Multiresolution Surfaces for Compression, Display, and Editing

Tony D. DeRose

Pixar Animation Studios

SIGGRAPH '96 Course Notes
Course Number 13
Wavelets in Computer Graphics

1 Introduction

Surfaces play a central role in many three-dimensional computer graphics applications. Objects with flat faces are naturally represented by polyhedral meshes, while objects with curved surfaces are typically represented by tensor-product or triangular spline patches. We would like to be able to construct hierarchical representations of all of these types of objects, in order to provide the opportunity for compression, multiresolution editing, and many of the other operations that we've seen applied to images and curves.

Tensor-product surface patches are one type of surface representation that is quite easily converted into a multiresolution form. In 1988, Forsey and Bartels described a hierarchical framework for tensor-product surface constructions called *hierarchical B-splines*. Their framework creates an *over-representation* of the geometry—in other words, there may be more than one way of representing a given tensor-product surface as a hierarchical B-spline.

Alternatively, a wavelet representation for tensor-product B-spline surfaces can be constructed by applying either the standard or nonstandard tensor-product construction to one-dimensional B-spline wavelets. Such a basis provides a single, unique representation for every tensor-product surface, and it requires the same amount of storage as the original surface control points. The wavelet basis allows us to perform on surfaces many of the same operations as described in elsewhere in these course notes for images and curves.

Unfortunately, tensor-product constructions are limited in the kinds of shapes they can model seamlessly. In particular, tensor products can only be used for functions parameterized on $\mathbb{R}^2$. They are not applicable to functions defined on more general topological domains, such as spheres (see Figure 1(a)) or surfaces of genus larger than one (Figure 1(b)).

In this chapter we provide a brief description for how multiresolution analysis can be extended to arbitrary topological surfaces, and we survey a number of the applications such a representation affords. More detail on these topics can be found elsewhere [1, 2, 5, 6]. To simplify the discussion, and to make it more concrete, we restrict these notes to polyhedral (that is, piecewise linear) surfaces. As described in Stollnitz *et al.* [6], the results hold more generally for surfaces generated through recursive subdivision.

We begin in Section 2 with an overview of multiresolution analysis for polyhedra. Multiresolution analysis is fundamentally a toolkit for analyzing functions, so in Section 3 we develop the necessary mathematical framework for treating polyhedra as functions. Section 3.1 also summarizes the constrution of a particularly useful family of biorthogonal surface wavelets called the k-disk wavelets. Applications of k-disk wavelets to compression, display, and editing are then presented in Sections 5, 6, and 7, respectively.

2 Overview of multiresolution analysis for surfaces

Although the mathematics of multiresolution analysis for surfaces is somewhat involved, the resulting algorithms are relatively simple. Before diving into the details, we give here a brief overview of how the method can be applied to decompose the polyhedral object shown in Figure 2(a).

Just as for images and curves, the idea behind multiresolution analysis for surfaces is

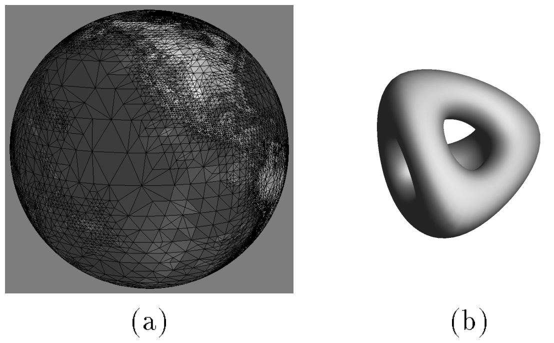


Figure 1: Non-tensor product surfaces.

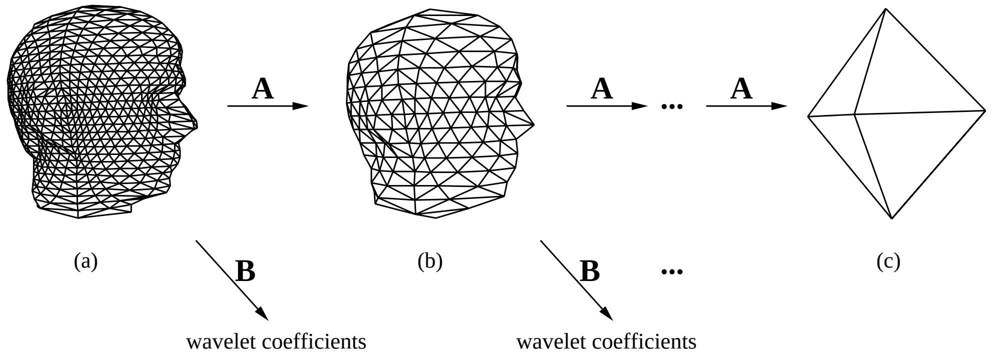


Figure 2: Decomposition of a polyhedral surface.

to split a high-resolution surface (in this case a polyhedral version of a bust of Spock) into a low-resolution part and a detail part. For example, the low-resolution part of the polyhedron in Figure 2(a) is shown in Figure 2(b). The vertex positions in (b) are computed as weighted averages of the vertex positions in (a). Again, just as for curves, the extration of the low resolution part is linear, so it can be expressed as multiplication by a matrix $\mathbf{A}^j$. The wavelet coefficients representing the detail can similarly be computed by multiplying by a matrix $\mathbf{B}^j$. This process is recursively applied to the low-resolution part, until the coarsest representation of the surface is obtained in Figure 2(c).

The so-called "analysis filters" $\mathbf{A}^j$ and $\mathbf{B}^j$ can be inverted to produce "synthesis filters" $\mathbf{P}^j$ and $\mathbf{Q}^j$. Synthesis, that is, the recovery of the original model from the lowest resolution approximation and with wavelet coefficients, can be viewed more concretely as involving two steps: splitting each triangular face of the low-resolution polyhedron into four subtriangles by introducing new vertices at edge midpoints; and perturbing the resulting collection of vertices according to the wavelet coefficients.

The challenge in creating a multiresolution analysis for surfaces is in designing the four analysis and synthesis filters so that:

1. the low-resolution versions are good approximations to the original object;
2. the magnitude of a wavelet coefficient provides a useful measure of the error introduced when that coefficient is set to zero; and
3. the analysis and synthesis processes have time complexities that grow linearly with the number of vertices.

3 The mathematical framework

In the remainder of these notes, we'll assume that the reader is familiar with the basic mathematical notions of multiresolution analysis, including the importance of a sequence of nested function spaces $V^0 \subset V^1 \subset \cdots$, an inner product $\langle \cdot \mid \cdot \rangle$, and the definitions of scaling functions, wavelets, birothogonality, and so on. The purpose of this section is to frame the analysis of polyhedral surfaces in the language of multiresolution analysis.

We'll use a sequence of increasingly faced polyhedra to help us define the sequence of increasingly large spaces V^j. In particular, we'll start with an arbitrary triangulated mesh M^0, that we call the *bash mesh.* Figure 3 shows the simplest possible base mesh, a tetrahedron. We create a mesh M^1 by subdividing each face into four subfaces by introducing edge midpoints. The faces of M^1 can be further subdivided to produce M^2, and so on. This recursive subdivision process is illustrated in Figure 3 for the tetrahedron, but keep in mind that the base mesh is arbitrary — it can have any topological type, and any number of vertices.

Notice that these meshes are nested when considered as surfaces in three space, in that all points of M^j are also points of M^{j+1}. We use these nested meshes to define a sequence of nested spaces as follows: With each mesh M^j, we define V^j as the set of all continuous functions that are linear on each face of M^j. These spaces are nested since any function

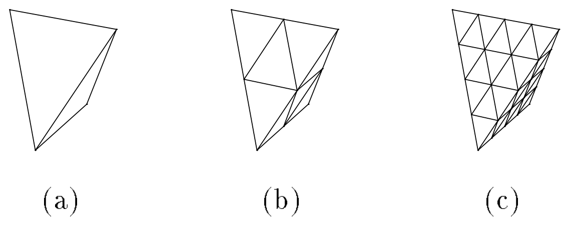


Figure 3: Recursive subdivision of a tetrahedron: (a) M^0, (b) M^1, (c) M^2.

that's linear on the faces of M^j is also linear on the faces of M^{j+1}. Formally, each function in V^j maps points of M^0 into a real number:

$$V^j := \{f | f : M^0 \to \mathbb{R} \text{ and } f \text{ linear on faces of } M^j\}$$

Next we need scaling functions $\{\phi_i^j(\mathbf{x})\}_i$ that span V^j. First note that since V^j contains only piecewise linear functions, a member of V^j is uniquely determined by its values at the vertices of M^j. A natural basis therefore consists of the so-called "hat" functions: associated with every vertex i of M^j is the unique member $\phi_i^j(\mathbf{x})$ in V^j that achieves value 1 at vertex i, and is zero at all other vertices of M^j. Wavelets $\psi_i^j(\mathbf{x})$ are simply basis functions for complement spaces $W^j = V^{j+1} - V^j$. A convenient family of wavelets is constructed in the next section.

Armed with these definitions, we now turn to the analysis of polyhedral surfaces. Observe that the hat functions can be used to construct polyhedral surfaces in 3-space. Specifically, a function such as

$$S(\mathbf{x}) = \sum_{i \in v(M^J)} c_i^J \phi_i^J(\mathbf{x}), \qquad \mathbf{x} \in M^0 \tag{1}$$

defines a polyhedron with vertex positions $c_i^J = (x_i^J, y_i^J, z_i^J)$ that is topologically equivalent to the bash mesh M^0. The set $v(M^J)$ indexes the vertices of M^J.

Notice that the vertices of S have the connectivity of M^J; that is, the connectivity of M^0 recursively subdivided J times. A mesh of this type is said to have *subdivision connectivity*. An example is shown in Figure 2 where the full resolution surface has the connectivity of an octahedron recursively subdivided 5 times.

Expressing $S(\mathbf{x})$ in wavelet form means writing it as

$$S(\mathbf{x}) = \sum_{i \in v(M^0)} c_i^0 \phi_i^0(\mathbf{x}) + \sum_{j=0}^{J-1} \sum_{i \in v(M^{j+1}) - v(M^j)} d_i^j \psi_i^j(\mathbf{x}) \tag{2}$$

for an appropriate choice of wavelets $\psi_i^j(\mathbf{x})$. The construction of such wavelets is summarized in the next section.

3.1 k-disc wavelets

In this section we construct a biorthogonal wavelet basis for polyhedral surfaces with the following properties:

- Analysis and synthesis can both be accomplished in linear-time.
- The wavelets are "nearly orthogonal" to the scaling functions, in the sense that their inner products with scaling functions are close to zero. A practical implication is that low-resolution surface approximations are close to least-squares best.

We start with lazy wavelets, which for polyhedral surfaces consist of the scaling functions (i.e, the hat functions) in V^{j+1} centered on edge midpoints of M^j. Although lazy wavelets have the advantage of being very simple, they are far from orthogonal to members of V^j, meaning that the coarse versions of a full-resolution surface are far from least-squares best. Fortunately, lifting can be used to make them "more orthogonal", resulting in what we call k-disk wavelets [5].

More precisely, consider a vertex i of M^{j+1} located at the midpoint of an edge e of M^j. The k-disk wavelet centered at vertex i is a function of the form

$$\psi_i^j = \phi_i^{j+1} - \sum_{v \in N_k} s_{iv}^j \phi_v^j \,, \tag{3}$$

where N_k denotes a set of vertices of M^j in a neighborhood of vertex i. The neighborhoods N_k are defined recursively: The neighborhood N_0 for the 0-disk wavelet consists of the endpoints of e; N_k contains the vertices of all triangles incident on N_{k-1} (see Figure 2).

The coefficients s_{iv}^j are chosen to minimize the norm of the orthogonal projection of ψ_i^j onto V^j. They are determined by solving the following system of linear equations [5]:

$$\sum_{v \in N_k} \langle \phi_u^j, \phi_v^j \rangle \, s_{iv}^j = \langle \phi_u^j, \phi_i^{j+1} \rangle, \quad \text{for all } u \in N_k.$$

Note that the system is local to vertex i. The size of the system for 0-disk wavelets is only 2×2. For larger values of k the size of the system depends on the valence of the parent vertices; in regular regions of the mesh where all vertices have valence 6, the system has size 10×10 for $k = 1$, and size 24×24 for $k = 2$.

3.2 k-disk analysis

An advantage of the k-disk wavelets is the ease with which a polyhedral surface $S(\mathbf{x})$ as in Equation 1 can be converted to multiresolution form as given in Equation 2. As is usual in wavelet analysis, this is accomplished with a filterbank procedure that successively splits a mesh $S^{j+1}(\mathbf{x})$ described by vertex positions c_i^{j+1} into a low resolution part $S^j(\mathbf{x})$, with vertex positions c_i^j and a detail part described by wavelet coefficients d_i^j. However, the decomposition is particularly simple with k-disk wavelets.

Let u and v be vertices of M^j, and let i be their midpoint. The wavelet ψ_i^j is therefore centered at i. The key observation is that the wavelet coefficient d_i^j can be computed

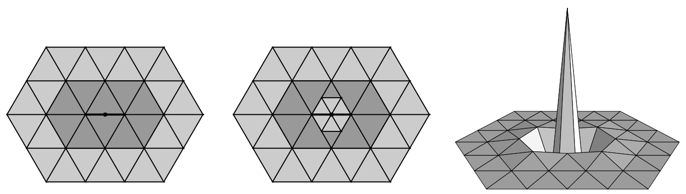


Figure 4: (a) The darkly shaded triangles form the set N_0 of the midpoint of i the center edge; the lightly shaded triangles are the additional needed to form N_1, the support of the 1-disk wavelet ψ_i^j centered at i. (b) The faces on which ψ_i^j is linear. (c) The graph of ψ_i^j .

according to

$$d_i^j = \frac{1}{\psi_i^j(i)} \left(c_i^{j+1} - \frac{c_u^j + c_v^j}{2} \right). \tag{4}$$

Once all wavelet coefficients at level j have been computed, $S^j(\mathbf{x})$ can determined by subtracting out their collective contribution, since

$$S^j(\mathbf{x}) = S^{j+1}(\mathbf{x}) - \sum_i d_i^j \psi_i^j(\mathbf{x}).$$

The following pseudocode summarizes the k-disk filterbank analysis process:

```
for $j$ decreasing to 0 do
    for all $i \in v(M^j)$ do $c_i^j = c_i^{j+1}$
    for all $i \in v(M^{j+1}) - v(M^j)$ do
        $d_i^j = \frac{1}{\psi_i^j(i)}(c_i^{j+1} - \frac{c_u^j + c_v^j}{2})$
        for all $i' \in v(M^j)$ within the support of $\psi_i^j$ do
            $c_{i'}^j - = d_i^j \psi_i^j(i')$
        end for
    end for
end for
```

4 Conversion to multiresolution form

In Section 3 it was shown how a polyhedron $S(\mathbf{x})$ parameterized on a simple base mesh M^0 could be decomposed onto the basis of k-disk wavelets if $S(\mathbf{x})$ possessed subdivision connectivity. The principal difficulty for practical application is that, in most situations, neither the simple mesh M^0 nor the parameterization of the surface is known. For instance, the bunny shown in Figure 5(a) is initially defined only as a collection of approximately 70,000 triangles stitched together into a complicated mesh.

The first step of converting an arbitrary surface into multiresolution form is therefore to determine a simple base mesh M^0 that is topologically equivalent to the given surface, and a parametric function $S(\mathbf{x})$ that maps points $\mathbf{x} \in M^0$ into 3-space. An algorithm for solving this problem was given last year by Eck *et al.* [2]. Their algorithm produces the base mesh M^0 for the bunny shown in Figure 5(b).

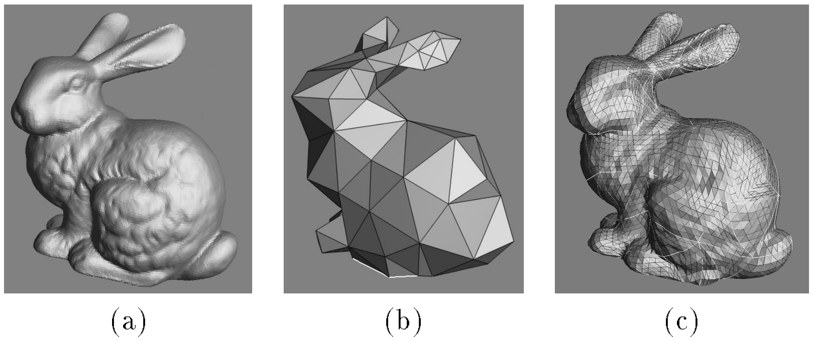


Figure 5: (a) A complex mesh consisting of approximately 70,000 triangles, created using an optical scanner and the zippering technique of Turk and Levoy [7]; (b) the base mesh M^0 constructed by the algorithm of Eck *et al.* [2]; (c) the projection of the model into V^5.

Eck *et al.* show not only how to construct an appropriate base mesh and parametrize $S(\mathbf{x})$ over it, but also how remesh it with subdivision connectivity so that the remeshed surface and the original deviate by no more than a user-specified tolerance. This illustrated in Figure 5(c) for the bunny at level $J = 5$ of recursive subdivision. The final step of the conversion to multiresolution form is the application of k-disk analysis as in Section 3.2.

5 Surface compression

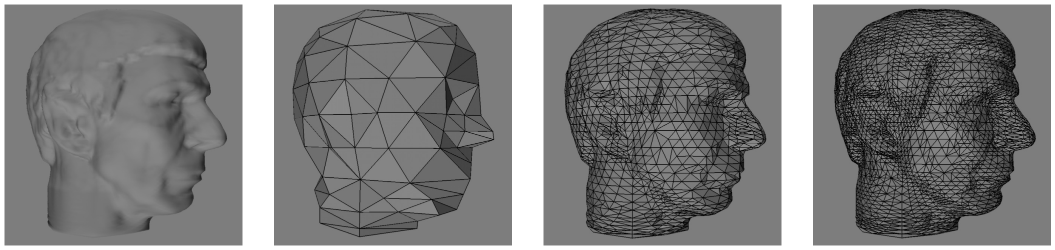
In this section, k-disk wavelets are used to address two compression applications: compression of complex surfaces, and compression of texture maps defined on surfaces.

5.1 Polyhedral compression

The first application is to compress polyhedral models such as the one shown in Figure 6(a). This particular model, consisting of 32,768 triangles, was created from range data provided by Cyberware, Inc. Since the original data was gridded and the surface was known to be topologically equivalent to a sphere, conversion to multiresolution form did not require the general parameterization algorithm of Eck *et al.* Instead, a special-purpose procedure was used to parameterize the model on an octahedral base mesh [4]. The surface was then converted to multiresolution form using recursive subdivision followed by filter-bank analysis, as described earlier.

The wavelet coefficients computed by the filter-bank algorithm are coefficients of unnormalized basis functions, so the magnitude of a coefficient is not a good measure of the least-squares error that would result if that coefficient were removed. If we multiply each wavelet coefficient d_i^j at level j by 2^{-j}, we get coefficients for an L^2 normalized basis. Just as for images, these normalized coefficients have magnitudes that are meaningful in a compression algorithm.

The surface approximations shown in Figures 6(b-d) were computed by sorting the normalized coefficients of the 2-disk wavelets, then removing 99%, 88%, and 70% of the



(a) (b) (c) (d)

Figure 6: Various compressions of Spock.

smallest-magnitude coefficients, respectively. Notice that, just as for images, this simple strategy causes the approximation to refine more deeply in areas of high detail, while leaving large triangles in areas of relatively low detail.

5.2 Texture map compression

Compression can also be applied to the representation of a texture map defined on a surface. If the surface is parameterized on the unit square, a texture map is no different from an ordinary image, and hence can be compressed using traditional wavelet techniques such as Haar or spline wavelets. However, if the surface is topologically more complicated, traditional wavelets no longer suffice, but k-disk wavelets can be used. The idea is to treat each color component—red, green, and blue—as a scalar function defined on the base mesh M^0. Each of the color functions can be converted to multiresolution form using filter-bank analysis, and the normalized wavelet coefficients can then be sorted by magnitude and truncated just as in image compression.

In the example shown in Figure 7, elevation and bathymetry data obtained from the U.S. National Geophysical Data Center were used to create a piecewise-linear coloring of the globe. The resulting color function contains 2,097,152 triangles and 1,048,578 vertices. The full-resolution coloring was too large to be rendered on a graphics workstation with 128 megabytes of memory, and is therefore not shown in its entirety.

The approximations shown in Figures 7(a-c) were produced by leaving out 2-disk piecewise-linear wavelet coefficients with magnitudes smaller than a certain threshold, resulting in compression rates of 99.9%, 98%, and 90%, respectively.

6 Display

6.1 Continuous level-of-detail control

When viewing a complex object, it is unnecessary and inefficient to draw a highly detailed representation if the observer is far away from the object. Instead, we would like to use some form of *level-of-detail control*—allowing information about the view to determine the complexity of the model that is rendered. Currently, perhaps the most common approach

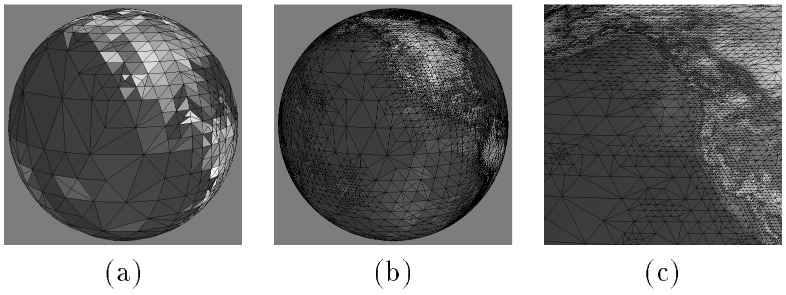


Figure 7: Texture map compression of the earth.

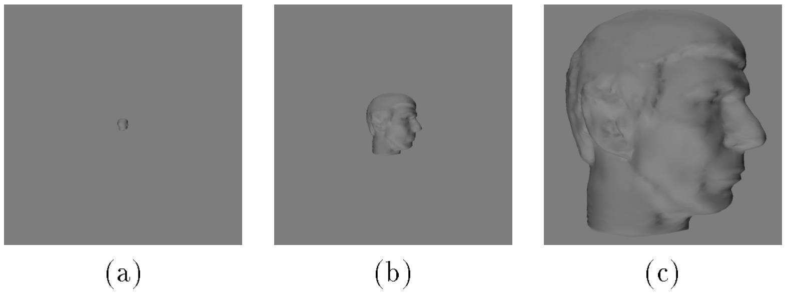


Figure 8: Level-of-detail control.

to creating LOD models is to have the user craft them by hand. By contrast, the surface compression technique described in Section 5 provides a mechanism for *automatically* producing LOD models.

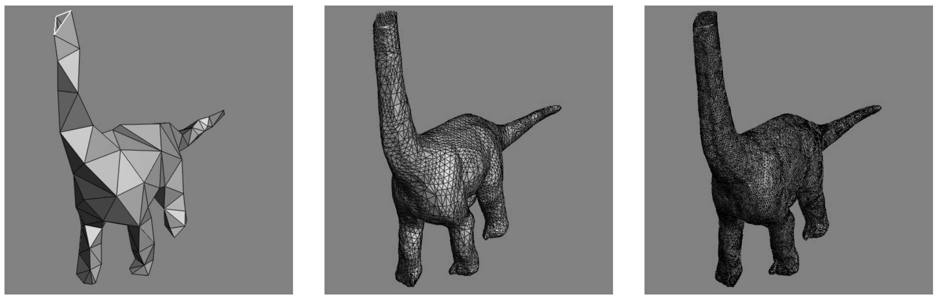
The images in Figure 8 illustrate the use of wavelet approximations for automatic level-of-detail control in rendering. When viewing the original polyhedron from distant vantage points, there is no need to render all 32,000 triangles. The compressed versions can be rendered instead. For instance, the three views shown in Figure 8 were created with the three approximations shown in Figure 6.

Switching suddenly between models with different levels of detail in an animation can produce objectionable "popping." This problem is easily solved by using continuous levels of smoothing. In effect, as the viewer approaches an object, each wavelet coefficient is smoothly varied from zero to its correct value. Likewise, as the viewer recedes, each wavelet coefficient is smoothly reduced to zero. More generally, each wavelet coefficient can be made a continuous function of the viewing distance.

6.2 Progressive transmission

Text, images, and video are commonplace on the World Wide Web, and complex geometric models are becoming very common as well. The ever-growing production and distribution of these geometric objects motivates the need for efficient transmission of models across relatively low-bandwidth networks.

The most straightforward way to transmit a complicated polyhedron is by sending



(a) (b) (c)

Figure 9: Progressive transmission: (a) the base mesh, consisting of 229 triangles; (b) the mesh after approximately 2,000 wavelet coefficients have been received; (c) the mesh after approximately 10,000 wavelet coefficients have been received. Original model courtesy of Greg Turk and Marc Levoy.

each of the triangles of the highest-resolution representation over the network. However, transmitting complex meshes in this way forces the user to wait until the entire model is received before anything can be displayed. A more attractive alternative is to use a wavelet representation for *progressive transmission*, as illustrated in Figure 9. First, the base mesh is transmitted; since this mesh contains very few triangles, it is received and displayed quickly. Next, the normalized wavelet coefficients are transmitted in order of decreasing magnitude. As these coefficients are received, the renderer can update and redisplay the model.

Certain *et al.* [1] have recently incorporated these ideas into a multiresolution viewer that operates as a helper application to Netscape.

7 Multiresolution editing

Finkelstein and Salesin [3] described a method for editing curves at multiple resolutions using a B-spline wavelet decomposition. The basic idea is that broad scale changes can be achieved by modifying a few coarse-scale wavelet coefficients, whereas narrow changes can be accommodated by modifying fine-scale coefficients. The same ideas can be applied to surfaces, as illustrated in Figure 10.

Figure 10(a) is the original model; Figure 10(b) shows the effect of changing a single scaling function coefficient at level-0. Because finer-level vertices in the same region are defined relative to the coarse shape, they move along with the modification. However, the geometry in areas away from the change is not affected. A more local change is shown in Figure 10(c) where a pair of level-3 coefficients have been changed in the neighborhood of the ears.

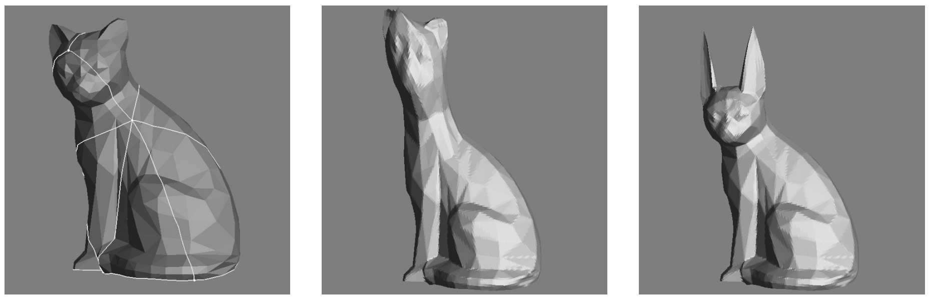

(a) (b) (c)

Figure 10: Multiresolution editing.

8 Future directions for surface wavelets

The application of multiresolution analysis to surfaces of arbitrary topology is a relatively recent development in the graphics community. The k-disk wavelets presented and applied here are certainly not the last word in multiresolution surface representations. The k-disk representation described could be enhanced in several ways:

- In the methods we've described, the wavelet decomposition of a surface always retains the topological type of the input surface. However, when the input is a relatively simple object with many small holes, it might be desirable to decompose the input into a "topologically simpler" surface, that is, one with lower genus or fewer boundary curves.
- The images in Figure 8 were generated by simply incorporating the wavelet coefficients of greatest magnitude. A view-dependent error metric could be used to produce images of better quality using even fewer triangles.
- Many objects encountered in practice have sharp discontinuities that should be preserved in coarse approximations. It may be possible to accomplish this using wavelets with discontinuities that adapt to the shape of the model being analyzed.

References

[1] Andrew Certain, Jovan Popovic, Tony DeRose, Tom Duchamp, David Salesin, and Werner Stuetzle. Interactive multiresolution surface viewing. In *SIGGRAPH 96 Conference Proceedings*, Computer Graphics Annual Conference Series, August 1996. To appear.

[2] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings*, Computer Graphics Annual Conference Series, pages 173–182, August 1995.

[3] Adam Finkelstein and David H. Salesin. Multiresolution curves. In *SIGGRAPH 94 Conference Proceedings*, Computer Graphics Annual Conference Series, pages 261–268, July 1994.

[4] J. Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington, 1994.

[5] Michael Lounsbery, Tony DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. To appear in ACM Transactions on Graphics. Preliminary version available as Technical Report 93-10-05B, University of Washington, Department of Computer Science and Engineering, October 1993.

[6] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, 1996.

[7] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH 94 Conference Proceedings*, Computer Graphics Annual Conference Series, pages 311–318, July 1994.