)

(c) (d)

Fig. 14. Trimmed parametric Bezier patch: (a) trimming curve; (b) polygonization; (c) initial mesh; (d) final mesh.

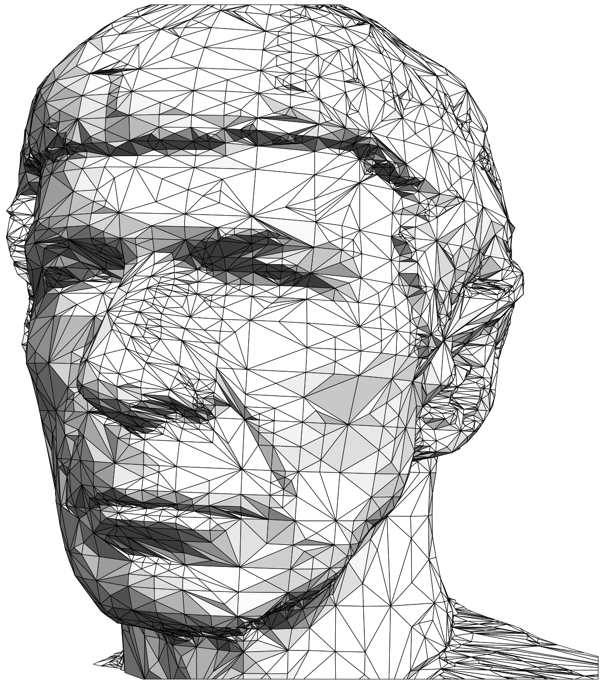


Fig. 15. Spock bust, a parametric surface from sampled data.

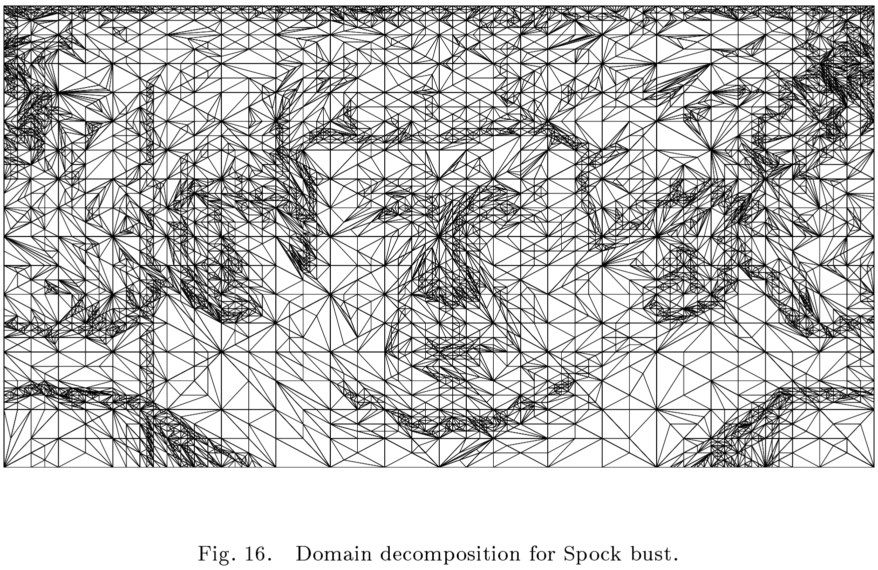


Fig. 16. Domain decomposition for Spock bust.

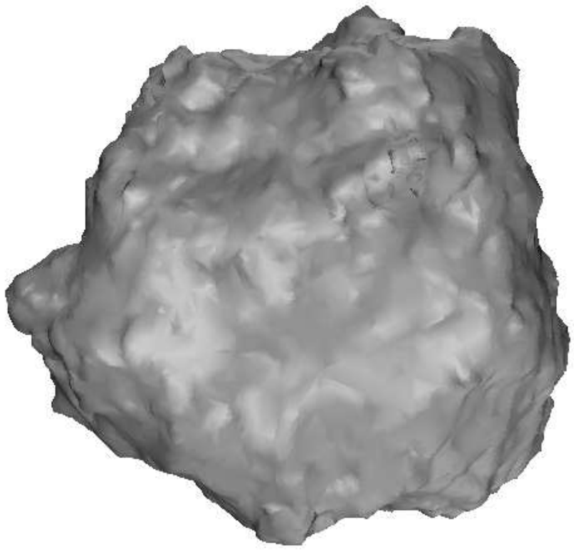


Fig. 17. Fractal rock, a high-frequency implicit surface.

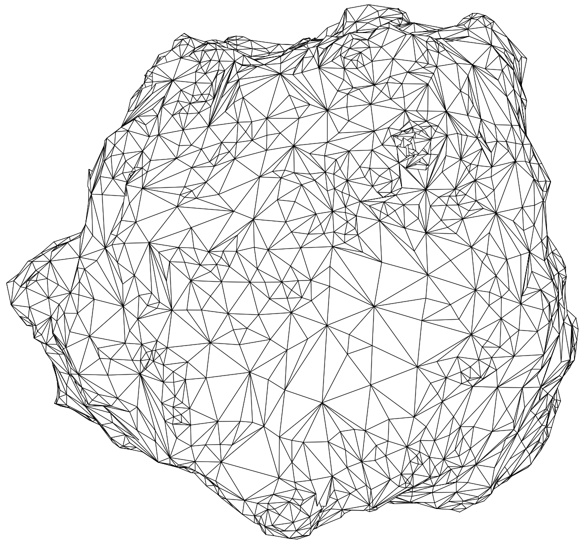


Fig. 18. The polygonal approximation for the fractal rock in Figure 17.

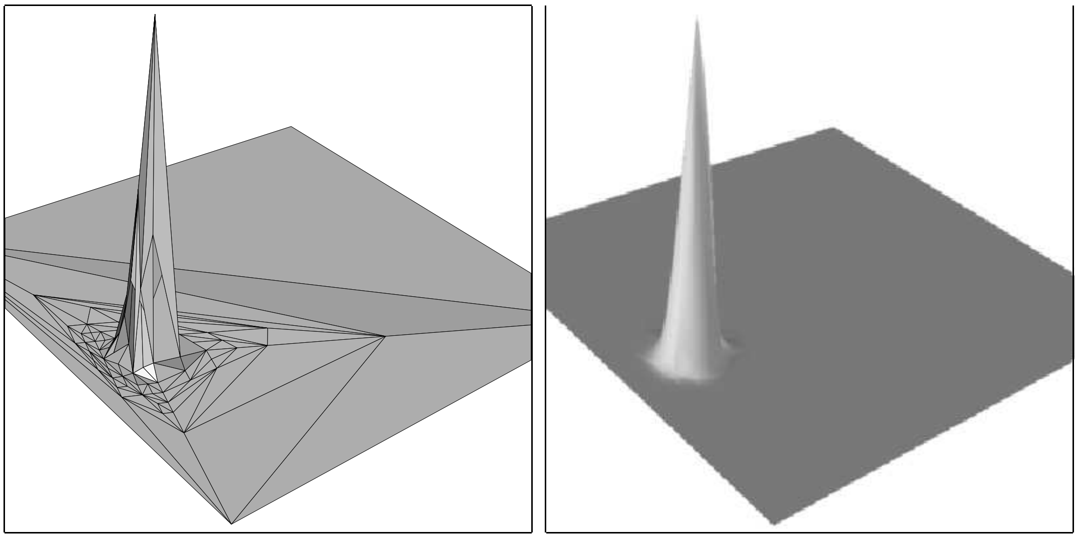


Fig. 19. Polygonization of a Gaussian spike inside a flat cell (left) and Gouraud-shaded rendering of the mesh (right).

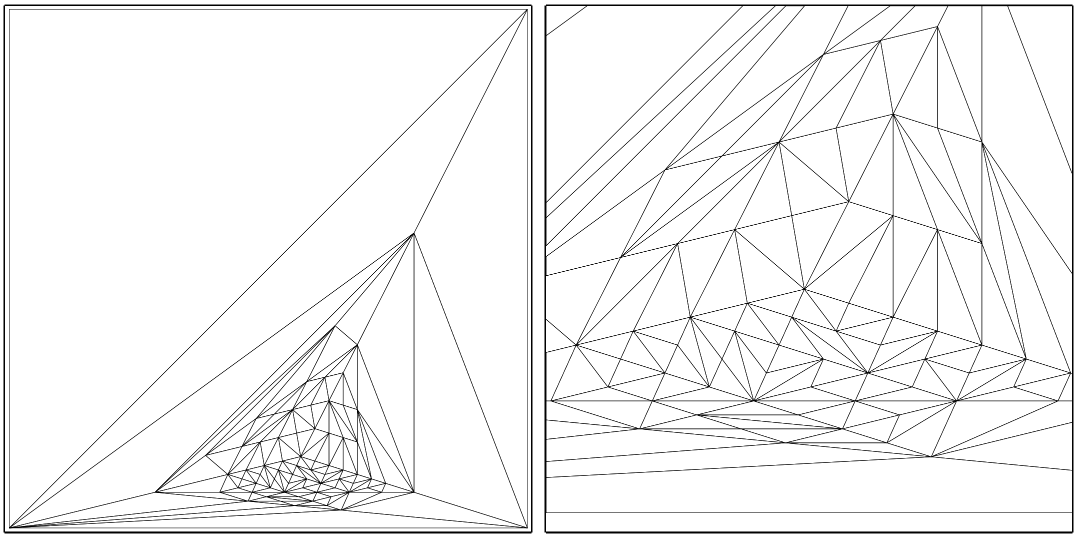
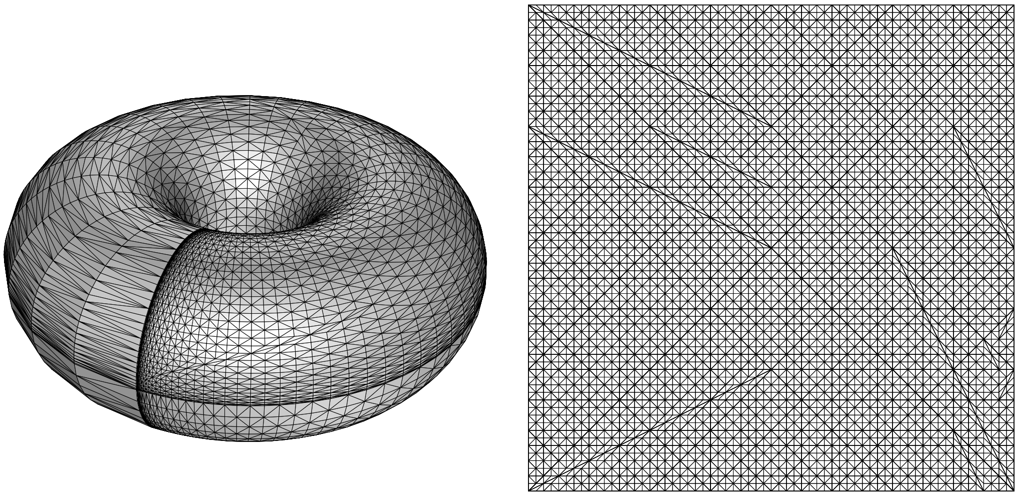
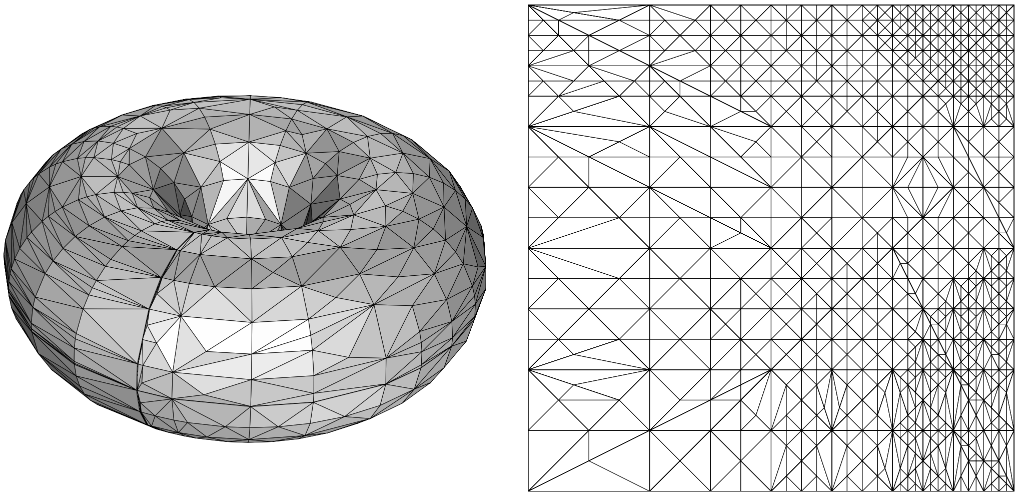


Fig. 20. Domain decomposition of the parametric function in Figure 19 (left) and detail of the high variation area (right).



(a) (b)

Fig. 21. Uniform polygonal approximation of torus over warped domain (left) and corresponding domain decomposition (right).



(a) (b)

Fig. 22. Adaptive polygonal approximation of the torus in Figure 21 (left) and corresponding domain decomposition (right).

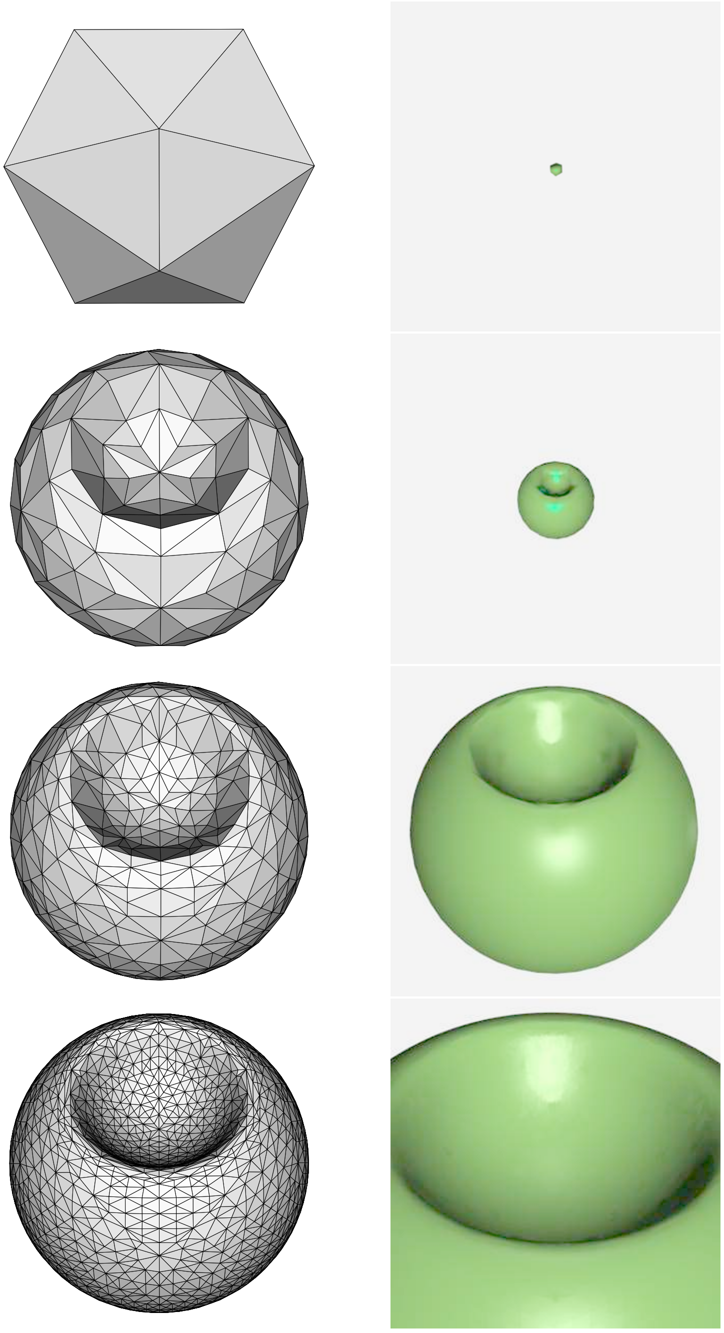


Fig. 23. Multiresolution sequence of polygonal approximations: base mesh and levels 3, 4 and 5.