Multiresolution Analysis of Arbitrary Meshes

Technical Report # 95-01-02

Matthias Eck* Tony DeRose* Tom Duchamp*
Hugues Hoppe† Michael Lounsbery‡ Werner Stuetzle*

Abstract

In computer graphics and geometric modeling, shapes are often represented by triangular meshes. With the advent of laser scanning systems, meshes of extreme complexity are rapidly becoming commonplace. Such meshes are notoriously expensive to store, transmit, render, and are awkward to edit. Multiresolution analysis offers a simple, unified, and theoretically sound approach to dealing with these problems. A multiresolution representation of a mesh consists of a simple base mesh together with a sequence of local corrections of decreasing magnitude. By omitting small correction terms we can efficiently construct approximations with fewer triangles for compact storage and fast transmission and rendering. When editing meshes it is often desirable to modify the gross shape without affecting surface details. We can accomplish this by modifying terms in the approximation with large spatial extent.

Lounsbery *et al.* have recently developed a technique for creating multiresolution representations for a restricted class of meshes with *subdivision connectivity*. Unfortunately, meshes encountered in practice typically do not meet this requirement. In this paper we present a method for overcoming the subdivision connectivity restriction, meaning that completely arbitrary meshes can now be converted to multiresolution form. The method is based on the approximation of an arbitrary initial mesh M by a mesh M^J that has subdivision connectivity and is guaranteed to be within a specified tolerance.

Multiresolution approximation of an arbitrary mesh thus proceeds in two steps: in the first step we approximate M by M^J, and in the second step we convert M^J to a multiresolution representation using the methods of Lounsbery *et al.*

*University of Washington, Seattle, WA
†Microsoft Research, Redmond, WA
‡Alias Research, Toronto, Ontario, Canada

This work was supported in part by Alias Research Inc., Microsoft Corp., and the National Science Foundation under grants CCR-8957323 and DMS-9103002.

The key ingredient of our algorithm is the construction of a parametrization of M over a simple domain. We expect this parametrization to be of use in other contexts, such as texture mapping or the approximation of complex meshes by NURBS patches.

1 Introduction

In computer graphics and geometric modeling, shapes are often represented by triangular meshes. With the advent of laser scanning systems, meshes of extreme complexity are rapidly becoming commonplace. The objects shown in Color Plates 2(f) and 3(f) for instance, consist of 69,451 and 103,713 triangles, respectively. Such meshes are notoriously expensive to store, transmit, and render. They are also awkward to edit, as many vertices typically must be moved to make a change of substantial spatial extent.

Multiresolution analysis offers a promising new approach for addressing these difficulties in a simple, unified, and theoretically sound way. A multiresolution representation of a mesh (Color Plate 1(e)), as recently developed by Lounsbery *et al.* [10], consists of a simple base mesh (Color Plate 1(d)) together with a sequence of local correction terms, called *wavelet coefficients*, capturing the detail present in the object at various resolutions. Color Plates 1(f)–(i) show a sequence of intermediate resolution models incorporating an increasing number of wavelets.

Multiresolution mesh representations are particularly convenient for a number of applications, including:

- *Compression/simplification*: A multiresolution mesh can be compressed by removing small wavelet coefficients. Moreover, the threshold for removal can be chosen such that the resulting approximation is guaranteed to be within a specified error tolerance of the original mesh. A number of examples are shown in the color plates.

- *Progressive display and transmission*: An attractive method for displaying a complex object is to begin with a low resolution version that can be quickly rendered, and then progressively improve the display as more detail is obtained from disk or over a network. Using a multiresolution representation, this is simply achieved by first displaying the base mesh, and then progressively adding the contributions of wavelet coefficients in order of decreasing magnitude.

- *Level-of-detail control*: High performance rendering systems often use a level-of-detail hierarchy, that is, a sequence of approximations at various levels-of-detail. The crudest approximations are used when the viewer is far from the object, while higher detail versions are substituted as the viewer approaches. Multiresolution representations naturally support this type of display by adding successively smaller wavelet coefficients as the viewer approaches the object, and by removing them as the viewer recedes. Moreover, the coefficients can be

added smoothly, thereby avoiding the visual discontinuities encountered when switching between approximations of different resolution. This use of multiresolution representations is illustrated in Color Plates 2 and 3 and on the accompanying video tape.

- *Multiresolution editing*: Editing at various scales can proceed along the lines developed by Finkelstein and Salesin [3] by ordering coefficients according to their support, that is, by the spatial extent of their influence. Color Plates 4(e) and (f) show edits of a mesh at low and high levels of detail.

Although the multiresolution analysis of Lounsbery *et al.* [10] can be applied to meshes of arbitrary topological type, it has a serious shortcoming: it is restricted to meshes with *subdivision connectivity*, that is, to meshes obtained from a simple base mesh by recursive 4-to-1 splitting (see Figure 1). Figure 6(a) shows an example of a mesh with subdivision connectivity — it results from recursively splitting the faces of an octahedron five times. Unfortunately, few of the meshes encountered in practice have this restricted structure.

In this paper we present a method for overcoming the subdivision connectivity restriction, meaning that completely arbitrary meshes can now be converted to multiresolution form. Our approach is to develop an algorithm for approximating an arbitrary mesh M (as in Color Plate 1(a)), which might not have subdivision connectivity, by a mesh M^J that does (as in Color Plate 1(e)), and is guaranteed to be within a prescribed tolerance ϵ_1 of M. We refer to this process as *remeshing*, and we call M^J the *remesh*.

Multiresolution analysis of an arbitrary mesh M thus proceeds in two steps: we first use remeshing to approximate M by a mesh M^J with subdivision connectivity, and then use the method of Lounsbery *et al.* to convert M^J to multiresolution representation. (Although we cannot reproduce here all the results of Lounsbery *et al.* [10], we have included a brief summary in Appendix A.)

The key ingredient of the remeshing procedure — and the principal technical contribution of the paper — is the construction of a homeomorphism between M and a mesh K^0 possessing a small number of faces. Stated another way, we construct a continuous parametrization for M over a domain mesh K^0. We then sample the parametrization to produce the remesh. Considerable care is taken to create a parametrization and a sampling pattern so that the resulting remesh can be well approximated with relatively few wavelet coefficients.

The construction of parametrizations for complex shapes over simple domains is a fundamental problem that occurs in numerous applications, including texture mapping, and the approximation of meshes by NURBS patches. We therefore expect that our parametrization algorithm will have uses outside of remeshing.

The remainder of the paper is organized as follows. In Section 2, we describe the relationship between our work and previously published methods. In Section 3, we give a high level overview of the major steps of the remeshing algorithm. The details of the algorithm are presented in Sections 4-7. In Section 8, we apply our method to meshes of varying complexity, and give examples of compression, level-of-detail control, and editing. We close with conclusions and future work in

Section 9.

2 Related Work

The difficulty of dealing with complicated shapes is evidenced by the extensive recent research on the topic.

The problems of compression/simplification and level-of-detail control have been addressed by Turk [18], Schroeder *et al.* [17], Hoppe *et al.* [7], Rossignac and Borrel [15], and Varsney [20]. Our approach differs from these methods in three principal respects. First, it provides guaranteed error bounds, whereas the approaches of Turk, Schroeder *et al.*, and Hoppe *et al.* do not. Second, it produces a single compact representation from which a continuous family of lower resolution approximations can be quickly and easily constructed, whereas the previous methods generate a discrete set of models of varying complexity. (We should note, however, that Turk, and Rossignac/Borrel, and Varsney present methods for interpolating between models.) Third, our representation can be simply and conveniently edited at multiple scales, whereas it is hard to imagine how one would achieve similar results using the previous approaches.

The editing of complex shapes was a central motivation for the introduction of hierarchical B-splines by Forsey and Bartels [5]. Forsey and Bartels [4] and Forsey and Wang [6] have subsequently developed methods for fitting hierarchical B-splines to meshes topologically equivalent to a disk. Finkelstein and Salesin [3] have demonstrated how wavelet representations of B-spline curves and tensor product surfaces can be used to achieve similar benefits. The main advantage of our method is its ability to deal with shapes of arbitrary topological type.

Finally, the problem of parametrizing meshes has recently been considered by Maillot *et al.* [11] in the context of texture mapping. However, the parametrizations they construct are not useful for our purpose: their surface tiles are not triangular, and their local parametrizations do not fit together continuously. Additionally, our local parametrizations, based on the well-established theory of harmonic maps, are simpler to compute than the ones used by Maillot *et al.*, and seem to produce parametrizations of comparable quality (see Section 4).

3 Overview of the Remeshing Algorithm

The basic idea of remeshing is to construct a parametrization of M over a suitably determined domain mesh K^0, and then to resample the parametrization to produce a mesh M^J with subdivision connectivity.

Our remeshing algorithm consists of three steps, as illustrated in Color Plate 1:

1. *Partitioning:* Partition M into a number of triangular regions $T_1, ..., T_r$, as shown in Color

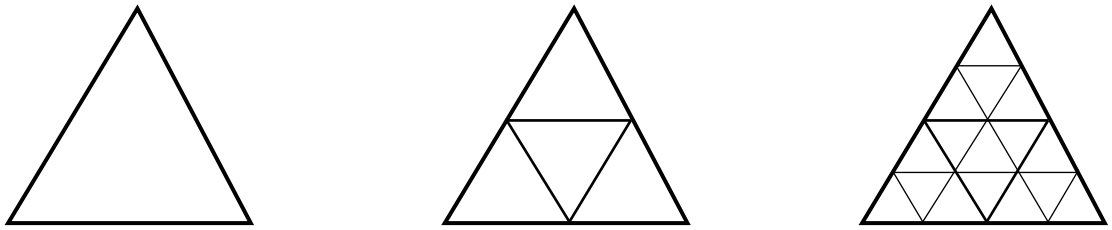


Figure 1: 4-to-1 splitting of a triangular face: (a) the initial face; (b) after one 4-to-1 split; (c) after two 4-to-1 splits.

Plate 1(c). We want the number r of regions to be small, because the lowest complexity approximation we can construct has r faces, as shown in Color Plate 1(d). Basic tools used in partitioning are harmonic maps, maps that preserve as much of the metric structure (lengths, angles, etc.) of M as possible. Harmonic maps are described in Section 4. A detailed description of our partitioning algorithm is given in Section 5.

Let m be the number of vertices or *nodes* of the triangulation $T_1, ..., T_r$. This triangulation determines the structure of a simple mesh K^0, called the *base complex*, with a face corresponding to each of the r triangular regions. This mesh serves as the domain of the parametrization constructed in the next step.

2. *Parametrization:* For each region T_i of M construct a (local) parametrization $\rho_i : F_i \rightarrow T_i$ over the corresponding face F_i of the base complex K^0. The local parametrizations are made to fit together continuously, meaning that collectively they define a globally continuous parametrization $\rho : K^0 \rightarrow M$. We want the coordinate functions of the parametrization to vary as little as possible since such functions have multiresolution approximations with few significant wavelet coefficients, leading to high compression ratios. Harmonic maps in a sense minimize distortion and therefore are particularly well suited for this purpose.

   A detailed description of the parametrization step is presented in Section 6.

3. *Resampling:* Perform J recursive 4-to-1 splits on each of the faces of K^0 (see Figure 1). This results in a triangulation K^J of K^0 with subdivision connectivity. The remesh M^J, as shown in Color Plate 1(e), is obtained by mapping the vertices of K^J into $\mathbf{R}^3$ using the parametrization ρ, and constructing an interpolating mesh in the obvious way; M^J therefore has vertices lying on M, and has subdivision connectivity.

   The resampling step is described more fully in Section 7, and it is shown that J can be determined so that M^J and M differ by no more than a specified remeshing tolerance ϵ_1.

4 Harmonic maps

A crucial building block of our remeshing algorithm is a method for constructing a parametrization of a (topological) disk $D \subset M$ over a convex polygonal region $P \subset \mathbf{R}^2$. This method is used in two places: in the construction of the triangulation $T_1, \ldots, T_r$ of M (see Section 5), and in the parametrization of M over the base complex K^0 (see Section 6). We want this parametrization to have small distortion; for example, if D is (close to) planar, we want the parametrization to be (close to) linear. Because the region may be geometrically complex (see, for example, Figure 2), some distortion is usually inevitable.

While it is not clear in general how to find a parametrization ρ with small distortion, there is a closely related and well-studied problem that has a unique solution: Fix a homeomorphism g between the boundary of D and the boundary of the polygonal region P; then there is a unique *harmonic map* $h : D \to P$ that agrees with g on the boundary of D and minimizes *metric dispersion* (see Eells and Sampson [2], pages 114–115, and the survey article by Eells and Lemaire[1]). Metric dispersion is a measure of the extent to which a map stretches regions of small diameter in D. It is thus a measure of metric distortion.

In addition to minimizing metric distortion, the harmonic map h has a number of important properties: (i) It is infinitely differentiable on each face of D; (ii) it is an embedding [16]; and (iii) it is independent of the triangulation of D. Because $h : D \to P$ is an embedding, the inverse h^{-1} is a parametrization of D over P. We will return below to the issues of choosing the boundary map g and of computing approximations to h.

The dispersion minimizing property of harmonic maps is illustrated in Figure 2, which shows a piecewise linear approximation of a harmonic map from a geometrically complex region onto a polygon. The relatively dense regions of the polygon correspond to the ears and nose of the cat. Notice that the aspect ratios of triangles tend to be preserved. Notice also that the map introduces a certain amount of area compression. This is inevitable because the region has a large area relative to its circumference, and consequently any embedding must introduce some distortion in edge lengths. The harmonic map tends to minimize such distortion while maintaining the embedding property and attempting to preserve aspect ratios of triangles.

Harmonic maps can be visualized as follows. Imagine D to be composed of elastic, triangular rubber sheets sewn together along their edges. Stretch the boundary of D over the boundary of the polygon P according to the map g. The harmonic map minimizes the total energy $E_{harm}[h]$ of this configuration of rubber sheets.

Rather than constructing the harmonic map directly, we compute a piecewise linear approximation. Assume that n vertices $v_1, \ldots, v_n$, called *corners*, have been selected on the boundary ∂D of D (see Figure 2), and (for technical reasons) assume that the degree of each of the remaining boundary vertices is at least 3.

We choose the polygon P by mapping the corners of D onto the vertices of an n-gon in $\mathbf{R}^2$ whose vertices all lie on a circle and whose sides subtend angles proportional to the arc lengths

of the boundary segments of D joining the corresponding corners. We then define g to be the piecewise linear map that maps the corners of ∂D to the vertices of P, and is a homothety (i.e. an isometry up to a constant factor) between each boundary segment of D and the corresponding side of P (Figure 2).

Now suppose that h is any piecewise linear map that agrees with g on the boundary. By explicitly integrating the functional E_{harm} over each face, one finds that E_{harm} can be reinterpreted as the energy of a configuration of springs with one spring placed along each edge of D:

$$E_{harm}[h] = 1/2 \sum_{\{i,j\} \in \text{Edges}(D)} \kappa_{i,j} \|h(i) - h(j)\|^2 , \tag{1}$$

where the spring constants $\kappa_{i,j}$ are computed as follows: For each edge $\{i, j\}$, let $\mathrm{L}_{i,j}$ denote its length as measured in the initial mesh D, and for each face $\{i, j, k\}$, let $\text{Area}_{i,j,k}$ denote its area, again as measured in D. Each interior edge $\{i, j\}$ is incident to two faces, say $\{i, j, k_1\}$ and $\{i, j, k_2\}$. Then

$$\kappa_{i,j} = \left(\mathrm{L}_{i,k_1}^2 + \mathrm{L}_{j,k_1}^2 - \mathrm{L}_{i,j}^2\right) / \text{Area}_{i,j,k_1} + \left(\mathrm{L}_{i,k_2}^2 + \mathrm{L}_{j,k_2}^2 - \mathrm{L}_{i,j}^2\right) / \text{Area}_{i,j,k_2}$$

The formula for spring constants associated to boundary edges has only one term.

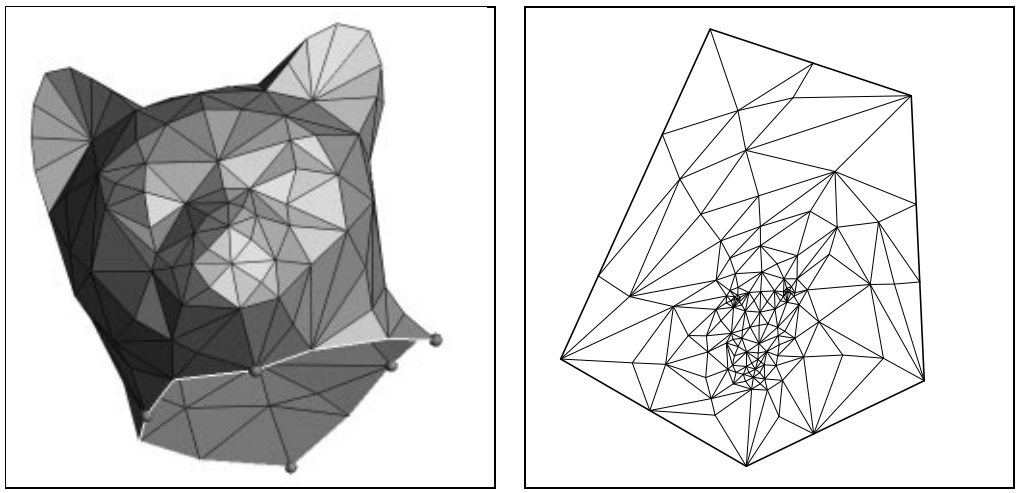
Although the spring constants $\kappa_{i,j}$ can assume negative values, the function (1) is positive definite, and its unique minimum can be found by solving a sparse linear least-squares problem. In contrast to the harmonic map itself, its piecewise linear approximation is not always an embedding. In our experience, this problem occurs extremely rarely (3 times in the roughly 1000 harmonic maps we computed). In these cases we use uniform spring constants.

For the remainder of this paper we refer to the unique piecewise linear function minimizing (1) as a harmonic map, although strictly speaking it is only an approximation.

Others have developed similar approaches to embedding disk-like regions. One such approach, described by Kent *et al.* [8], is also based on minimizing the energy of a network of springs. They choose spring constants to be either all equal or inversely proportional to edge lengths. Maillot *et al.* [11] introduced another functional, also based on elasticity theory.

Figure 3 illustrates the behavior of the various embedding schemes in a simple example where the region D (see Figure 3(a)) is a triangulation of a planar polygon P and $g : \partial D \to \partial P$ is the identity. The harmonic map (Figure 3(b)) is the identity map and therefore has no metric distortion. The method of Kent *et al.* with either choice of spring constants produces considerable metric distortion (Figure 3(c) and (d)).

The mathematical properties of the functional proposed by Maillot *et al.* are not entirely clear. In particular, the smooth theory to which it is an approximation does not yield planar embeddings of geometrically complex regions. This led them to introduce a user-specified tuning parameter α. In the example of Figure 3, the choice $\alpha = 1$ also produces the identity map, whereas the choice $\alpha = 1/2$ leads to small distortion (see Figure 3(e)). While the method of Maillot *et al.*



(a) Original mesh tile (b) Harmonic embedding

Figure 2: The harmonic map for the head of a cat. The neck of the cat is mapped onto the boundary of the polygon. The "corner" vertices (thoses sent to vertices of the polygon) are indicated by small balls.

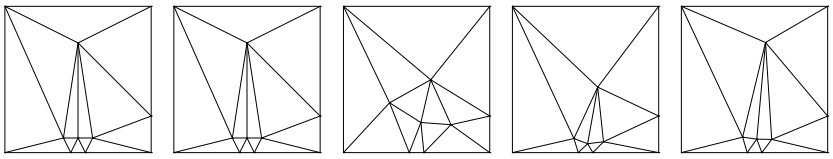


Figure 3: Comparison of various "spring embeddings". From left to right: (a) Original mesh; (b) Harmonic map and embedding of Maillot *et al.* with $\alpha = 1$; (c) $\kappa_{i,j} = 1$; (d) $\kappa_{i,j} = 1/\mathrm{L}_{i,j}$; (e) Embedding of Maillot *et al.* with $\alpha = 1/2$.

appears to have performance comparable to ours (for appropriately chosen α), it requires non-linear optimization, whereas our method requires only the solution of a sparse linear least-squares problem.

5 Partitioning

Our partitioning scheme is based on the concepts of Voronoi diagrams and Delaunay triangulations. Let us first see how these concepts could be used to partition a dense triangulation of a planar region into a small number of large triangles. We could begin by selecting a set of relatively uniformly distributed vertices of the dense triangulation, and then compute the Delaunay triangulation for the selected vertices. One method for computing the Delaunay triangulation for a set of sites in the plane is to first construct the Voronoi diagram. Its polyhedral dual is the Delaunay triangulation if

Voronoi tiles meet three at a corner.

By analogy, our approach is to first partition M into a set of Voronoi-like tiles τ_i using a discrete approximation of the Voronoi diagram as described in Section 5.1.

We then construct the dual to the Voronoi diagram, resulting in a Delaunay-like partition of M into triangular regions T_i, as described in Section 5.2.

5.1 Construction of the Voronoi diagram

As mentioned above, we use a discrete version of the Voronoi diagram to partition M into a set of Voronoi-like tiles. An efficient algorithm for constructing true Voronoi diagrams on the surface of a mesh has been developed by Mount [14], but it is rather difficult to implement, and is unnecessary for our purposes.

We first describe an algorithm for computing tiles $\tau_1, ..., \tau_s$ given a set of sites logically positioned at the centroids of the *site faces* $S = \{f_1, \ldots, f_s\}$. We then present an algorithm for selecting a set S of site faces for which the induced Voronoi diagram is dual to a triangulation.

5.1.1 Computing the Voronoi diagram for a given set of site faces

Our algorithm for partitioning the original mesh M into a set of Voronoi-like tiles uses a "simultaneous advancing fronts" technique. The idea is to simultaneously grow the tiles τ_i from their site faces f_i until the tiles cover the mesh.

Our measure of distance between faces is an approximation of geodesic distance over the surface. It is defined by constructing a dual graph to the mesh: the nodes of the graph correspond to faces of M, and the edges of the graph connect nodes corresponding to adjacent faces. We set the cost of edges in this dual graph to be the distance between centroids of the corresponding faces. The distance between two faces is defined as length of the shortest path in this dual graph.

The advancing fronts idea is implemented as an s-source Dijkstra algorithm, which finds for each face of M the shortest path to the closest site face. Like the ordinary single source Dijkstra algorithm, the s-source version can be implemented efficiently using a priority queue.

5.1.2 Selecting the site faces

In this section we describe an algorithm for selecting a set S of site faces such that the induced Voronoi diagram, computed as above, is dual to a triangulation. Although our algorithm for selecting such site faces can be applied to any mesh M, let us assume for the moment that M does not possess boundaries. With this assumption, the Voronoi diagram must satisfy the following conditions to be dual to a triangulation:

1. tiles must be homeomorphic to disks;

2. no pair of tiles may share more than one *cut* (a cut is a contiguous set of edges of M along which a pair of tiles touch);

3. no more than three tiles can meet at any vertex.

Our approach to selecting site faces is to incrementally add them to the set S until all conditions are satisfied. The algorithm begins by initializing S with a single randomly chosen face. Tiles associated with the faces in S are grown until either they cover M, in which case tile growth terminates, or until conditions (1) is violated. If this happens, a tile touched itself, in which case one of the two faces that led to the violation is added to S and tile growth is resumed. When tile growth is complete, conditions (2) and (3) are checked, and if violated, a new site is added at the offending location. The process continues until all conditions are satisfied. This may be impossible to achieve. If the algorithm terminates unsuccessfully, the dual of the Voronoi diagram has regions with more than 3 sides. In this case we can simply triangulate those regions. This has never happened in any of the examples we have run.

A nice property of the s-source Dijkstra algorithm is that it can be updated incrementally as new sites are added to S, making the above process typically much more efficient than $O(sn \log n)$, where n is the number of faces of M.

To accommodate boundaries, we introduce a single *fictitious* Voronoi tile, logically outside of M, that touches each of the boundaries of M. Conditions (1) through (3) can then be applied without change. To ensure that the Delaunay-like triangulation covers M, we require that boundary tiles (those adjacent to the fictitious tile) have sites on the boundary of M. This issue is addressed again in the next section. To achieve this requirement, the algorithm adds a new boundary site face whenever an interior tile touches a boundary. As before, when tile growth stops, conditions (2) and (3) are checked, and if violated, appropriate new sites are added.

It sometimes happens that tiles have adjacent short cuts, a situation that leads to Delaunay-like triangles with poor aspect ratios, and hence to poor compression rates. We therefore add to the list of conditions one that disallows such tiles. Adjacent cuts of a tile are deemed short if the sum of their lengths is less than 10% of the length of the boundary of the tile. When an offending pair of cuts is found, one of the faces they share is added as a new site.

The results of applying this algorithm to various meshes are shown in Color Plates 1(b), 2(a), and 3(a).

5.2 Construction of the Delaunay-like triangulation

The partition of M into Voronoi tiles obtained in the previous section has the property that its dual graph consists of 3-sided faces. However, mapping these 3-sided faces onto the surface is a non-trivial problem. The obvious approach of connecting pairs of Voronoi sites by the shortest

paths on the surface — as is done in constructing the Delaunay triangulation in the plane — is not guaranteed to produce a valid triangulation for arbitrary manifolds since the resulting paths can cross. Moreover, finding the shortest paths between two points on a mesh is itself a difficult problem [13].

Rather than connecting points using shortest paths, we instead use an approximation based on harmonic maps. The minimum metric distortion property of harmonic maps generally results in relatively short (and hence relatively straight) polygonal curves on M that are guaranteed not to cross. These *paths* are drawn in yellow in Color Plates 1(c), 2(b), and 3(b). We now present the details of the construction.

The first step is to compute the harmonic map h_i that carries each Voronoi tile τ_i into an appropriate planar polygon P_i, as described in Section 4. The inverse of h_i provides a parametrization of τ_i over P_i which we use to construct the paths lying on M.

Let τ_i and τ_j denote two adjacent interior Voronoi tiles as illustrated in Figure 4. The path of the Delaunay triangulation joining these tiles is constructed as follows: the cut shared by the tiles is mapped to an edge $e_{i,j}$ of P_i by the harmonic map h_i; similarly, the cut is mapped to an edge $e_{j,i}$ of P_j by h_j (see Figure 4). We construct a line $L_{i,j}$ from the centroid of P_i to the midpoint of $e_{i,j}$, and a line $L_{j,i}$ from the centroid of P_j to the midpoint of $e_{j,i}$. The path is formed by mapping these lines onto M using the inverse harmonic maps. That is, the path is obtained by joining $h_i^{-1}(L_{i,j})$ and $h_j^{-1}(L_{j,i})$.[1]

The construction of a path between an interior tile τ_i and a boundary tile τ_k is slightly different, as indicated in Figure 4. In order for the Delaunay triangulation to cover M, it is necessary to construct paths from the boundary. (The site selection algorithm of Section 5.1.2 was designed with this goal in mind in that it guarantees that boundary tiles have site faces on the boundary.) We therefore select a boundary vertex v_k of the site face f_k, and construct $L_{k,i}$ as the line from $h_k(v_k)$ to the midpoint of $e_{k,i}$. The line $L_{i,k}$ is constructed as before from the centroid of P_i to the midpoint of $e_{i,k}$.

Finally, two adjacent boundary tiles τ_k and τ_ℓ are connected by the path along the boundary between v_k and v_ℓ.

The edges of the paths thus constructed are generally not edges of M. For convenience in constructing the parametrizations of Section 6, we refine M to include the path edges.

6 Parametrization

The result of the partitioning step is a triangulation $T_1, \ldots, T_r$ of M. As in Section 3, let m be the number of nodes in this triangulation. Identifying each of the nodes with one of the canonical basis

[1] Note that this path does not connect the site faces as one might expect. We have found that the method described here produces more uniform triangulations than were obtained by connecting site faces.

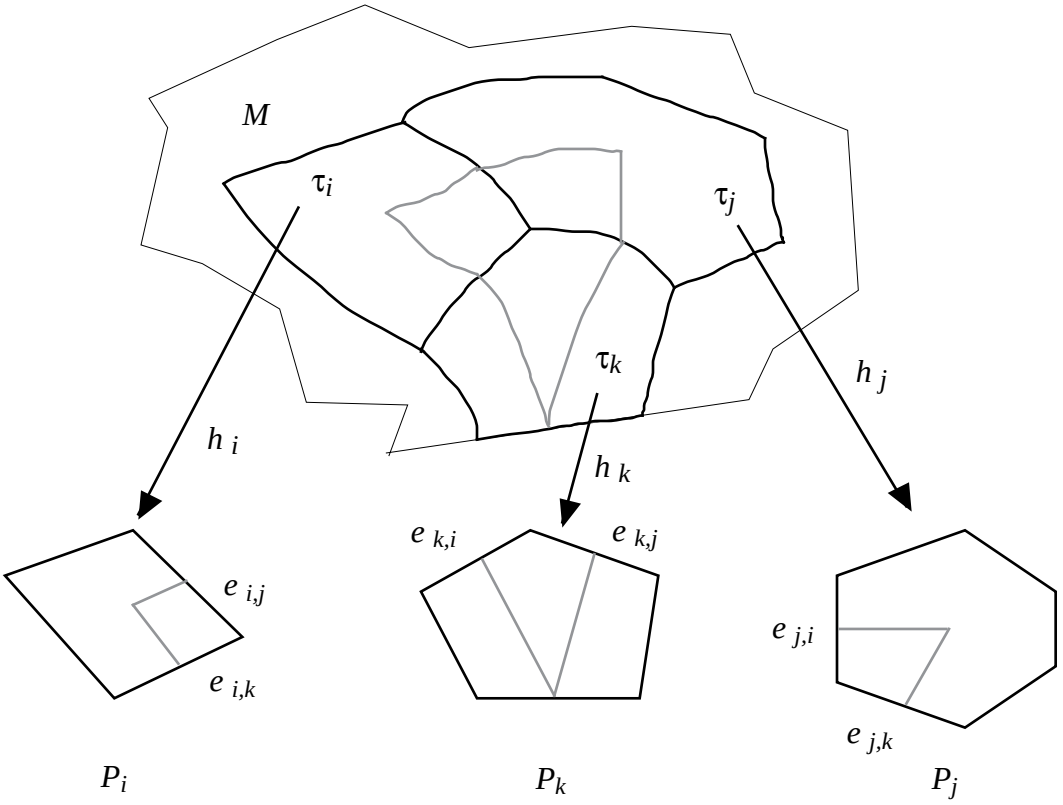


Figure 4: Construction of Delaunay paths on M.

.

vectors of $\mathbf{R}^m$ defines the base complex K^0, a triangular mesh that logically resides in $\mathbf{R}^m$.

The goal of this section is to construct a continuous parametrization $\rho : K^0 \rightarrow M$ of the initial mesh over the base complex. We map each triangle T_i onto a triangular region of the plane using the harmonic maps described in Section 4. We then affinely map the triangular region onto the corresponding face F_i of the base complex. The composition of the two maps is an embedding, and therefore its inverse ρ_i defines a parametrization of T_i over F_i. By construction, the maps ρ_i agree on shared boundaries, and thus the ρ_i collectively define a continuous parametrization ρ of M over K^0.

7 Resampling

In this section, we describe a method for producing a mesh M^J with subdivision connectivity from the parametrization $\rho : K^0 \rightarrow M$ constructed in Section 6. We also show how to determine the subdivision level J so that M^J and M differ by no more than a specified remeshing tolerance ϵ_1.

For a given value of J, we first produce a triangulation K^J of K^0 by performing J recursive 4-to-1 splits of the faces of K^0. We then approximate ρ by a function ρ^J defined as the piecewise linear interpolant to ρ on K^J; that is, ρ^J is such that $\rho^J(\mathbf{x}_i^J) = \rho(\mathbf{x}_i^J)$, where the points $\mathbf{x}_i^J$ (called *knots*) denote the vertices of K^J.

The simplest strategy for performing a 4-to-1 split of a face is to position the split points at midpoints of edges, as illustrated in Figure 1. We refer to this process as *parametrically uniform resampling* since the faces of K^J are of equal size. Alternatively, we could attempt to place the knots so that the images of triangles of K^J, that is, the triangles of the remesh M^J, are of equal size. We refer to this as *geometrically uniform resampling*.

As one of our fundamental objectives is high compression rate, we evaluate the performance of a resampling strategy by the number of wavelet coefficients needed for a given compression tolerance ϵ_2. This number is governed by at least two competing factors:

1. As mentioned in Section 3, the coordinate functions of ρ should be as slowly varying as possible; this is largely achieved by the distortion minimizing property of the harmonic map parametrizations.

2. The triangles of M^J should be of roughly uniform size. Lounsbery *et al.* define wavelets so that the magnitude of a wavelet coefficient is a measure of the "unweighted" least-squares error that would be incurred if the coefficient was set to zero. By unweighted we mean that deviations on large triangles of M^J are counted no more heavily than deviations on small triangles. If M^J has triangles of roughly uniform size, magnitudes of wavelet coefficients are better measures of geometric error.

The strategy that has performed best in our experiments is a hybrid strategy using geometrically uniform sampling in the first few splitting steps (the first three steps in all our examples), and

ϵ_2	Geom. Uniform	Hybrid	Param. Uniform
0.63%	(2313) [4659]	(1621) [3339]	(2668) [5367]
1.25%	(970) [1940]	(776) [1549]	(1078) [2133]
2.5%	(399) [801]	(370) [733]	(373) [750]
6.25%	(115) [253]	(111) [229]	(108) [231]

Table 1: Performance of the three sampling strategies on the cat model. Parentheses denote the number of significant wavelet coefficients; square brackets denote the number of triangles. All examples were run using $\epsilon_1 = 1.25\%$. Errors are measured as a percentage of the object's size.

parametrically uniform sampling in subsequent steps. Intuitively, this strategy does a reasonable job of uniformly distributing the triangles on a coarse scale, while still remaining faithful to the harmonic parametrization on smaller scales.

This intuition is supported by numerical results. Our tests have shown that hybrid resampling typically results in wavelet expansions with fewer significant coefficients than either parametrically uniform or geometrically uniform resampling. Moreover, the number of subdivisions J necessary to satisfy a remeshing tolerance ϵ_1 is often smaller and hence the remesh is often faster to compute and requires less storage. Table 1 presents the results of an experiment for the cat mesh (shown in Color Plate 4(a)) for various wavelet compression tolerances ϵ_2. Notice that hybrid resampling is particular advantageous for small tolerances.

7.1 Geometrically uniform resampling

The task of determining new knots $\mathbf{x}_i^j \in K^0$ so that the triangles generated are roughly uniform in size is an optimization problem whose solution we approximate using the following recursive greedy algorithm.

In the parametrically uniform resampling process the knot $\mathbf{x}_i^j$ at level j is simply computed as midpoint of the edge of the two (neighboring) knots $\mathbf{x}_{n_1(i)}^{j-1}$ and $\mathbf{x}_{n_2(i)}^{j-1}$ at level $j-1$. Instead of performing uniform subdivision, we define

$$\mathbf{x}_i^j = (1-\lambda_i^j)\cdot \mathbf{x}_{n_1(i)}^{j-1} + \lambda_i^j \cdot \mathbf{x}_{n_2(i)}^{j-1} \quad \text{with} \quad \lambda_i^j \in (0,1),$$

where the splitting parameter λ_i^j is determined as follows: Split the two faces adjacent to the edge from $\mathbf{x}_{n_1(i)}^{j-1}$ to $\mathbf{x}_{n_2(i)}^{j-1}$, as illustrated in Figure 5. Our goal is to find λ_i^j so that the regions $R_i^j = \triangle_{i,1}^j \cup \triangle_{i,3}^j$ and $S_i^j = \triangle_{i,2}^j \cup \triangle_{i,4}^j$ map to regions of equal area on M.

In the current implementation we have simplified the area computations by using a discrete approximation: We scatter a roughly uniform collection of points on the faces of M, then map these sample points back to K^0 using ρ^{-1} (ρ^{-1} is the harmonic map, so it is already known). We

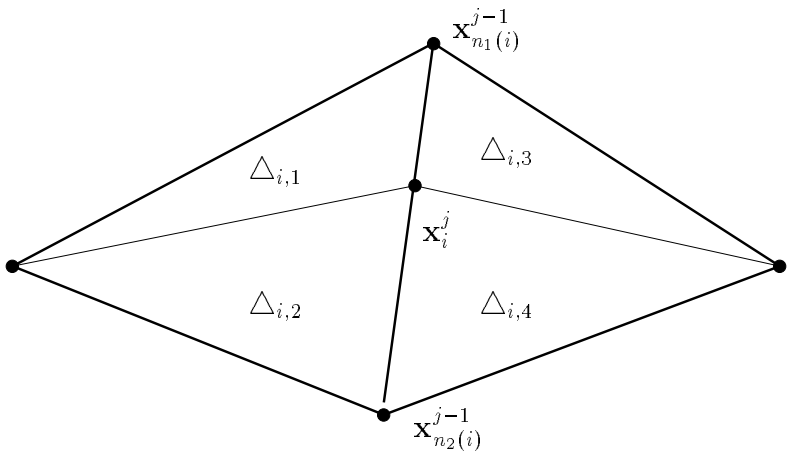


Figure 5: Computing the new knot $\mathbf{x}_i^j$

then use binary search to compute the parameter λ_i^j so that the number of sampled points in the regions R_i^j and S_i^j are nearly equal.

7.2 Bounding the remeshing error

In this section we describe how to determine J such that the remesh M^J and the initial mesh M deviate by no more than a remeshing tolerance ϵ_1 in an L^∞ sense. That is, we seek to find the smallest J such that

$$\max_{\mathbf{x}\in K^0} \|\rho(\mathbf{x}) - \rho^J(\mathbf{x})\| \leq \epsilon_1.$$

Our strategy for determining J will be to perform successive steps of 4-to-1 splitting until the error bound is satisfied.

To bound the error for a given value of J, let $E(\mathbf{x}) := \rho(\mathbf{x}) - \rho^J(\mathbf{x})$ denote the (vector-valued) error function. First, note that the preimages of the triangles of M under ρ form a partition π of K^0, and that ρ is a linear function on each triangle of π. Next, recall that ρ^J is linear within each of the triangles of the partition K^J of K^0. Thus, $E(\mathbf{x})$, the difference between the two, is linear within each cell of the union partition $\pi^J = \pi \cup K^J$. The squared norm of $E(\mathbf{x})$ is therefore quadratic and convex up over each cell of π^J, and so must achieve is maximum value at a vertex of π^J.

The L^∞ error for a given value of J can therefore easily be determined by evaluating $E(\mathbf{x})$ at the vertices of π^J. Using a local marching technique such as the one in Kent *et al.* [8], these vertices can be found in time proportional to the total number of vertices in π and K^J.

8 Results

Color Plates 1-4 illustrate the steps of the algorithm and present examples of its applications.

Color Plate	object	# faces of M	# Voronoi tiles τ_i	# Delaunay triangles T_i	remeshing tol. ϵ_1	subdiv. level J	time mins
1	holes3	11,776	31	70	0.75 %	4	6.3
2	bunny	69,451	84	151	1.50 %	5	22.5
3	dino	103,713	117	229	2.25 %	5	31.5
4	cat	698	7	9	1.25 %	6	0.5
4	hypersheet	3,817	39	65	1.25 %	4	2.2
4	phone	165,978	58	111	3.00 %	5	136.6

Table 2: Summary of results of the remeshing algorithm

Color Plate 1 demonstrates the complete process of multiresolution analysis for a mesh of genus 3. We first partition the mesh into Voronoi-like tiles shown in Color Plate 1(b). We then construct the corresponding Delaunay-like triangulation (Color Plate 1(c)). The Delaunay triangles define a simple base complex that serves as the domain for the parametrization of the mesh. Resampling this parametrization using the hybrid strategy described in Section 7 with a remeshing tolerance of $\epsilon_1 = 0.75\ \%$ required $J = 4$ subdivision steps and produced the remesh shown in Color Plate 1(e) consisting of 17,920 triangles. The lowest resolution approximation, shown in Color Plate 1(d), is a piecewise linear embedding of the base complex. Color Plates 1(f)-1(i) show approximations with increasing levels of detail, using, respectively, 183, 409, 803, and 1,558 wavelet coefficients.

Table 2 summarizes the remeshing process for a variety of other meshes. (All computing times were measured on a SGI Onyx Reality Engine 2 with 256MB of memory.) Note that the number of Voronoi tiles is influenced more by the geometry of the model than by the number of faces of M.

Approximating the dinosaur and the phone with low tolerances would require high subdivision levels. This is due to the presence of jagged boundaries which can only be well approximated using a large number of subdivisions.

Computing times are strongly dependent on the ratio of the number of faces of M to the number of Delaunay triangles, since the bottleneck of the algorithm, the harmonic map computation, requires solving sparse least-squares problems whose time complexity is proportional to the square of the number of vertices in the triangles.

Table 3 summarizes the results of wavelet compression applied to remeshed models. Each line of the table gives the number of faces of the remesh, the compression tolerance ϵ_2 used in the wavelet compression method described in Appendix A.2, the number of wavelet coefficients, and the number of faces of the resulting approximation. The total deviation between the compressed model and the original is bounded by $\epsilon = \epsilon_1 + \epsilon_2$, the sum of the remeshing tolerance and the compression tolerance. Note that for storage and transmission purposes, the relevant performance measure is the number of wavelet coefficients rather than the number of faces, since only the wavelet coefficients (and their indices) have to be stored or transmitted. As an indication of the

Color Plate	object	# faces of M^J	compression tol. ϵ_2	# wavelet coefficients	# faces
1(f)	holes3	17,920	4.0 %	183	376
1(g)	holes3	17,920	2.0 %	409	854
1(h)	holes3	17,920	1.0 %	803	1,618
1(i)	holes3	17,920	0.5 %	1,558	3,140
2(g)	bunny	154,624	2.5 %	1,155	2,280
2(h)	bunny	154,624	0.5 %	5,345	10,793
2(i)	bunny	154,624	0.1 %	25,477	50,979
3(g)	dino	234,496	4.5 %	944	1,919
3(h)	dino	234,496	1.0 %	5,298	10,833
3(i)	dino	234,496	0.1 %	43,945	88,344
4(c)	cat	36,864	12.5 %	41	78
4(d)	cat	36,864	4.0 %	225	431
4(i)	hypersheet	16,640	2.5 %	329	633
4(l)	phone	113,664	0.3 %	4,527	9,441

Table 3: Summary of results of the algorithm of Lounsbery *et al.*

time needed to execute the analysis and synthesis algorithms of Lounsbery *et al.*, converting the remesh of the bunny to multiresolution form (filterbank analysis) required 1.6 mins. Construction of the mesh shown in Color Plate 2(k) by adding wavelet coefficients to the base mesh required 3.0 mins. Smooth changes in resolution can be accomplished significantly faster by incrementally adding or subtracting wavelet coefficients.

Color Plates 2 and 3 illustrate level-of-detail control. The original models (Color Plates 2(f) and 3(f)) were created from laser range data using the mesh zippering algorithm of Turk and Levoy [19].[2] Note the enormous reduction in the number of triangles when the multiresolution approximations are viewed from afar.

Color Plates 4(e)-(f) show an example of *multiresolution editing*. Color Plate 4(e) illustrates a large-scale modification caused by changing a wavelet coefficient at the coarsest level, whereas Color Plate 4(f) corresponds to changing two coefficients at an intermediate level-of-detail.

Color Plates 4(g)-(l) show the application of remeshing and multiresolution analysis to two additional meshes.

[2] We are not responsible for chopping off the head of the dinosaur — that's how we received the models.

9 Conclusion

We have described an algorithm for solving the remeshing problem, that is, the problem of approximating an arbitrary mesh by a mesh with subdivision connectivity. Combined with the previous work of Lounsbery *et al.*, our remeshing algorithm allows multiresolution analysis to be applied to arbitrary meshes. Multiresolution representations support efficient storage, rendering, transmission, and editing of complex meshes in a simple, unified, and theoretically sound way.

We have applied our remeshing algorithm and multiresolution analysis to complicated meshes consisting of more than 100,000 triangles. Examples of compression, level-of-detail rendering, and editing are shown in the Color Plates.

The key ingredient of our remeshing procedure — and the principal technical contribution of the paper — is the construction of a continuous parametrization of an arbitrary mesh over a simple domain mesh. Parametrizing complex shapes over simple domains is a fundamental problem in numerous applications, including texture mapping and the approximation of meshes by NURBS patches. We therefore expect that our parametrization algorithm will have uses outside of multiresolution analysis. We intend to explore these uses in future work.

References

[1] J. Eells and L. Lemaire. Another report on harmonic maps. *Bull. London Math. Soc.*, 20:385–524, 1988.

[2] J. Eells and J.H. Sampson. Harmonic mappings of Riemannian manifolds. *Amer. J. Math.*, 86:109–160, 1964.

[3] Adam Finkelstein and David Salesin. Multiresolution curves. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(3):261–268, July 1994.

[4] D. Forsey and R. Bartels. Hierarchical B-spline fitting. *ACM Transactions on Graphics*. To appear.

[5] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics*, 22(4):205–212, 1988.

[6] David Forsey and Lifeng Wang. Multi-resolution surface approximation for animation. In *Proceedings of Graphics Interface*, 1993.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 19–26, August 1993.

[8] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):47–54, July 1992.

[9] J. Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Department of Computer Science and Engineering, University of Washington, September 1994. Available as ftp://cs.washington.edu/pub/graphics/LounsPhd.ps.Z.

[10] Michael Lounsbery, Tony DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. Submitted for publication. Preliminary version available as Technical Report 93-10-05b, Department of Computer Science and Engineering, University of Washington, January, 1994. Also available as ftp://cs.washington.edu/pub/graphics/TR931005b.ps.Z.

[11] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(3):27–34, August 1993.

[12] Stephane Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[13] J.S. Mitchell, D.M. Mount, and C.H. Papdimitrou. The discrete geodesic problem. *SIAM Journal of Computing*, 16(4):647–668, 1987.

[14] David M. Mount. Voronoi diagrams on the surface of a polyhedron. Department of Computer Science CAR-TR-121, CS-TR-1496, University of Maryland, May 1985.

[15] J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering. In B. Falcidieno and T.L. Kunii, editors, *Modeling in Computer Graphics*, pages 455–465. Springer-Verlag, June-July 1993.

[16] Richard Schoen and Shing-Tung Yau. Univalent harmonic maps between surfaces. *Inventiones math.*, 44:265–278, 1978.

[17] William Schroeder, Jonathan Zarge, and William Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):65–70, July 1992.

[18] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):55–64, July 1992.

[19] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(3):311–318, July 1994.

[20] Amitabh Varshney. *Hierarchical Geometric Approximations*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1994.

A Multiresolution Representation of Meshes with Subdivision Connectivity

As mentioned in Section 1, the main idea of multiresolution analysis is to decompose a function into a low resolution part and a set of correction or "detail" terms at increasing resolutions. Multiresolution analysis was first formalized by Mallat [12] for functions defined on $\mathbf{R}^n$. Lounsbery [9] and Lounsbery *et al.* [10] have recently extended the notion of multiresolution analysis to functions defined on base complexes of arbitrary topological type. Their results can be used to construct multiresolution representations of meshes with subdivision connectivity. The purpose of this appendix is to summarize their basic results and algorithms at a high level.

A.1 Background

The two basic ingredients of multiresolution analysis are a sequence of nested linear function spaces and an inner product. Lounsbery *et al.* use a sequence of spaces $V^0 \subset V^1 \subset \cdots$ associated with the base complex. To describe meshes, the approximation spaces V^j consist of piecewise linear functions; specifically, V^j is the space of continuous piecewise linear functions over a partition K^j of K^0 created by performing j recursive steps of 4-to-1 splitting to the faces of K^0, as shown in

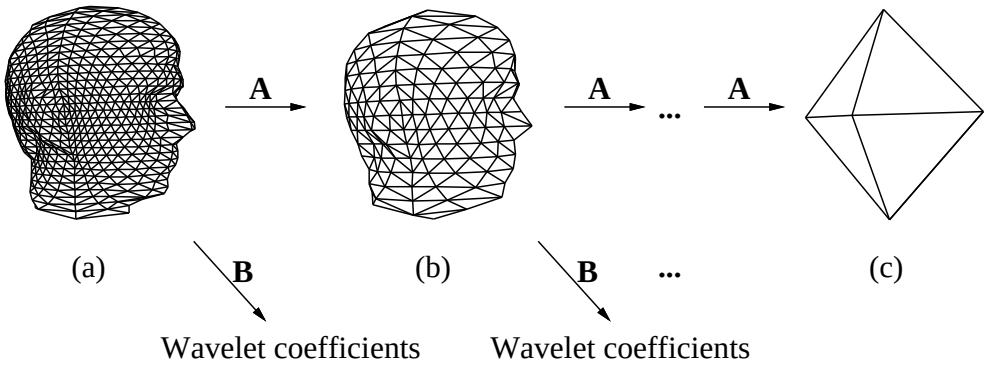


Figure 6: Decomposition of a mesh.

Figure 1. As j increases, the triangulation K^j becomes more dense, and so the functions in V^j are better able to model arbitrary continuous functions on K^0. The inner product used by Lounsbery *et al.* is the standard inner product defined as

$$\langle f, g \rangle := \int_{\mathbf{x} \in K^0} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}$$

where $d\mathbf{x}$ is the differential area of K^0 embedded in $\mathbf{R}^m$, so that all faces have unit area.

The inner product is used to define the following orthogonal complement spaces, also called *wavelet spaces*,

$$W^j := \{ f \in V^{j+1} \mid \langle f, g \rangle = 0 \quad \forall g \in V^j \}.$$

Intuitively, W^j captures the detail that is missed when a function in V^{j+1} is approximated by a function in V^j.

Basis functions for V^j are called *scaling functions.* In the piecewise linear case, particularly simple scaling functions for V^j are the "hat functions" on K^j: the i-th hat function $\phi_i^j \in V^j$ is the unique function in V^j that is one at $\mathbf{x}_i^j$ and zero at all other knots of K^j.

A *wavelet* $\psi_i^k(\mathbf{x})$ is a basis function for one of the wavelet spaces W^k. Lounsbery *et al.* [10] give constructions for wavelet bases on arbitrary base complexes K^0. A *wavelet basis* for V^j consists of a basis for V^0 together with bases for the wavelet spaces $W^0, ..., W^{j-1}$.

The parametrization $\rho^J \in V^J$ for V^j can be expanded in the hat function basis as

$$\rho^J(\mathbf{x}) = \sum_i v_i^J \phi_i^J, \qquad \mathbf{x} \in K^0 \tag{2}$$

where v_i^J denote the vertex positions of M^J. A *multiresolution representation* of ρ^J refers to its expansion in a wavelet basis

$$\rho^J(\mathbf{x}) = \sum_i v_i^0 \phi_i^0(\mathbf{x}) + \sum_{j=0}^{J-1} \sum_i w_i^j \psi_i^j(\mathbf{x}), \qquad \mathbf{x} \in K^0, \tag{3}$$

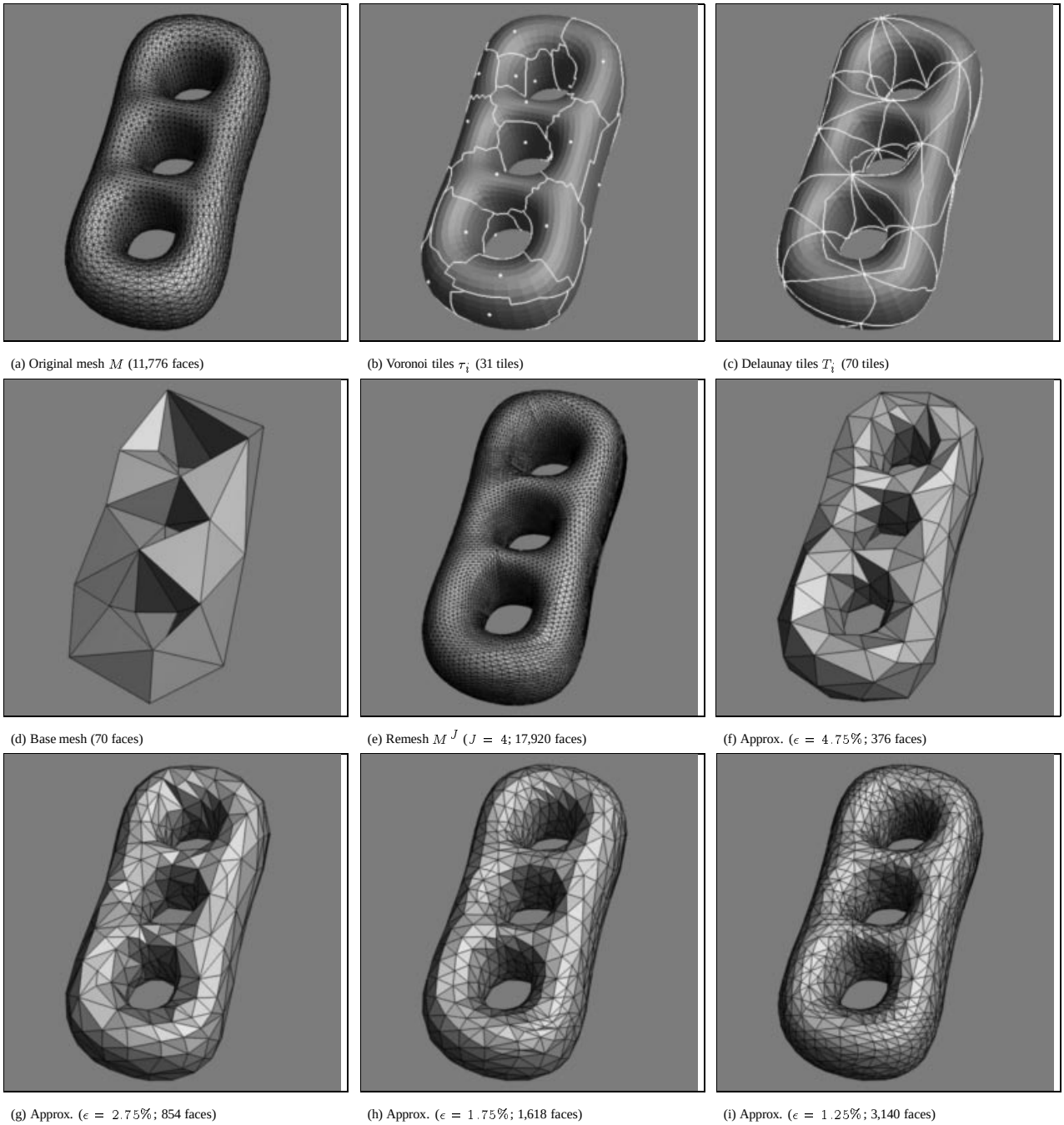
where w_i^j denote the wavelet coefficients.

An algorithm known as *filterbank analysis* can be used to convert between the hat function expansion and the multiresolution representation. The geometric interpretation of filterbank analysis is shown in Figure 6. The full detail model, described by $\rho^J(\mathbf{x})$ is successively decomposed into a lower resolution approximation together with a collection of coefficients that multiply the wavelets. The result is a simple base mesh together with wavelet coefficients at various levels of detail. The operators $\mathbf{A}$ and $\mathbf{B}$ in Figure 6 refer to sparse matrices whose entries are given by Lounsbery *et al.*. The filterbank analysis has an inverse process called filterbank synthesis that recovers the full resolution model from its multiresolution representation.

A.2 L^∞ Wavelet compression

The L^∞ error caused by wavelet compression can be bounded as follows. Setting a wavelet coefficient w_i^j to zero removes the term $w_i^j \psi_i^j(\mathbf{x})$ from the wavelet expansion. The L^∞ error incurred on a triangle Δ of K^J is exactly $\|w_i^j\| \max_{\mathbf{x}\in\Delta} \psi_i^j(\mathbf{x})$. The L^∞ error incurred on a triangle Δ by removing two coefficients is bounded by the sum of the individual errors.

This suggests the following simple algorithm proposed by Lounsbery [9] for achieving compression to within a specified L^∞ tolerance ϵ_2. (A related but different algorithm for L^∞ compression of curves represented by B-spline wavelets was presented by Finkelstein and Salesin [3]). Associate with each triangle of K^J a bound on the accumulated error incurred thus far. Consider removing each wavelet coefficients by visiting them in order of increasing magnitude. A coefficient is removed if the error incurred by removing it plus the already accumulated error in its support is less than ϵ_2. If the coefficient is removed, the accumulated error is replaced by the sum.



(a) Original mesh M (11,776 faces)

(b) Voronoi tiles τ_i (31 tiles)

(c) Delaunay tiles T_i (70 tiles)

(d) Base mesh (70 faces)

(e) Remesh M^J ($J = 4$; 17,920 faces)

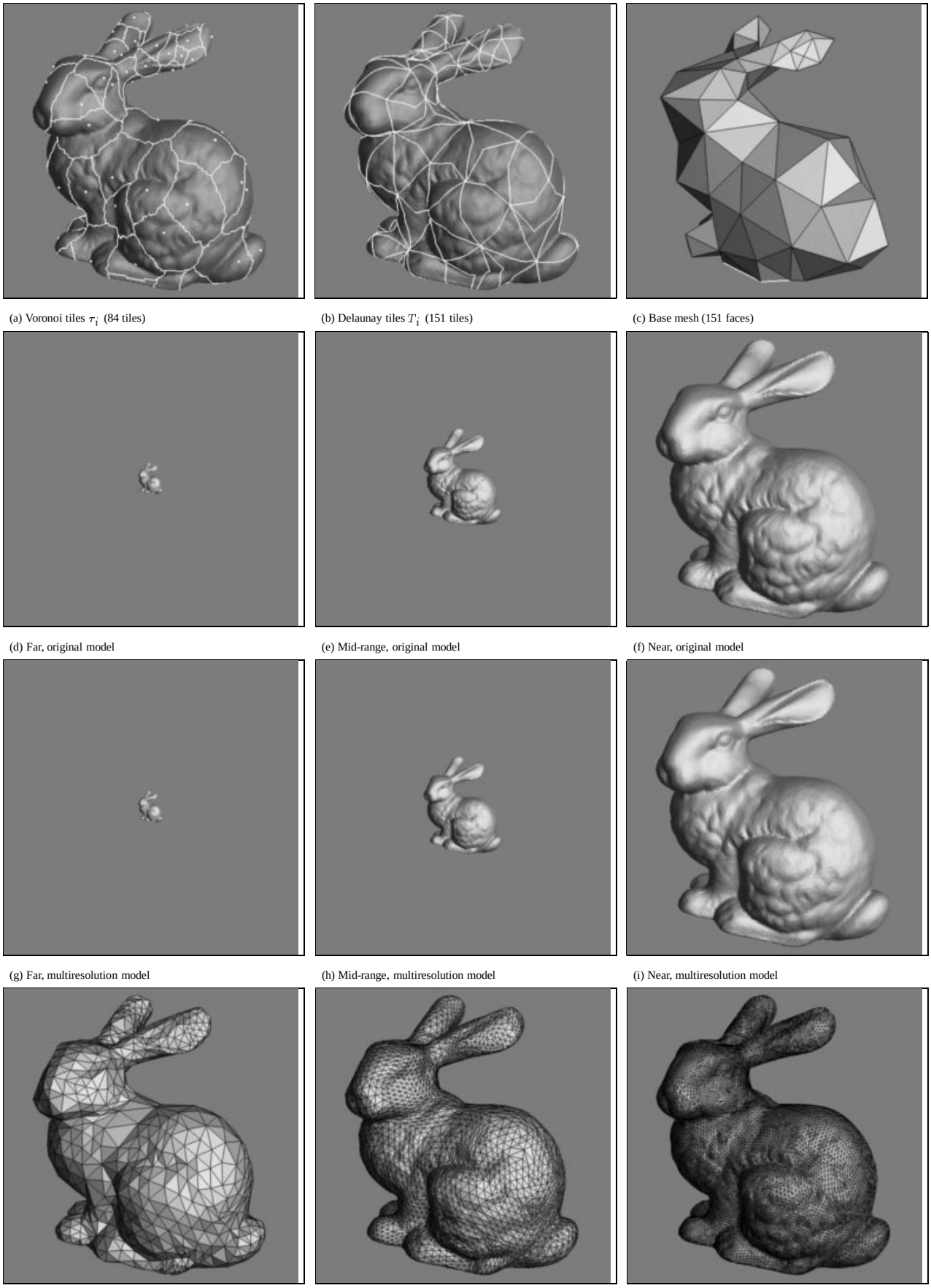
(f) Approx. ($\epsilon = 4.75\%$; 376 faces)

(g) Approx. ($\epsilon = 2.75\%$; 854 faces)

(h) Approx. ($\epsilon = 1.75\%$; 1,618 faces)

(i) Approx. ($\epsilon = 1.25\%$; 3,140 faces)

Color Plate 1: Example of partition, parameterization, resampling, and approximation of a mesh using multiresolution analysis. ($\epsilon = \epsilon_1 + \epsilon_2$ refers to maximum error tolerance, as a percentage of object diameter.)



(a) Voronoi tiles τ_i (84 tiles)

(b) Delaunay tiles T_i (151 tiles)

(c) Base mesh (151 faces)

(d) Far, original model

(e) Mid-range, original model

(f) Near, original model

(g) Far, multiresolution model

(h) Mid-range, multiresolution model

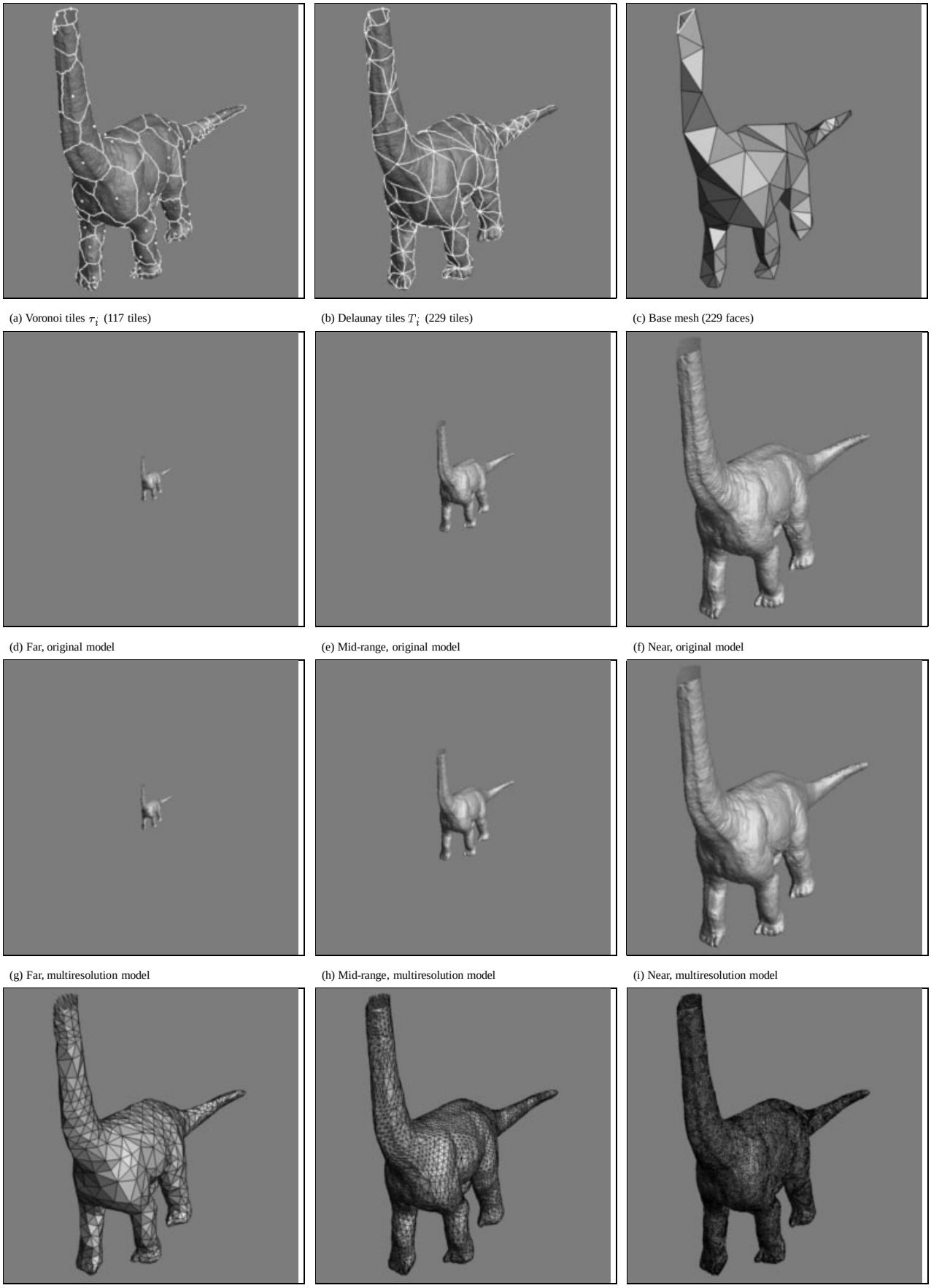
(i) Near, multiresolution model

(j) Mesh in (g) ($\epsilon = 4.0\%$; 2,280 faces)

(k) Mesh in (h) ($\epsilon = 2.0\%$; 10,793 faces)

(l) Mesh in (i) ($\epsilon = 1.6\%$; 50,979 faces)

Color Plate 2: Level-of-detail approximations of a dense bunny mesh (69,451 faces) using multiresolution analysis.



(a) Voronoi tiles τ_i (117 tiles)

(b) Delaunay tiles T_i (229 tiles)

(c) Base mesh (229 faces)

(d) Far, original model

(e) Mid-range, original model

(f) Near, original model

(g) Far, multiresolution model

(h) Mid-range, multiresolution model

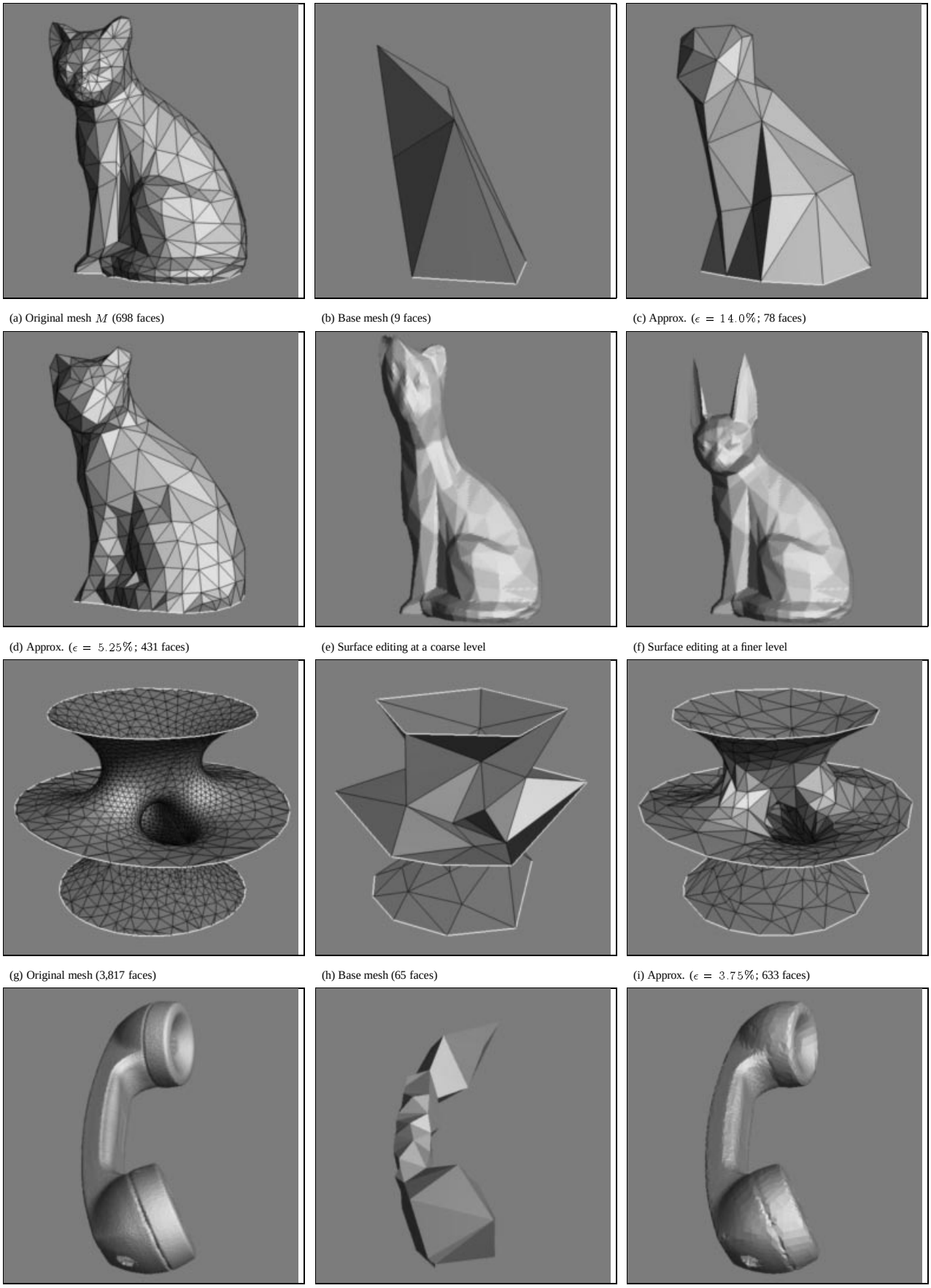
(i) Near, multiresolution model

(j) Mesh in (g) ($\epsilon = 6.75\%$; 1,919 faces)

(k) Mesh in (h) ($\epsilon = 3.25\%$; 10,833 faces)

(l) Mesh in (i) ($\epsilon = 2.35\%$; 83,344 faces)

Color Plate 3: Level-of-detail approximations of a dense dinosaur mesh (103,713 faces) using multiresolution analysis.



(a) Original mesh M (698 faces)

(b) Base mesh (9 faces)

(c) Approx. ($\epsilon = 14.0\%$; 78 faces)

(d) Approx. ($\epsilon = 5.25\%$; 431 faces)

(e) Surface editing at a coarse level

(f) Surface editing at a finer level

(g) Original mesh (3,817 faces)

(h) Base mesh (65 faces)

(i) Approx. ($\epsilon = 3.75\%$; 633 faces)

(j) Original mesh (165,978 faces)

(k) Base mesh (111 faces)

(l) Approx. ($\epsilon = 3.3\%$; 9,441 faces)

Color Plate 4: Example of multiresolution surface editing, and more results of multiresolution surface approximation.