Multiresolution Analysis for Surfaces of Arbitrary Topological Type

Michael Lounsbery[1] Tony D. DeRose Joe Warren[2]

Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195

Abstract

Multiresolution analysis and wavelets provide useful and efficient tools for representing functions at multiple levels of detail. Wavelet representations have been used in a broad range of applications, including image compression, physical simulation, and numerical analysis.

In this paper, we present a new class of wavelets, based on subdivision surfaces, that radically extends the class of representable functions. Whereas previous two-dimensional methods were restricted to functions defined on $\mathbb{R}^2$, the subdivision wavelets developed here may be applied to functions defined on compact surfaces of arbitrary topological type. We envision many applications of this work, including continuous level-of-detail control for graphics rendering, compression of geometric models, and acceleration of global illumination algorithms. Level-of-detail control for spherical domains is illustrated using two examples: shape approximation of a polyhedral model, and color approximation of global terrain data.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - surfaces and object representations; J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD); G.1.2 [Approximation]: Spline Approximation.

Additional Keywords: Wavelets, geometric modeling, compression, level-of-detail control, subdivision surfaces, splines.

[1] Currently at Alias Research, Toronto

[2] Work done while on sabbatical from Rice University

1 Introduction.

Multiresolution analysis and wavelets have received considerable attention in recent years, fueled largely by the diverse collection of problems that benefit from their use. The basic idea behind multiresolution analysis is to decompose a complicated function into a "simpler" low resolution part, together with a collection of perturbations, called wavelet coefficients, necessary to recover the original function. For many of the functions encountered in practice, a large percentage of the wavelet coefficients are small, meaning that good approximations can be obtained by using only a few of the largest coefficients. Impressive "lossy" compression rates for images have been achieved using this type of approximation [11].

Two-dimensional wavelets are important for a variety of applications including image compression. They are generally constructed by forming tensor products of univariate wavelets [9], in much the same way that tensor product B-spline surfaces are formed by products of univariate B-splines. Unfortunately, tensor-product constructions require that the functions to be decomposed be defined on $\mathbb{R}^2$ or on a periodic version of $\mathbb{R}^2$, such as the cylinder or torus. There also exist nontensor-product constructions for wavelets on $\mathbb{R}^2$ [9, 25], but none of these methods are applicable to functions defined on more general topological domains, such as spheres. Thus, existing methods are not well suited for decomposing and compressing surfaces such as the ones shown in Figures 10 and 12, since they are described by parametric functions on the sphere.

In this paper we show that by using techniques from subdivision surfaces, multiresolution analysis can be extended to functions defined on domains of arbitrary topological type (the topological type of a two-dimensional surface or domain refers to its genus, presence of boundary curves, etc.). This generalization, which we term "subdivision wavelets," dramatically extends the class of applications to which multiresolution analysis can be applied, including:

- *Polyhedral compression.* Using wavelet-based techniques, most polyhedral models may be compressed to yield a more compact approximation. Compression saves both storage space and the time that is required to process a surface model. This dissertation develops efficient wavelet compression methods for surfaces similar to those that have proven effective for images and one-dimensional functions. These techniques are capable of L^∞ and L^2 approximation, and run more quickly than many other compression methods.

- *Continuous level-of-detail control.* When a complex shape is rendered in an animation, a fully detailed representation of the shape contains much more detail than is required for all but the closest view. Using a compressed subdivision wavelet representation of complex objects, it is possible to greatly reduce the number of polygons in a scene without significantly impacting the visible detail (see Figure 10). Moreover, it is possible to smoothly vary the detail, avoiding the discontinuous jumps that occur when suddenly switching between distinct models. This application is discussed in more detail in Section 7.5. Non-wavelet treatments of this problem may be found in Turk [37], Schroeder et al. [35], and Hoppe et al. [24].

- *Compression of functions defined on surfaces.* Consider the situation shown in Figure 12 where a globe (a geometric sphere) is pseudo-colored according to elevation. The pseudo-coloring can be thought of as a function that maps each point on the sphere to an RGB triple. A straightforward method for storing the function is to store its value at a large number of regularly distributed points; in this case more than one million points were used. The methods in this paper can be used to create compressed wavelet approximations of varying complexity. (The mesh lines on the original surface are so dense that the image shown in Figure 11(b) is nearly black.)

- *Multiresolution editing of surfaces.* Hierarchical B-splines, as introduced by Forsey and Bartels [17], provide a powerful mechanism for editing shapes at various levels of detail. However, hierarchical B-splines can only represent a restricted class of surface topologies. The methods described here provide an alternative to hierarchical B-splines, and are capable of representing smooth multiresolution surfaces of arbitrary topological type. Editing at fractional levels of detail can also be achieved using the methods developed by Finkelstein and Salesin [16].

- *Global illumination.* Tensor product wavelets have been used to rapidly solve physical simulation problems such as radiosity [19, 21]. The use of wavelets in the more general setting of spherical radiance distributions [18] requires wavelets adapted to the sphere. Wavelets with a spherical domain were were used to create the images in Figure 12.

- *Polygon clustering for hierarchical radiosity.* An important method for accelerating hierarchical radiosity involves the clustering of groups of polygons into an appropriate approximation [36]. Subdivision wavelets offer a wavelet-based method for a limited form of clustering.

- *Surface optimization.* The multiple levels of approximation produced by wavelet techniques offer a sort of multigrid technique for optimization. Pentland [33] uses wavelet methods to implement multigrid optimization for surface interpolation over regular grids. Meyers [31, 32] shows how wavelets can accelerate the reconstruction of surfaces from contour data. The techniques of subdivision wavelets described in this dissertation can be used to accelerate optimization over surfaces of arbitrary topological type.

A carefully constructed special-purpose algorithm may produce superior results for some of these applications. However, the above examples demonstrate the versatility of subdivision wavelets, and show them to be a reusable tool that naturally and efficiently solve or accelerate a wide range of common problems in computer graphics and modeling. As an illustration of this versatility, a complex object of arbitrary toplogical type may be modeled at multiple levels of detail, stored in a compressed form, viewed in an animation, and rendered using global illumination — all naturally implemented using the same underlying surface wavelet representation.

This paper presents a theoretical foundation for developing multiresolution analysis for surfaces of arbitrary topological type. Because the theory that we present is entirely motivated by its practical applications, special care is taken to place our work in the context

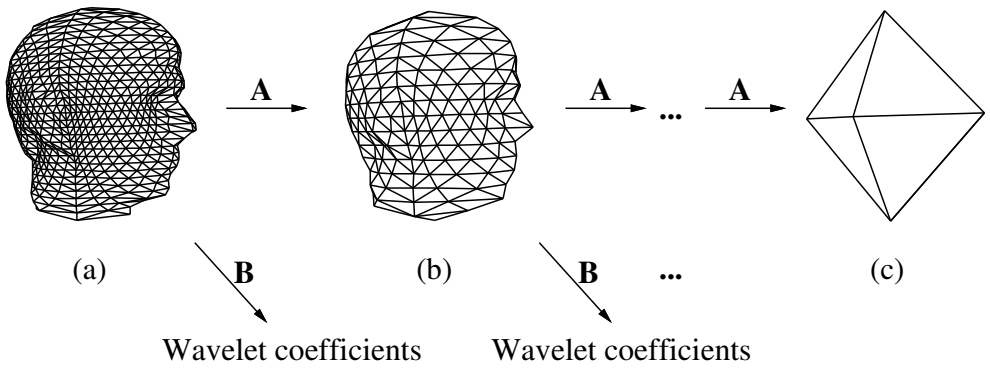


Figure 1: *Decomposition of a polyhedral surface.*

of its efficient implementation. We therefore purposely avoid problems of theoretical interest in which we see little to gain in practice. At the same time, we intentionally omit many details of implementation, except to outline how a particularly inefficient implementation may be avoided. Additional details of implementation and an expanded treatment of applications are discussed in a related PhD thesis [27].

The construction of subdivision wavelets described herein applies directly to a variety of already existing subdivision schemes. These include piecewise linear subdivision (which produces polyhedra); the schemes of Catmull & Clark [3, 20], Loop [26] or Dyn et al. [14] (which produce tangent-plane-smooth G^1 subdivision surfaces); and the modifications by Hoppe *et al.* [23, 22] that produce piecewise smooth surfaces with selected discontinuities. More generally, the techniques presented here may be used to construct wavelets for any uniformly convergent local and continuous stationary subdivision rule; that is, any locally defined stationary subdivision rule that converges to a continuous and integrable surface.

Together with the theoretical development, two applications of the theory are described in Section 7.5. Even though these examples use C^0 polyhedral surfaces, we emphasize that our methods are equally applicable to tangent-plane smooth subdivision surfaces, piecewise-smooth subdivision surfaces [23], and to open surfaces possessing boundary curves.

2 A Preview of the Method.

Although the mathematical underpinnings of multiresolution analysis of surfaces are somewhat involved, the resulting algorithms are quite simple. Before diving into the details, we give here a brief description of how the method can be applied to decompose the polyhedral object shown in Figure 1(a).

As mentioned in Section 1, a main idea behind multiresolution analysis is the decomposition of a function (in this case a polyhedron expressed as a parametric function on the sphere) into a low resolution part and a "detail" part. The low resolution part of the polyhedron in Figure 1(a) is shown in Figure 1(b). The vertices in (b) are computed as certain weighted averages of the vertices in (a). These weighted averages essentially implement

a low pass filter denoted as **A**. The detail part consists of a collection of fairly abstract coefficients, called wavelet coefficients, computed as weighted differences of the vertices in (a). These differencing weights form a high-pass filter **B**. The decomposition process, technically called *analysis*, can be used to further split (b) into an even lower resolution version and corresponding wavelet coefficients. This cascade of analysis steps, referred to as a *filter bank* algorithm, culminates with the coarsest-level representation in (c), together with wavelet coefficients at each level.

The analysis filters **A** and **B** are constructed so that the original polyhedron can be recovered exactly from the low-resolution version and the wavelet coefficients. Recovery, technically called *synthesis*, reconstructs (a) from (b) and the finest-level wavelet coefficients. Recovery refines each triangle of (b) into four subtriangles by introducing new vertices at edge midpoints, followed by perturbing the resulting collection of vertices according to the wavelet coefficients. The refining and perturbing steps are described by two other filters **P** (the refining filter) and **Q** (the perturbing filter), collectively called synthesis filters.

The trick is to develop the four analysis and synthesis filters so that: (1) the low-resolution versions are good approximations of the original object (in a least-squares sense); (2) the magnitude of a wavelet coefficient reflects a coefficient's importance by measuring the error introduced when the coefficient is set to zero; and (3) analysis and synthesis filter banks should have time complexity that is linear in the number of vertices.

3 Background on Multiresolution Analysis.

Multiresolution analysis as formulated by Mallat [29] provides a convenient framework by which to develop the analysis and synthesis filters. There are two basic ingredients for a multiresolution analysis: an infinite chain of nested linear function spaces $V^0 \subset V^1 \subset V^2 \subset \cdots$ and an inner product $\langle f, g \rangle$ defined on any pair of functions $f, g \in V^j$, for some $j < \infty$. Intuitively, V^j contains functions of resolution j, with the detail increasing as j increases.

The inner product is used to define the orthogonal complement spaces W^j as

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

Orthogonal complements are often written as $V^{j+1} = V^j \oplus W^j$ because any function $f^{j+1} \in V^{j+1}$ can be written uniquely as an orthogonal decomposition

$$f^{j+1} = f^j + h^j,$$

where $f^j \in V^j$ and $h^j \in W^j$. Orthogonal decompositions are important for approximation purposes: it is easy to show that f^j is the best approximation to f^{j+1} in that it is the unique function in V^j that minimizes the least-squares residual $\langle f^{j+1} - f^j, f^{j+1} - f^j \rangle$. Thus, given a high-resolution function f^{j+1}, its low-resolution part is f^j, and its detail part is h^j.

The following terminology is now standard: *scaling functions* refers to bases for the spaces V^j, and *wavelets* refers to bases for the orthogonal complement spaces. As shown in Section 6.4, the analysis and synthesis filters are determined by considering various ways of changing bases between scaling functions and wavelets.

4 Nested Linear Spaces Through Subdivision.

A fundamental requirement for multiresolution analysis is a sequence of nested linear spaces. In this section, we carry this property to surfaces of arbitrary genus, demonstrating the existence of scaling functions on subdivision surfaces. In Section 5, we show how these scaling functions may be efficiently and exactly integrated so as to compute the inner product functions necessary for orthogonality. In Section 6, we develop wavelets computed from the inner product, and we produce locally supported approximations to them which are more appropriate for practical use. Our methods may be used in general to construct wavelets from any local subdivision procedure which constructs a continuous and integrable surface in the limit.

The nested sequence of linear spaces required by multiresolution analysis are ordinarily obtained by defining a single scaling function $\phi(x)$ that satisfies a *refinement equation* of the form

$$\phi(x) = \sum_i p_i \; \phi(2x - i)$$

for some fixed constants p_i. The refinement equation (sometimes called a two-scale relation) guarantees that the spaces defined as

$$V^j := \text{Span}\{\phi(2^j x - i) \mid i = -\infty, ..., \infty\}$$

are nested. In other words, the nested spaces are generated by translations and dilations of a single refinable function $\phi(x)$.

To generalize these ideas to domains of arbitrary topological type, one could attempt to make definitions for what it means to dilate and translate a function on an arbitrary topological domain. One could then try to find a refinable scaling function and proceed as before to define orthogonal complements, wavelets, and so on. We have instead chosen what appears to be a simpler approach.

In this section, recursive subdivision is used to define a collection of functions $\phi_i^j(\mathbf{x})$ that are refinable — in the sense that each function with superscript j lies in the span of the functions with superscript $j+1$; the argument $\mathbf{x}$ is a point that ranges over a domain 2-manifold of arbitrary topological type. In one respect, this is a generalization of the approach taken by Daubechies [9], whose locally supported orthogonal scaling functions are also defined through a recursive subdivision procedure. Although in general the $\phi^{j+1}(\mathbf{x})$ are not simple dilates of the $\phi^j(\mathbf{x})$, one can nonetheless use them to define a sequence of nested spaces.

4.1 Subdivision Surfaces.

Intuitively speaking, subdivision surfaces are defined by iteratively refining a control polyhedron M^0 so that the sequence of increasingly faceted polyhedra $M^1, M^2, ...$ converge to some limit surface $S = M^\infty$. In each subdivision step, the vertices of M^{j+1} are computed as affine combinations of the vertices of M^j. Thus, if $\mathbf{V}^j$ is a matrix whose i-th row consists of

the x, y, and z coordinates of vertex i of M^j, there exists a non-square matrix of constants $\mathbf{P}^j$ such that

$$\mathbf{V}^{j+1} = \mathbf{P}^j \mathbf{V}^j. \tag{1}$$

The matrix $\mathbf{P}^j$ therefore characterizes the subdivision method. The beauty of subdivision surface schemes is that the entries of $\mathbf{P}^j$ depend only on the connectivity of the vertices in M^0, not on the geometric positions of the vertices.

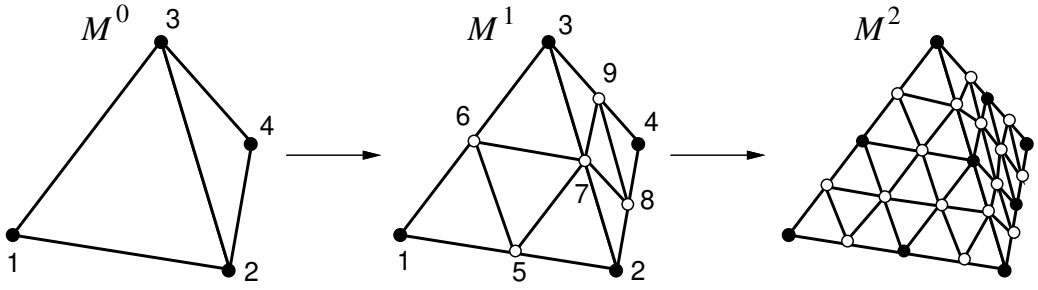
The simplest example of such a scheme is polyhedral subdivision. Given a polyhedron M^0 with triangular faces, a new polyhedron M^1 is built by splitting each triangular face of M^0 into four subfaces as in Figure 2. The matrix $\mathbf{P}^0$ characterizing the first subdivision step is also shown in Figure 2. Running this subdivision scheme for j steps on an initial triangular mesh M^0 produces a mesh M^j. M^j includes the vertices of M^0 together with new vertices introduced through subdivision. The valence (the number of edges incident to a vertex) of the vertices of M^j corresponding to the original vertices in M^0 remains fixed. The new vertices introduced through subdivision however, are always of valence six, corresponding to a regular triangular tiling of the surface. As the mesh is further subdivided, the so-called *extraordinary* points (any original vertex of valence other than six) become increasingly isolated in an otherwise regular tiling of the surface.

Polyhedral subdivision converges to the original polyhedron covering M^0, that is, to a C^0 surface. However, other schemes have been developed that converge to tangent-plane smooth limit surfaces that either approximate or interpolate the vertices of M^0. Subdivision schemes can be further categorized as being either *primal* or *dual*. A subdivision scheme is primal if the faces of the mesh are split into subfaces by the refinement procedure. Catmull-Clark subdivision [3, 20] is a primal scheme based on subdivision of quadrilateral faces. Polyhedral subdivision, the butterfly scheme of Dyn, Gregory, and Levin [14] and Loop's method [26] are primal schemes based on subdivision of triangular faces. A scheme is dual if the structure of the refined mesh is obtained by doing a primal subdivision followed by taking the polyhedral dual of the result. Doo-Sabin subdivision [13, 12] is a dual scheme based on quadrilaterals.

Although one can certainly construct fractal-like subdivision schemes that do not converge to well defined surfaces, in this paper we shall consider only local schemes that uniformly converge to a surface no matter where the control vertices are placed (see DeRose *et al.* [10] for a discussion of conditions under which a scheme is uniformly convergent). In the remainder of the paper, subdivision schemes will be assumed to be local and uniformly convergent. For simplicity we shall also restrict the discussion to primal triangular subdivision schemes, although our results hold more generally for any uniformly convergent primal subdivision scheme.

4.2 Refinable Basis Functions Through Subdivision.

We show here that subdivision can be used to define a collection of refinable basis functions. We do this by first showing that subdivision surfaces can be parametrized by a function $S(\mathbf{x})$, where $\mathbf{x}$ ranges over M^0. We then show that the parametrization can be used to define the scaling functions. Parametrizing the scaling functions over a domain M^0 of arbitrarily



$$\mathbf{P}^0 = \left(\frac{\mathbf{O}^0}{\mathbf{N}^0}\right) = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{array}\right) \quad \mathbf{Q}^0 = \left(\begin{array}{cccccc} -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & -\frac{3}{8} \\ -\frac{3}{8} & \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & -\frac{3}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & -\frac{3}{8} \\ \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{1}{8} \\ -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} \\ \frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{5}{8} \end{array}\right)$$

$$\left(\frac{\mathbf{A}^0}{\mathbf{B}^0}\right) = \left(\begin{array}{cccccccccc} \frac{7}{16} & -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} \\ -\frac{1}{16} & \frac{7}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{16} & \frac{7}{16} & -\frac{1}{16} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{7}{16} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{3}{8} \\ \hline -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

$$\mathbf{I}^0 = \left(\begin{array}{cccc} 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 1 \end{array}\right) \quad \boldsymbol{\alpha}^0 = \left(\begin{array}{cccccc} \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} \\ \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{3}{8} \end{array}\right)$$

Figure 2: *Polyhedral subdivision of a tetrahedron and various associated filters.*

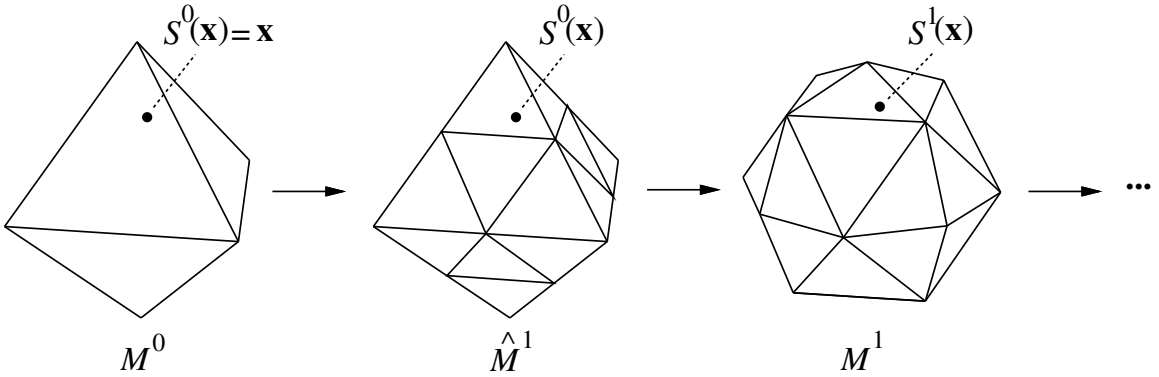


Figure 3: *The subdivision limiting process.*

topological type differs sharply from the more usual method of parametrizing surface basis functions over a piece of the plane.

In general terms, a surface parametrization is nothing more than a correspondence between points in a two-dimensional domain and points on the surface. The idea behind establishing a parametrization for a subdivision surface is to track a point $\mathbf{x}$ on M^0 through the subdivision process (see Figure 3). The point $\mathbf{x}$ being tracked will converge to a point on the limit surface, thereby establishing a correspondence S between points on M^0 and points on the limit surface. We now make these ideas more precise.

For primal subdivision schemes, the refinement step that carries the mesh M^{s-1} into M^s consists of two substeps: the *splitting* step and the *averaging* step. In the splitting step, each face of M^{s-1} is split into four subfaces by introducing vertices at midpoints of edges, creating an auxiliary mesh $\widehat{M}^s$, as shown in Figure 3. The averaging step uses local weighted averaging to compute the vertex positions of M^s from the vertex positions of $\widehat{M}^s$. All primal subdivision schemes share the splitting step — they differ only in the weights used in the averaging step.

The parametrization $S(\mathbf{x})$ can now be established by using a limiting process. The limiting process producing $S(\mathbf{x})$ consists of three steps:

1. $S^0(\mathbf{x}) := \mathbf{x}, \mathbf{x} \in M^0$.

2. Suppose that $S^{s-1}(\mathbf{x})$ lies in triangle $(\widehat{v}_a^s, \widehat{v}_b^s, \widehat{v}_c^s)$ of $\widehat{M}^s$ with barycentric coordinates (α, β, γ). Then
$$S^s(\mathbf{x}) := \alpha v_a^s + \beta v_b^s + \gamma v_c^s,$$
where (v_a^s, v_b^s, v_c^s) is the triangle of M^s corresponding to $(\widehat{v}_a^s, \widehat{v}_b^s, \widehat{v}_c^s)$ of $\widehat{M}^s$.

3. $S(\mathbf{x}) := \lim_{s\to\infty} S^s(\mathbf{x})$.

Next, we show that the parametrization $S(\mathbf{x})$ induces a collection of refinable scaling functions.

Lemma 1 *For all $j \geq 0$ and $s \geq j$ there exist row vectors $\Phi^{s \leftarrow j}(\mathbf{x})$, $\Phi^{s \leftarrow j} : M^0 \rightarrow \mathbb{R}$ such that*

$$S^s(\mathbf{x}) = \Phi^{s \leftarrow j}(\mathbf{x}) \mathbf{V}^j,$$

where $\mathbf{V}^j$ denotes the column vector of vertices of M^j.

Proof: The linear combination of Item 2 above can be rewritten in matrix notation as

$$S^s(\mathbf{x}) = \mathbf{b}^s(\mathbf{x}) \mathbf{V}^s,$$

where $\mathbf{b}^s(\mathbf{x})$ is the barycentric coordinate vector of $\mathbf{x}$ with respect to M^s; that is,

$$\mathbf{b}^s(\mathbf{x}) = (0 \cdots 0 \ \alpha \ 0 \cdots 0 \ \beta \ 0 \cdots 0 \ \gamma \ 0 \cdots 0),$$

where α occurs at index a, β occurs at index b, γ occurs at index c. At each refinement step $k = 1, ..., s$, the vertices of M^k can be computed from affine combinations of the vertices of M^{k-1}. Therefore, there must exist a chain of (non-square) matrices $\mathbf{P}^0, ..., \mathbf{P}^{k-1}$ such that

$$\mathbf{V}^k = \mathbf{P}^{k-1} \mathbf{P}^{k-2} \cdots \mathbf{P}^0 \mathbf{V}^0.$$

Thus,

$$S^s(\mathbf{x}) = \mathbf{b}^s(\mathbf{x}) \mathbf{P}^{s-1} \mathbf{P}^{s-2} \cdots \mathbf{P}^j \mathbf{V}^j.$$

The desired result follows immediately by making the definition

$$\Phi^{s \leftarrow j}(\mathbf{x}) := \mathbf{b}^s(\mathbf{x}) \mathbf{P}^{s-1} \mathbf{P}^{s-2} \cdots \mathbf{P}^j. \ \square$$

As a simple corollary to Lemma 1, we note that

$$\Phi^{s \leftarrow j}(\mathbf{x}) = \Phi^{s \leftarrow j+1}(\mathbf{x}) \mathbf{P}^j. \tag{2}$$

Theorem 1 *For any subdivision procedure, and for any $j \geq 0$, there exist scalar-valued scaling functions $\phi_i^j(\mathbf{x})$, $\mathbf{x} \in M^0$ such that*

$$S(\mathbf{x}) = \sum_i v_i^j \phi_i^j(\mathbf{x}). \tag{3}$$

Proof: Using Lemma 1 and the definition of $S(\mathbf{x})$:

$$S(\mathbf{x}) = \lim_{s \rightarrow \infty} \left(\Phi^{s \leftarrow j} \mathbf{V}^j \right).$$

Since the subdivision scheme is assumed to be local and uniformly convergent for any choice of control points, $S(\mathbf{x})$ must exist if we choose all entries of $\mathbf{V}^j$ to be the origin, except for v_i^j, the i-th entry of $\mathbf{V}^j$. This choice of control points leads to a surface of the form

$$\left(\lim_{s \rightarrow \infty} \phi_i^{s \leftarrow j}(\mathbf{x}) v_i^j \right) = v_i^j \left(\lim_{s \rightarrow \infty} \phi_i^{s \leftarrow j}(\mathbf{x}) \right).$$

Since the surface is well defined, the sequence on the right must converge to $v_i^j \phi_i^j(\mathbf{x})$, where

$$\phi_i^j(\mathbf{x}) := \lim_{s \to \infty} \phi_i^{s \leftarrow j}(\mathbf{x}).$$

By linearity, the surface for an arbitrary set of control points can be written as in Equation 3. □

We will find it convenient to rewrite Equation 3 in matrix form as

$$S(\mathbf{x}) = \Phi^j(\mathbf{x})\, \mathbf{V}^j, \tag{4}$$

where $\Phi^j(\mathbf{x})$ denotes the row matrix of scaling functions $\phi_i^j(\mathbf{x})$, and where $\mathbf{V}^j$ is as in Equation 1. Equation 4 shows an analogy with B-splines, where the $\phi_i^j(\mathbf{x})$ are comparable to the B-spline basis functions, and the $\mathbf{V}^j$ are akin to the control points.

As a corollary to Theorem 1, if the subdivision scheme generates continuous surfaces, the scaling functions ϕ_i^j are continuous, hence they are also integrable [1].

We may now establish the refinability of the scaling functions defined in Theorem 1.

Theorem 2 *The scaling functions $\phi_i^j(\mathbf{x})$ are refinable.*

Proof: Starting with Equation 2 and taking limits as s tends toward infinity, it follows from the existence of the $\phi_i^j(\mathbf{x})$ and the continuity of matrix products that

$$\Phi^j(\mathbf{x}) = \Phi^{j+1}(\mathbf{x})\mathbf{P}^j. \tag{5}$$

This equation establishes refinability since it states that each of the functions $\phi_i^j(\mathbf{x})$ can be written as a linear combination of the functions $\phi_i^{j+1}(\mathbf{x})$. □

For primal subdivision schemes, it is convenient to write Equation 5 in block matrix form by writing $\Phi^{j+1}(\mathbf{x})$ as

$$\Phi^{j+1}(\mathbf{x}) = (\mathcal{O}^{j+1}(\mathbf{x})\ \ \mathcal{N}^{j+1}(\mathbf{x})), \tag{6}$$

where $\mathcal{O}^{j+1}(\mathbf{x})$ consists of all scaling functions $\phi_i^{j+1}(\mathbf{x})$ associated with the old vertices of M^j (the black vertices in Figure 2) and $\mathcal{N}^{j+1}(\mathbf{x})$ consists of the remaining scaling functions associated with the new vertices added when obtaining M^{j+1} from M^j (the white vertices in Figure 2). Equation 5 can now be expressed in block matrix form:

$$\Phi^j(\mathbf{x}) = (\mathcal{O}^{j+1}(\mathbf{x})\ \ \mathcal{N}^{j+1}(\mathbf{x})) \begin{pmatrix} \mathbf{O}^j \\ \mathbf{N}^j \end{pmatrix}, \tag{7}$$

where $\mathbf{O}^j$ and $\mathbf{N}^j$ represent the portions of the subdivision matrix $\mathbf{P}^j$ which weight the "old" and "new" vertices, respectively. The block matrix decomposition of $\mathbf{P}^0$ for the example tetrahedron appears in Figure 2.

4.3 Nested Linear Spaces.

Given these relations, a chain of nested linear spaces $V^j(M^0)$ associated with a mesh M^0 can now be defined as follows:

$$V^j(M^0) := \text{Span}(\Phi^j(\mathbf{x})),$$

where the $V^j(M^0)$ are spaces of scalar-valued functions.

Equation 5 implies that these spaces are indeed nested; that is,

$$V^0(M^0) \subset V^1(M^0) \subset \cdots.$$

The notation $V^j(M^0)$ is to emphasize that the linear spaces are adapted to M^0 in that they consist of functions having M^0 as the domain.

5 Inner Products.

Given a chain of nested linear spaces, the other necessary ingredient for the creation of a multiresolution analysis is the existence of an inner product on these spaces. In this section, we define an inner product and give a method for exactly computing the inner product of functions defined through any uniformly convergent local and continuous subdivision procedure.

The inner product values are used to build linear systems defining the wavelets. For an efficient implementation of subdivision wavelets, the actual values of neither the inner products nor the wavelets need to be computed at run time, if the base mesh M^0 is known in advance.

5.1 Definition.

Let two functions $f, g \in V^j(M^0)$, $j < \infty$ be linear combinations of (scalar-valued) scaling functions defined through subdivision, as in Section 4. When the subdivision procedure is compactly supported, continuous, and uniformly convergent, f and g must be continuous, and hence, integrable.

We may define the inner product of f and g to be

$$\langle f, g \rangle := \int_{\mathbf{x} \in M^0} f(\mathbf{x})\ g(\mathbf{x})\ d\mathbf{x}, \tag{8}$$

where the area form $d\mathbf{x}$ is taken to be the area for a triangulation homeomorphic to M^0 consisting of equilateral triangles with unit length sides. Equivalently, the inner product can be expressed as

$$\langle f, g \rangle := \sum_{\tau \in \Delta(M^0)} \frac{1}{\text{Area}(\tau)} \int_{\mathbf{x}' \in \tau} f(\mathbf{x}')\ g(\mathbf{x}')\ d\mathbf{x}',$$

where $\Delta(M^0)$ denotes the set of triangular faces of M^0, and where $d\mathbf{x}'$ is the usual Euclidean area form for the triangle τ in $\mathbb{R}^3$.

Our definition of inner product implies that triangles of different geometric size and shape are weighted equally; that is, the inner product is independent of the geometric positions of the vertices of M^0. This inner product has two important consequences.

First, in the process of constructing the least-squares best wavelet approximation to a function, each approximated triangle is weighted equally, independent of its true geometric size. The effect of this weighting depends on the particular application, but we have found no problems for the real-world examples described in Section 7.5.

Moreover, this measure has an important practical benefit. Because the orthogonal complement spaces are invariant of the geometry of the mesh, a significant amount of precomputation of inner products and wavelets can be performed — which allows the wavelet algorithms to be implemented much more efficiently.

An alternative is to define the inner product so as to weight the integral by the areas of triangles in M^0. Whether such a definition has enough important practical benefit to offset its much increased computation may be an interesting topic for future research.

5.2 Computation.

For piecewise linear subdivision (leading to polyhedral surfaces), the scaling functions $\phi_i^j(\mathbf{x})$ are simply the hat functions over M^0. For the case when functions f and g are combinations of piecewise linear scaling functions, it is therefore fairly simple to directly compute the integral of Equation 8.

In general, however, the lack of a closed form for the scaling functions makes explicit integration quite difficult. In these cases, one could estimate the inner product $\langle f, g \rangle$ by subdividing the scaling functions some number of times and directly integrating the resulting piecewise linear approximation. In this section, we will see that such estimation is unnecessary, and that exact integration is still possible for the general case.

One way to compute inner products is to follow the approach in Halstead *et al.* [20]. Their approach for computing integrals over subdivision surfaces was to observe that away from extraordinary points, integrals could be computed exactly since Catmull-Clark subdivision converges to uniform bicubic B-splines in *regular* regions (those regions removed from extraordinary points). They also observed that the subdivision process could then be used to compute an integral over the entire surface by evaluating a geometric series. To generalize their method to other schemes, one must first compute inner products in regular regions. This is relatively easy for some schemes, such as Loop's, that converge to polynomials in regular regions, but it is harder for schemes such as the butterfly method that are nowhere polynomial.

A general method for integration in regular regions was recently developed by Dahmen and Micchelli [8]. It operates by reducing the problem of computing inner products to one of computing eigenvectors of a matrix defined by the refinement equations.

Although it is possible to combine the methods of Halstead *et al.* and Dahmen and Micchelli, we present here a method that is somewhat simpler, in that it computes inner products directly as the solution to a homogeneous system of linear equations.

For any ϕ_i^j constructed from a uniformly convergent subdivision procedure, one may

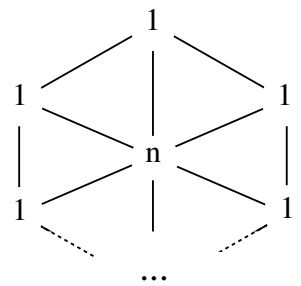


Figure 4: *The polyhedral inner product mask.*

compute $\langle f, g \rangle$ exactly whenever f and g are given as expansions in ϕ_i^j:

$$f(\mathbf{x}) = \sum_i f_i^j \phi_i^j(\mathbf{x}) \qquad\qquad g(\mathbf{x}) = \sum_i g_i^j \phi_i^j(\mathbf{x}).$$

Bi-linearity of the inner product allows $\langle f, g \rangle$ to be written in matrix form as

$$\langle f, g \rangle = \mathbf{g}^T \mathbf{I}^j \mathbf{f},$$

where $\mathbf{f}$ and $\mathbf{g}$ are column matrices consisting of the coefficients of f and g, respectively, and where $\mathbf{I}^j$ is the square matrix whose i, i'-th entry is $(\mathbf{I}^j)_{i,i'} = \langle \phi_i^j, \phi_{i'}^j \rangle$. The inner product matrix $\mathbf{I}^0$ for the example tetrahedron appears in Figure 2.

When the scaling functions in $\mathbf{\Phi}^j(\mathbf{x})$ are locally supported, the subdivision matrices $\mathbf{P}^j$ are sparse, meaning that $\mathbf{I}^j$ is also sparse. $\mathbf{I}^j$ can be computed exactly by solving a system of linear equations; hence, $\langle f, g \rangle$ can also be computed exactly. This result is somewhat surprising because there is no closed-form expression for the scaling functions — they are known only as limit functions defined through subdivision.

The i-th row of $\mathbf{I}^j$ contains the inner product of ϕ_i^j with each of the other scaling functions $\phi_{i'}^j$. It is convenient to view these entries geometrically by constructing an *inner product mask* around each vertex. The inner product mask for the i-th vertex of M^j assigns to each vertex i' of M^j a multiple of the scalar $\langle \phi_i^j, \phi_{i'}^j \rangle$.

In the case of polyhedral subdivision, explicit calculation leads to the inner product mask around a vertex of valence n shown in Figure 4, where the central weight is simply the valence of the central vertex.

For more general subdivision procedures, such as Dyn, Levin and Gregory's butterfly scheme, the limit surface has no closed form, precluding a brute-force explicit integration. However, if the subdivision scheme is local, that is, if the averaging mask of the scheme has strictly local support, it is possible to exactly determine the entries of $\mathbf{I}^j$ by solving a linear system, without resorting to numerical integration.

The key to this linear system is to observe that a recurrence relation exists between $\mathbf{I}^j$ and $\mathbf{I}^{j+1}$: $\mathbf{I}^j$ can be written as

$$\mathbf{I}^j = \int_{\mathbf{x} \in M^0} (\mathbf{\Phi}^j(\mathbf{x}))^T \, \mathbf{\Phi}^j(\mathbf{x}) d\mathbf{x}, \tag{9}$$

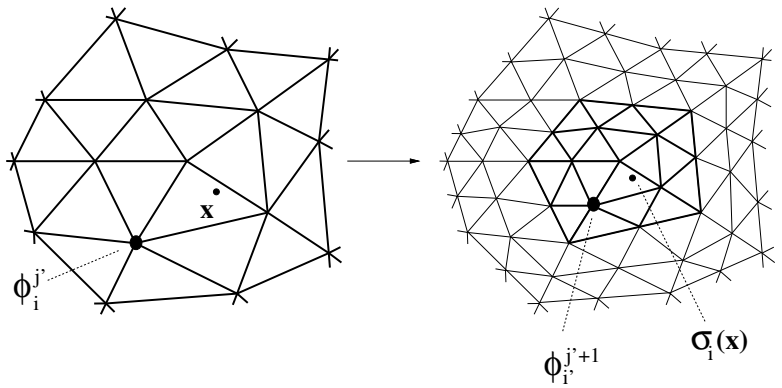


Figure 5: *The diagram on the left depicts the triangulation of M^0 produced after j' subdivision steps, that is, after j' recursive midpoint splits; the diagram on the right corresponds to $j'+1$ subdivision steps. $\phi_i^{j'}$ denotes the scaling function for the i-th vertex of $M^{j'}$, and $\phi_{i'}^{j'+1}$ denotes the scaling function for the corresponding i'-th vertex of $M^{j'+1}$. The map $\sigma_i(\mathbf{x})$ is such that the barycentric coordinates of $\mathbf{x}$ and $\sigma_i(\mathbf{x})$ within their respective surrounding triangles are equal.*

where the integrand represents a matrix outer product, and where the integral of a matrix of functions is defined to be the matrix of integrals. The refinement property of Equation 5 can now be used to establish the recurrence

$$\begin{aligned} \mathbf{I}^j &= \int_{\mathbf{x}\in M^0} (\mathbf{P}^j)^T \, (\Phi^{j+1}(\mathbf{x}))^T \, \Phi^{j+1}(\mathbf{x}) \, \mathbf{P}^j d\mathbf{x} \\ &= (\mathbf{P}^j)^T \, \mathbf{I}^{j+1} \, \mathbf{P}^j. \end{aligned}$$

If the subdivision procedure is local, the support of any particular $\phi_i^j(\mathbf{x})$ overlaps with only a constant number of other such functions on the same level. As the subdivision continues, the scaling function support shrinks at each finer level. Thus, after some number of subdivision steps j' the support of each of the $\phi_i^{j'}(\mathbf{x})$ will contain at most one extraordinary point. Furthermore, the local neighborhood at level $j'+1$ will be a shrunken version of the local neighborhood at level j'. Hence, the scaling functions in $\Phi^{j'}(\mathbf{x})$ and $\Phi^{j'+1}(\mathbf{x})$ are related.

More precisely, for each scaling function $\phi_i^{j'}(\mathbf{x})$ there is at least one i' such that

$$\phi_i^{j'}(\mathbf{x}) = \phi_{i'}^{j'+1}(\sigma_i(\mathbf{x})),$$

where $\sigma_i : M^0 \to M^0$ is a piecewise linear function that maps triangles in the support of $\phi_i^{j'}(\mathbf{x})$ into the corresponding triangles of $\phi_{i'}^{j'+1}(\mathbf{x})$, as depicted in Figure 5. Because triangle areas of M^0 shrink by a factor of 4 under subdivision, the Jacobian of σ_i is $\frac{1}{4}$. Consequently, each of the entries $(\mathbf{I}^{j'})_{hi}$ in $\mathbf{I}^{j'}$ has one or more corresponding entries $(\mathbf{I}^{j'+1})_{h'i'}$ in $\mathbf{I}^{j'}$, up to a factor of $\frac{1}{4}$; that is, $\frac{1}{4}(\mathbf{I}^{j'})_{hi} = (\mathbf{I}^{j'+1})_{h'i'}$.

The resulting $m \times m$ matrix equation

$$\mathbf{I}^{j'} = (\mathbf{P}^{j'})^T \, \mathbf{I}^{j'+1} \, \mathbf{P}^{j'} \tag{10}$$

represents a homogeneous system of m^2 equations in the m^2 unknown entries of $\mathbf{I}^{j'}$. Due to the symmetry of $\mathbf{I}^{j'}$, the system reduces to $m(m+1)/2$ homogeneous equations in as many unknowns. Although we have been unable to prove that the system is uniquely solvable (equivalently, that the system matrix is rank deficient by one), this has been the case for all examples that we have tried, including all those used to generate the figures in this paper. We therefore conjecture that under mild conditions on the subdivision scheme the system is uniquely solvable.

Thus, once an absolute scale for the homogeneous system is chosen, we conjecture that the system may be uniquely solved. This scale may be set by adding the condition that the sum of the entries of $\mathbf{I}^j$ must be 1, which provides the normalization necessary to turn the homogeneous system into an inhomogeneous one.

As an additional comment, for the common case when the subdivision procedure is local, the support of any particular $\phi_i^j(\mathbf{x})$ overlaps with only a constant number of other such functions on the same level; hence, the matrix $\mathbf{I}^{j'}$ is sparse.

Once the entries of $\mathbf{I}^{j'}$ have been determined, the remaining inner product matrices $\mathbf{I}^{j'-1}, \mathbf{I}^{j'-2}, ..., \mathbf{I}^0$ can be successively determined via Equation 10.

As an example of this process, consider the case when the $\phi_i^j(x)$ are piecewise linear functions parametrized over the infinite real line, as illustrated in Figure 6. For this case, a single function $\phi_{2i}^j(x)$ with knots on the even integers is refinable in terms of finer-scale versions with knots on the integers:

$$\phi_{2i}^j(x) = \frac{1}{2}\phi_{i-1}^{j-1}(x) + \phi_i^{j-1}(x) + \frac{1}{2}\phi_{i+1}^{j-1}(x). \tag{11}$$

For any level j, these simple functions lead to only two nonzero cases of inner products:

1. $\langle \phi_i^j(x), \phi_i^j(x) \rangle$ — the inner product of a scaling function with itself.

2. $\langle \phi_i^j(x), \phi_{i+1}^j(x) \rangle$ — the inner product of a scaling function with its neighbor.

All other possible inner products are either equivalent to one of these, or zero, due to the local support of $\phi_i^j(x)$.

Choosing our scale such that case 1 evaluates to 1, exact integration of the inner product for these piecewise linear functions shows that the inner product value for case 2 is $\frac{1}{4}$.

For the purpose of illustration, we will derive these same values using the techniques presented earlier in this section. We will do this by creating a homogeneous system of equations based on the refinement equations. This system may then be solved for the inner product values, subject to an arbitrary scale.

We define the unknown inner product values for case 1 and case 2 to be

$$\begin{aligned} x_1 &:= \langle \phi_{2i}^j(x), \phi_{2i}^j(x) \rangle \\ x_2 &:= \langle \phi_{2i}^j(x), \phi_{2(i+1)}^j(x) \rangle. \end{aligned} \tag{12}$$

Figure 6(a) illustrates case 2 — the inner product of two neighboring piecewise linear scaling functions at level j with knots on the even integers. The second equation in 12 is shown as the inner product of the solid function with the dashed-line function.

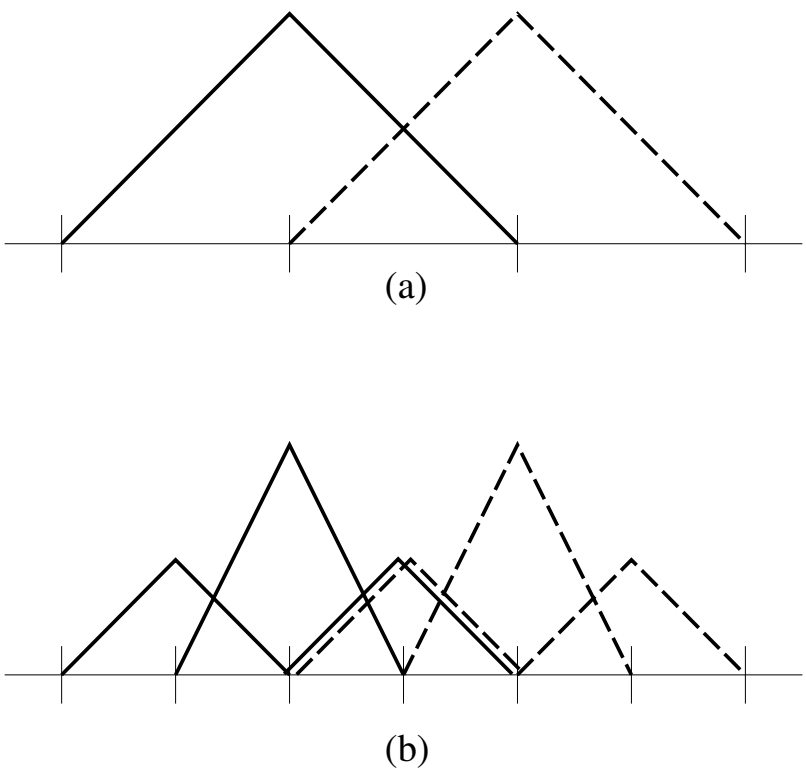


Figure 6: *Computing the inner product for the two piecewise linear curves shown in (a). After refinement, the problem reduces to computing the nine combinations of inner products between a solid function and a dashed function in (b).*

The next step is to refine with Equation 11:

$$x_2 = \langle \frac{1}{2}\phi_{i-1}^{j-1}(x) + \phi_i^{j-1}(x) + \frac{1}{2}\phi_{i+1}^{j-1}(x),\ \frac{1}{2}\phi_{i+1}^{j-1}(x) + \phi_{i+2}^{j-1}(x) + \frac{1}{2}\phi_{i+3}^{j-1}(x)\rangle.$$

Bilinearity of inner products allows this to be expanded to:

$$\begin{aligned} x_2 \quad = \quad & \tfrac{1}{4}\langle \phi_{i-1}^{j-1}(x), \phi_{i+1}^{j-1}(x)\rangle + \tfrac{1}{2}\langle \phi_{i-1}^{j-1}(x), \phi_{i+2}^{j-1}(x)\rangle + \tfrac{1}{4}\langle \phi_{i-1}^{j-1}(x), \phi_{i+3}^{j-1}(x)\rangle \\ & + \tfrac{1}{2}\langle \phi_{i}^{j-1}(x), \phi_{i+1}^{j-1}(x)\rangle + \langle \phi_{i}^{j-1}(x), \phi_{i+2}^{j-1}(x)\rangle + \tfrac{1}{2}\langle \phi_{i}^{j-1}(x), \phi_{i+3}^{j-1}(x)\rangle \\ & + \tfrac{1}{4}\langle \phi_{i+1}^{j-1}(x), \phi_{i+1}^{j-1}(x)\rangle + \tfrac{1}{2}\langle \phi_{i+1}^{j-1}(x), \phi_{i+2}^{j-1}(x)\rangle + \tfrac{1}{4}\langle \phi_{i+1}^{j-1}(x), \phi_{i+3}^{j-1}(x)\rangle. \end{aligned}$$

The result after refinement is shown in Figure 6(b). Through refinability, Equation 12 has been transformed to include each of the nine possible inner products involving a solid function and a dashed-line function. Most of these terms drop out, because the supports of their respective functions do not overlap:

$$x_2 \quad = \quad \tfrac{1}{2}\langle \phi_{i}^{j-1}(x), \phi_{i+1}^{j-1}(x)\rangle \ + \ \tfrac{1}{4}\langle \phi_{i+1}^{j-1}(x), \phi_{i+1}^{j-1}(x)\rangle \ + \ \tfrac{1}{2}\langle \phi_{i+1}^{j-1}(x), \phi_{i+2}^{j-1}(x)\rangle. \qquad (13)$$

The inner products in the first and last terms of Equation 13 are similar to case 2. However, they are parametrized over a narrower domain, and are therefore reduced by a scale of $\frac{1}{2}$ (recall that the scale for surfaces is $\frac{1}{4}$ instead). Likewise, the inner product of the middle term is a scaled-down version of case 1. We can therefore rewrite Equation 13 in terms of the unknowns x_1 and x_2:

$$x_2 = \frac{1}{8}x_1 + \frac{1}{2}x_2.$$

Similar analysis of case 1 yields:

$$x_1 = \frac{3}{4}x_1 + x_2.$$

After subtracting out the unknowns on the left, these equations set up a homogeneous system of the form

$$\begin{pmatrix} \frac{1}{8} & -\frac{1}{2} \\ -\frac{1}{4} & 1 \end{pmatrix}.$$

Arbitrarily setting $x_1 := 1$ again yields the inner product for case 2: $x_2 = \frac{1}{4}$. This agrees with the result obtained earlier through exact integration.

6 Multiresolution Analysis Based on Subdivision.

We have established nested linear spaces and an inner product relative to a uniformly convergent primal subdivision rule. We are now in a position to construct wavelets, that is, a set of functions $\Psi^j(\mathbf{x}) = (\psi_1^j(\mathbf{x}), \psi_2^j(\mathbf{x}), ...)$ that span the orthogonal complement space $W^j(M^0)$. (The elements of $\Psi^j(\mathbf{x})$ are not mutually orthogonal. Some authors, including Chui [4], refer to such functions as *pre-wavelets*.)

It is of significant practical importance that the decomposition and reconstruction filters associated with these wavelets are constructed and applied in linear time. This practical concern drives much of the development in this section.

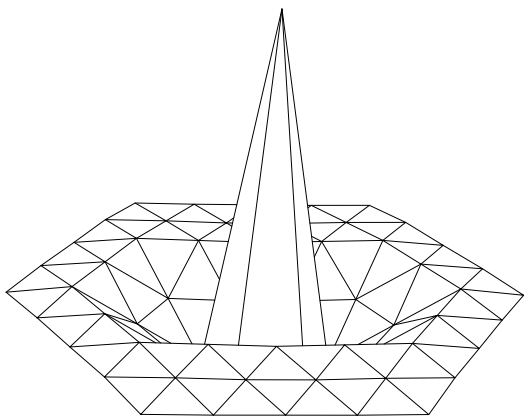


Figure 7: *A polyhedral wavelet centered on a vertex of valence 6.*

6.1 The Construction.

Our construction consists of two steps. First, we build a basis for $V^{j+1}(M^0)$ using the scaling functions $\Phi^j(\mathbf{x})$ and the new scaling functions $\mathcal{N}^{j+1}(\mathbf{x})$ in $V^{j+1}(M^0)$. It is straightforward to show that the $\Phi^j(\mathbf{x})$ and $\mathcal{N}^{j+1}(\mathbf{x})$ together span $V^{j+1}(M^0)$ if and only if the matrix $\mathbf{O}^j$ (encoding the subdivision rule around the old vertices) is invertible. Most primal subdivision methods, such as polyhedral subdivision and the butterfly method, have this property.[3] Given a function $S^{j+1}(\mathbf{x})$ in $\mathbf{V}^{j+1}(M^0)$ expressed as an expansion in the basis $(\Phi^j(\mathbf{x}) \;\; \mathcal{N}^{j+1}(\mathbf{x}))$, an approximation in $V^j(M^0)$ can be obtained by *restriction to* $\Phi^j(\mathbf{x})$, that is, by setting to zero the coefficients corresponding to $\mathcal{N}^{j+1}(\mathbf{x})$. However, this generally does not produce the best least-squares approximation.

To ensure the best least-squares approximation after restriction to $\Phi^j(\mathbf{x})$, we may orthogonalize the new basis functions $\mathcal{N}^{j+1}(\mathbf{x})$ by computing their projection into $W^j(M^0)$. The resulting functions $\Psi^j(\mathbf{x})$ are wavelets because they form a basis for $W^j(M^0)$. Expressed in matrix form:

$$\mathcal{N}^{j+1}(\mathbf{x}) = \Psi^j(\mathbf{x}) + \Phi^j(\mathbf{x})\,\boldsymbol{\alpha}^j. \tag{14}$$

Figure 7 is a plot of one such wavelet for the case of polyhedral subdivision. If $S^{j+1}(x)$ is expanded in terms of $\Phi^j(\mathbf{x})$ and $\Psi^j(\mathbf{x})$, then the restriction of $S^{j+1}(\mathbf{x})$ to $\Phi^j(\mathbf{x})$ is guaranteed to be the best approximation to $S^{j+1}(\mathbf{x})$ in $V^j(M^0)$ in a least-squares sense.

6.2 Computation of Wavelets.

The coefficients $\boldsymbol{\alpha}^j$ are the solution to the linear system formed by taking the inner product of each side of Equation 14 with $\Phi^j(\mathbf{x})$:

$$\begin{aligned} \langle \Phi^j(\mathbf{x}), \Phi^j(\mathbf{x}) \rangle\, \boldsymbol{\alpha}^j &= \langle \Phi^j(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle, \\ &= (\mathbf{P}^j)^T \; \langle \Phi^{j+1}(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle \end{aligned} \tag{15}$$

[3] One notable exception is Catmull-Clark subdivision for vertices of valence three. However, the subdivision rule for such vertices can be easily modified to produce an invertible matrix.

where $\langle \mathbf{F}, \mathbf{G} \rangle$ stands for the matrix whose i, i'-th entry is $\langle (\mathbf{F})_i, (\mathbf{G})_{i'} \rangle$. The matrix $\langle \Phi^j(\mathbf{x}), \Phi^j(\mathbf{x}) \rangle$ is therefore simply $\mathbf{I}^j$, and the matrix $\langle \Phi^{j+1}(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle$ is a submatrix of $\mathbf{I}^{j+1}$ consisting of those columns corresponding to members of $\mathcal{N}^{j+1}(\mathbf{x})$. The matrix $\boldsymbol{\alpha}^0$ for the example tetrahedron appears in Figure 2.

This system of equations may be solved for the coefficients $\boldsymbol{\alpha}^j$. Following Equation 14, the values $\boldsymbol{\alpha}^j$ are sufficient to construct the wavelets centered around the vertex for each scaling function in $\mathcal{N}^{j+1}(\mathbf{x})$.

6.3 Locally Supported Approximations to the Wavelets.

Although this construction produces wavelets orthogonal to the scaling functions at level j, there are some important practical difficulties that result. First, the inner product matrix $\mathbf{I}^j$ used to solve Equation 15 must be inverted. Second, the inverse of $\mathbf{I}^j$ is dense even though $\mathbf{I}^j$ is sparse. As a consequence, the resulting wavelets are globally supported on M^0, implying that filter bank decomposition and reconstruction algorithms using these wavelets require quadratic time. In order to apply these operations in linear time, we must build wavelets that are locally supported.

Constructing wavelets of local support is a common problem in the wavelet literature and has been handled in various ways. Chui [4] is able to develop unique locally supported wavelets for B-spline curves. Mallat [29] builds wavelets that, in addition to being orthogonal to the scaling functions, are mutually orthogonal. Unlike Chui, Mallat's conditions do not allow the construction of locally supported wavelets. Instead, in practice Mallat approximates his wavelets with locally supported truncations that are not strictly orthogonal to their scaling functions. Because the values in Mallat's wavelets decay exponentially away from the center, the deviation from orthogonality is bounded by choosing the size of the local support.

When truncation of this sort is used, decomposition and reconstruction are not inverses. This is a disadvantage, because it means that decomposition followed by reconstruction does not exactly reproduce the original surface. Nevertheless, Mallat has found the resulting nonorthogonal approximation perfectly adequate for practical use.

Like those of Mallat, the wavelets resulting from Equation 15 are orthogonal, and are globally supported over the entire domain mesh M^0, with values that appear to decay exponentially. We currently do not know of a construction leading to unique locally supported versions of subdivision wavelets, nor whether such a construction always exists. We will therefore obtain locally supported functions by relaxing the condition that the $\psi_i^j(\mathbf{x})$'s lie in $W^j(M^0)$. Instead, one may construct them to span a space V_*^j that is some (nonorthogonal) complement of $V^j(M^0)$ in $V^{j+1}(M^0)$. This choice is similar to the one in Dahmen *et al.* [6] in which orthogonality is sacrificed in order to achieve sparse filter banks. Our approach, which works well in practice, improves on Mallat's approach of merely truncating coefficients. Instead, we build functions in V_*^j that are the least-squares projections of $\psi_i^j(\mathbf{x})$ into W^j. This practice ensures that they are as close as possible to orthogonal, given their restricted support. Additionally, this construction ensures that the analysis and synthesis filters are inverses, which truncation alone cannot achieve. We will see below that

it is possible to make these approximations $V_*^j(M^0)$ arbitrarily close to $W^j(M^0)$, at the expense of increasing their support.

These approximations are constructed by selecting their supports *a priori*. For each $\psi_i^j(\mathbf{x})$, those members of $\Phi^j(\mathbf{x})$ whose supports are sufficiently distant from the support of $(\mathcal{N}^{j+1})_i$ have their corresponding coefficients in the i-th column of $\boldsymbol{\alpha}^j$ set to zero. The remaining nonzero coefficients can be found by solving a smaller, local variant of Equation 15. By allowing more of the coefficients of $\boldsymbol{\alpha}^j$ to be nonzero, the supports grow. The wavelets we have constructed have values which are observed to decay exponentially. Therefore, as the support grows, the local approximation $V_*^0(M^0)$ built for them quickly approaches $V^0(M^0)$. Additionally, observed results, such as those shown in Sections 7.5 and 8.1 empirically demonstrate the power of these approximations.

For actual implementation, we use these approximations instead of the globally supported wavelets described above. Using the approximations $V_*^j(M^0)$ has important consequences. Because they are locally supported, the resulting synthesis filters can be applied in linear instead of quadratic time, which allows a significant increase in the efficiency of algorithms that use them.

The drawback is that while truncation of a wavelet expansion of a complex function leads to a parametric least-squares best approximation of the function, truncation of an expansion using the nonorthogonal local wavelet approximations does not. In practice this does not seem to be much of a disadvantage for use with parametric surfaces because least-squares norms are themselves convenient approximations to more geometric norms such as Hausdorff or Frechet distance. Furthermore, any practical effects of nonorthogonality may always be made negligible by controlling the size of the local support.

6.4 A Filter Bank Algorithm.

Multiresolution analysis on the infinite real line is based on an assumption of spatial invariance — every place looks like every other place. This means that standard analysis and synthesis filters can be represented by a convolution kernel, that is, by a sequence of real numbers. This is not the case for multiresolution analysis on arbitrary topological domains. The filter coefficients in general must vary over the mesh, so the filters are represented by (hopefully sparse) matrices.

The analysis and synthesis filters can be conveniently expressed using block matrix equations. Let $\Psi^j(\mathbf{x})$ denote the row matrix of the locally supported wavelet approximations spanning $V_*^j(M^0)$. For any multiresolution analysis the synthesis filters are defined by the relation

$$\left(\Phi^j(\mathbf{x}) \;\; \Psi^j(\mathbf{x})\right) = \Phi^{j+1}(\mathbf{x}) \begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix}, \tag{16}$$

and the analysis filters are obtained from the inverse relation

$$\begin{pmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{pmatrix} = \begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix}^{-1}. \tag{17}$$

For the construction, it is again convenient to write $\Phi^{j+1}(x)$ in block form as

$$(\mathcal{O}^{j+1}(\mathbf{x})\, \mathcal{N}^{j+1}(\mathbf{x})).$$

It then follows from Equation 14 that the synthesis filters can be written in block form as

$$\begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix} = \begin{pmatrix} \mathbf{O}^j & -\mathbf{O}^j\,\boldsymbol{\alpha}^j \\ \mathbf{N}^j & \mathbf{1} - \mathbf{N}^j\,\boldsymbol{\alpha}^j \end{pmatrix}, \tag{18}$$

where $\mathbf{1}$ denotes the identity matrix. The analysis filters are obtained from Equation 17. (Examples for both are shown for the tetrahedron in Figure 2.)

From a practical standpoint, it is critical that the analysis and synthesis matrices are sparse. To achieve linear time decomposition and reconstruction, they must each have a constant number of nonzero entries in each row. If $\mathbf{P}^j$ and $\boldsymbol{\alpha}^j$ are sparse, then $\mathbf{Q}^j$ is sparse. Unfortunately, the analysis filters derived from Equation 17 need not be sparse. For interpolating subdivision schemes such as polyhedral subdivision and the G^1 "butterfly" scheme of Dyn *et al.* [14], the situation is much improved. Such interpolating schemes have the property that $\mathbf{O}^j$ is the identity matrix. Equation 18 in this case is greatly simplified; the resulting filters are

$$\begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix} = \begin{pmatrix} \mathbf{1} & -\boldsymbol{\alpha}^j \\ \mathbf{N}^j & \mathbf{1} - \mathbf{N}^j\,\boldsymbol{\alpha}^j \end{pmatrix} \qquad \begin{pmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{pmatrix} = \begin{pmatrix} \mathbf{1} - \boldsymbol{\alpha}^j\,\mathbf{N}^j & \boldsymbol{\alpha}^j \\ -\mathbf{N}^j & \mathbf{1} \end{pmatrix}.$$

If $\mathbf{P}^j$ and $\boldsymbol{\alpha}^j$ are sparse, then all four filters are also sparse, leading to linear time decomposition and reconstruction. The situation is less desirable for methods related to B-splines, such as Loop's scheme and Catmull-Clark surfaces. For these subdivision schemes, the synthesis filters are sparse, but the analysis filters are dense. This implies that decomposition is still possible in $O(n)$ time, but that the speed of reconstruction depends on the time to invert the sparse decomposition matrix of Equation 17, or to solve the related sparse linear system. Whether these methods can be made efficient for multiresolution analysis is a topic for future investigation. Table 1 shows what we currently know about the time required to develop reconstruction filters for various subdivision methods.

Table 1: *Time to construct synthesis filters for various subdivision methods.*

Method	Continuity	Asymptotic run time
Polyhedral	C^0	$O(n)$
Butterfly	G^1	$O(n)$
Loop	G^1	Speed of sparse linear equation solution
Catmull & Clark	G^1	Speed of sparse linear equation solution

The analysis filters can be used to decompose a surface $S^{j+1}(\mathbf{x})$ in $V^{j+1}(M^0)$ given by

$$S^{j+1}(\mathbf{x}) = \sum_i v_i^{j+1} \phi_i^{j+1}(\mathbf{x}) \tag{19}$$

into a lower resolution part in $V^j(M^0)$ plus a detail part in $V_*^j(M^0)$

$$S^{j+1}(\mathbf{x}) = \sum_i v_i^j \phi_i^j(\mathbf{x}) + \sum_i w_i^j \psi_i^j(\mathbf{x})$$

as follows. Let $\mathbf{V}^j$ be as in Equation 4, and let $\mathbf{W}^j$ denote the corresponding matrix of wavelet coefficients w_i^j. We can rewrite Equation 19 in matrix form and substitute the definition of the analysis filters. Thus:

$$\begin{aligned} S^{j+1}(\mathbf{x}) &= \Phi^{j+1}(\mathbf{x})\,\mathbf{V}^{j+1} \\ &= \left(\Phi^j(\mathbf{x})\,\Psi^j(\mathbf{x})\right)\begin{pmatrix}\mathbf{A}^j\\ \mathbf{B}^j\end{pmatrix}\mathbf{V}^{j+1} \\ &= \Phi^j(\mathbf{x})\mathbf{A}^j\,\mathbf{V}^{j+1} + \Psi^j(\mathbf{x})\mathbf{B}^j\,\mathbf{V}^{j+1} \end{aligned}$$

therefore,

$$\mathbf{V}^j = \mathbf{A}^j\,\mathbf{V}^{j+1} \tag{20}$$

$$\mathbf{W}^j = \mathbf{B}^j\,\mathbf{V}^{j+1}. \tag{21}$$

Of course, the analysis filters $\mathbf{A}^{j-1}$ and $\mathbf{B}^{j-1}$ can now be applied to $\mathbf{V}^j$ to yield $\mathbf{V}^{j-1}$, $\mathbf{W}^{j-1}$, etc. A similar argument shows that $\mathbf{V}^{j+1}$ can be recovered from $\mathbf{V}^j$ and $\mathbf{W}^j$ using the synthesis filters:

$$\mathbf{V}^{j+1} = \mathbf{P}^j\,\mathbf{V}^j + \mathbf{Q}^j\,\mathbf{W}^j. \tag{22}$$

The filter bank process is much simpler for the piecewise linear polyhedral functions. Details are given in Section 7.1.

6.5 Stability.

One important question concerns the stability of these synthesis and analysis filters. One measure of the stability of a transformation is its L^∞ norm. The L^∞ norm for any matrix T is measured by $||T||_\infty$, which gives the maximum sum of the absolute values of the elements in any row of T, and indicates the maximum scaling effect of T on any vector that it transforms.

For truly orthogonal wavelet bases over regular grids, the L^∞ norm is bounded by a constant, independent of the number of filtering steps. In this section, we briefly analyze the L^∞ norms of the analysis and synthesis filters described in the previous section, based on local approximations to the globally supported wavelets.

We first consider repeated application of the synthesis filter. This transformation maps the coarse-level scaling coefficients $\mathbf{V}^0$ and the wavelet coefficients $\mathbf{W}^0$, ..., $\mathbf{W}^{n-1}$ into the fine-level scaling coefficients $\mathbf{V}^n$. Expanding equation 22 repeatedly yields

$$\mathbf{V}^n = \prod_{i=0}^{n-1}\mathbf{P}^i\mathbf{V}_0 + \sum_{j=0}^{n-1}(\prod_{i=j}^{n-1}\mathbf{P}^i)\mathbf{Q}_j\mathbf{W}_j.$$

The L^∞ norm of this transformation depends on the L^∞ norm of the product $\prod_{i=0}^{j}\mathbf{P}^j$. We claim that this norm is bounded independent of n. By definition, convergence of the subdivision scheme defined by the $\mathbf{P}^j$ implies that the entries of $\prod_{i=0}^{j}\mathbf{P}^j$ are also bounded independent of n. Since only there are only a finite number of nonzero entries per row of

$\prod_{i=0}^{j} \mathbf{P}^j$, the L^∞ norm of $\prod_{i=0}^{j} \mathbf{P}^j$ is bounded independent of n. To conclude, we note that $\mathbf{V}^n$ is the sum of $n+1$ transformations with bounded L^∞ norms. Therefore, the norm for the entire transformation is at most $O(n)$. As a common special case, when the subdivision rule encoded by $\mathbf{P}^j$ is a convex combination (such as with piecewise linear subdivision or with the Catmull-Clark scheme), the L^∞ norm of any $\mathbf{P}^j$ is exactly 1, implying that the L^∞ of their product is, too.

We next consider repeated application of the analysis filter. This transformation maps the fine scaling coefficients $\mathbf{V}^n$ into the coarse scaling coefficients $\mathbf{V}^0$ and the wavelet coefficients $\mathbf{W}^0$, ..., $\mathbf{W}^{n-1}$. Expanding equations 20 and 21 yields

$$\begin{aligned} \mathbf{V}^0 &= \prod_{i=0}^{n-1} \mathbf{A}^i \mathbf{V}^n. \\ \mathbf{W}^j &= \mathbf{B}^j \prod_{i=j+1}^{n-1} \mathbf{A}^i \mathbf{V}^n. \end{aligned}$$

The stability of these transformations depends on the L^∞ norm of $\prod_{i=0}^{n-1} \mathbf{A}^i$. Note that the product of these matrices is the projection operator that maps $\mathbf{V}^n$ into $\mathbf{V}^0$. We conjecture that the norm for this transformation is bounded independent of n.

(To support this conjecture, we briefly sketch a possible line of proof when M^0 is an unbounded, regular grid. We note that the same projection transformation $\mathbf{A}$ maps $\mathbf{V}^{j+1}$ to $\mathbf{V}^j$ for all $j > 0$. We wish to show that $(\mathbf{A})^n$ has bounded L^∞ norm independent of n. Choose $\boldsymbol{\alpha}$ in equation 14 so that columns of $\mathbf{A}$ sum to $\frac{1}{4}$. We conjecture that $4 * \mathbf{A}^T$ is a convergent subdivision operator. If so, then the maximum entry in $(\mathbf{A})^n$ is $O(4^{-n})$ and maximum number of nonzero entries in row of $(\mathbf{A})^n$ is $O(4^n)$. Therefore, $(\mathbf{A})^n$ has bounded norm independent of n.)

Table 2: *L^∞ norms for the analysis filters $\mathbf{A}^j$ that arise from piecewise linear subdivision on the octahedron.*

	Size 1	Size 2	Size 3
$\|\|\mathbf{A}^0\|\|_\infty$	1.25	1.3125	1.3125
$\|\|\mathbf{A}^1\|\|_\infty$	1.27174	1.40771	1.4434
$\|\|\mathbf{A}^2\|\|_\infty$	1.27174	1.40859	1.45371
$\|\|\mathbf{A}^3\|\|_\infty$	1.27174	1.40859	1.45372
$\|\|\mathbf{A}^4\|\|_\infty$	1.27174	1.40859	1.45372

In practice, the norms for these filters appear to be very small, implying good stability. Table 2 gives the L^∞ norms for the analysis filter $\mathbf{A}^j$ over a varying number of levels using a varying size support. These $\mathbf{A}^j$ are built using the wavelet approximations described above, for the case of piecewise linear subdivision over the octahedron. In the table, *size* measures the distance from the finer-level wavelet vertex of the coarser-level support functions. (All examples in this paper were generated for a size of 2.)

Table 3: L^∞ *norms for successive products of the* $\mathbf{A}^j$*, beginning at* $\mathbf{A}^4$*.*

Matrix product	Size 1	Size 2	Size 3
$\mathbf{A}^4$	1.27174	1.40859	1.45372
$\mathbf{A}^3 * \mathbf{A}^4$	1.61574	1.95076	2.04751
$\mathbf{A}^2 * \mathbf{A}^3 * \mathbf{A}^4$	2.02336	2.50179	2.60807
$\mathbf{A}^1 * \mathbf{A}^2 * \mathbf{A}^3 * \mathbf{A}^4$	2.35466	2.78397	2.86811
$\mathbf{A}^0 * \mathbf{A}^1 * \mathbf{A}^2 * \mathbf{A}^3 * \mathbf{A}^4$	2.22687	2.63381	2.75473

Table 3 gives the cumulative effect of the analysis filters. It gives the L^∞ norms for each stage of the composition of the $\mathbf{A}^j$, beginning with $\mathbf{A}^4$, down to $\mathbf{A}^0$.

The L^∞ norm for the synthesis filters $\mathbf{P}^j$ arising from piecewise linear subdivision is simply 1 for every level, and is therefore not shown in a table.

Dahmen [7] also gives a similar, but more detailed analysis of the stability of multiresolution schemes over irregular grids.

7 Wavelet Compression of Surfaces.

Multiresolution analysis is widely used for data compression applications. Mallat [29] and DeVore *et al.* [11], among others, use wavelet techniques for efficient image compression. Finkelstein and Salesin [16] develop a wavelet-based method for approximating B-spline curves within an L^∞ tolerance. Other examples of wavelet-based compression techniques abound [5, 28, 30, 2].

Using wavelet techniques, lossy compression of a function is typically implemented with a three-stage algorithm:

1. *Filter bank decomposition.* The original function, represented by the sequence v^j, is decomposed into a coarse-level approximation v^0 together with wavelet coefficients $w^0, \ldots, w^{j-1}$ at the levels from 0 to $j-1$.

2. *Selection.* A subset d of the detail is chosen from each sequence $w^0, \ldots, w^{j-1}$, according to some measure of its importance.

3. *Reconstruction.* The selected detail d, in the form of scaled wavelets at various levels, is added back to the coarse-level approximation.

Throughout the rest of this section, we will discuss the application of subdivision wavelets to the compression of complex surface data. The common special case of polyhedral decomposition is treated in Section 7.1. Next, Section 7.2 discusses various rules for selecting wavelet coefficients. Section 7.4 discusses the preprocessing step sometimes necessary for converting general input into a form appropriate for compression. Finally, Section 7.5 illustrates the results of compression for two complex polyhedral data sets.

In this section, we exclusively consider the approximation of functions defined over triangular meshes. Quadrilateral methods require separate algorithms which are nevertheless similar in flavor.

7.1 Polyhedral Implementation.

Many applications of subdivision wavelets involve the special case of piecewise linear subdivision — an example is polyhedral compression. The techniques outlined in Section 4 apply to polyhedra, but it is possible to exploit the simplicity of piecewise linear subdivision to derive an even more efficient implementation. Moreover, decomposition and reconstruction may be achieved in linear time for polyhedral surfaces without the need for a sparse matrix representation. (However, it is still necessary to solve a small linear system for the local neighborhood of wavelet coefficients.)

The chief property of piecewise linear subdivision that leads to simpler algorithms is tight locality. Unlike more complex subdivision rules, the support of a hat function built around a vertex $\mathbf{v}$ does not extend beyond $\mathbf{v}$'s neighboring vertices. In this section, we show that this leads to a simpler technique for generating the filters $\mathbf{A}$ and $\mathbf{B}$.

As is discussed in Section 6.4, the wavelet coefficients $\mathbf{W}^j$ may be found through the matrix product $\mathbf{B}^j\mathbf{V}^{j+1}$. Construction of $\mathbf{B}^j$ is greatly simplified for interpolating schemes, including polyhedral subdivision, and for polyhedra, the wavelet coefficient $\mathbf{w}$ associated with a vertex $\mathbf{v}$ (where $\mathbf{v}$ is fine-level vertex added on the edge $\mathbf{pq}$) may be derived by the simple rule:

$$\mathbf{w} = \mathbf{v} - \frac{1}{2}(\mathbf{p} + \mathbf{q}). \tag{23}$$

This equation is shown geometrically in Figure 8. It leads to a very simple algorithm for computing the wavelet coefficients — without the need for implementing sparse matrix multiplication.

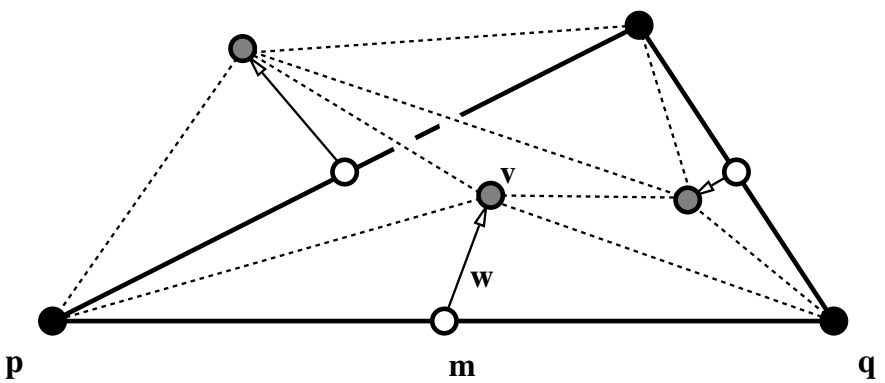


Figure 8: *Determining the wavelet coefficient* $\mathbf{w}$ *around vertex* $\mathbf{v}$*, with parents* $\mathbf{p}$ *and* $\mathbf{q}$*. The midpoint between* $\mathbf{p}$ *and* $\mathbf{q}$ *is* $\mathbf{m}$*, and the resulting wavelet coefficient is the difference* $\mathbf{w}$ *between* $\mathbf{m}$ *and* $\mathbf{v}$*.*

Determining all level $j-1$ wavelet coefficients w^{j-1} using Equation 23 is the first half

of the level j to level $j-1$ decomposition process. It is still necessary to derive the filter $\mathbf{A}$ that gives the scaling functions for the approximation at level $j-1$. This is not as easy, but one may still take advantage of the simplicity of the piecewise linear representation in order to achieve a procedure that is more efficient than the general case.

Once the wavelet coefficients w_i^{j-1} are derived, the coarser-level approximation at level $j-1$ may be determined by subtracting from the level j function the effect of every scaled wavelet term $w_i^{j-1}\psi_i^{j-1}$. The problem then reduces to determining the wavelets ψ_i^{j-1}.

The ψ_i^{j-1} may be computed using the techniques of Section 4, but specially adapted for linear subdivision. In particular, the inner product mask for the linear case is shown in Figure 4. This mask is especially easy to determine for polyhedra, because the central value is simply the valence of the vertex.

7.2 Coefficient Selection.

Once the surface has been decomposed via the filter bank, the next step in surface compression is to select appropriate coefficients to add back to the base mesh.

Before examining the wavelet coefficients for selection, it is useful to first *normalize* them. Normalization allows coefficients at differing subdivision levels to be equitably compared, despite the different domain sizes over which they are defined.

Least-squares (L^2) normalization of the wavelet coefficients assigns a weight to a wavelet coefficient that indicates how much least-squares error occurs when the coefficient is left out of the reconstruction. A wavelet ψ_i^j can be normalized to a "unit length" wavelet $\widehat{\psi_i^j}$ with the following equation:

$$\widehat{\psi_i^j} := \frac{\psi_i^j}{\sqrt{\langle \psi_i^j, \psi_i^j \rangle}}.$$

When the wavelet ψ_i^j has an associated wavelet coefficient w_i^j, the L^2 normalized coefficient $\widehat{w_i^j}$ is therefore assigned the value

$$\widehat{w_i^j} := w_i^j \sqrt{\langle \psi_i^j, \psi_i^j \rangle}.$$

All examples presented in this paper use selection based upon this L^2 normalization of the wavelet coefficients.

There are several possible techniques for selecting coefficients. These include:

- *Threshold testing.* Perhaps the simplest selection technique is to simply choose all wavelet coefficients whose magnitude is greater than some coefficient ϵ. Although thresholding is quite simple, the results are of remarkable quality, as we will see in Section 7.5. In the reconstruction, areas of high curvature are sampled more densely than relatively flat regions.

- L^2 *progressive refinement.* In certain applications, it is useful to ensure that the most important information is reconstructed first. When the wavelets are all mutually orthogonal, and when the L^2 normalization described above is used, it can be shown

that simply supplying the c largest coefficients in decreasing order is sufficient to produce a sequence of approximations, each of which is the best possible least-squares approximation using only c coefficients. The wavelet coefficients must first be sorted by their magnitude, which implies an $O(n \log n)$ run time in the input size.

The wavelets that we construct are not mutually orthogonal, so the property above cannot be shown for them. However, sorting coefficients and selecting them in largest to smallest order still produces good results.

- *Maximum error: L^∞ reconstruction.* An approximation constructed according to the L^2 norm may still contain an arbitrarily large error in a sufficiently small region. An alternative approach uses the L^∞ norm to guarantee that no part of the reconstruction is farther than a user-defined tolerance from its corresponding point on the original input.

  L^∞ reconstruction begins with the complete, unreduced reconstruction, and attempts to remove successive wavelet coefficients until no more are possible. L^∞ wavelet compression of complex surfaces is further studied in Eck *et al.* [15].

- *Location.* In some applications, it may be useful to compress only a portion of a model while the rest of it is outside the user's view. For example, a user may be interested in viewing a specific location on a map of the Earth, but uninterested in viewing geography elsewhere. In this case, the location information associated with a wavelet makes it possible to apply wavelet compression to the region of interest, but to avoid processing irrelevant regions of the model.

  Selection by location is usually applied in conjunction with another selection method. Only those coefficients in the relevant region are candidates for the secondary selection technique.

7.3 Reconstruction Algorithms.

To review, the result of decomposition is the base mesh and the coefficients of the wavelets at various levels of subdivision. The information at the vertices of the base mesh stores values of a very few scaling functions, providing a very coarse least-squares approximation of the input function. The wavelets are functions that represent the missing detail at each finer-level vertex. In order to build a wavelet approximation of the original input, we can add back to the base mesh a selected subset of these finer-level functions.

Reconstruction of the approximation is done by adding back to the base surface the scaled wavelet associated with each selected coefficient. Details of the reconstruction phase are given explicitly in Lounsbery [27], and are discussed by Eck *et al.* [15] for the case of polyhedra.

The effect of reconstruction is to produce vertices on the approximation surface that represent the addition of the selected wavelets to the base surface. However, this form is not appropriate for many applications that require an actual surface representation. If the approximation is to be rendered as a polyhedral surface, it is necessary to generate polygons

over the approximated surface. A separate top-down recursive triangulation algorithm is required in order to connect the points on the reconstruction into polygonal form.

When c is the number of selected coefficients and j is the subdivision depth of the input function, the asymptotic time for reconstruction and triangulation is at worst $O(j\ c)$, but in practice usually much closer to $O(c)$. These values are borne out by the empirical evidence of run times for reconstruction.

7.4 Subdivision Connectivity.

The compression techniques detailed in this section are useful for applications requiring decomposition of a function in $V^j(M^0)$, where M^0 may be a mesh of any topological type. An implicit assumption is that the connectivity of the input mesh must have the form of a mesh M^j that results from subdividing a simple mesh M^0 j times. We call this property *subdivision connectivity.*

For many applications, such as global illumination or surface editing, producing input with subdivision connectivity is a simple matter. For other purposes, including multiresolution compression of an arbitrary mesh, an initial preprocessing step is required to convert the input into an approximation with the necessary connectivity. Such a conversion algorithm is presented in Eck *et al.* [15] for the general case. The examples presented in this paper were converted with an algorithm written for the special case of converting input taken in cylindrical sections into points on a subdivided octahedron.

7.5 Polyhedral Examples.

In this section, subdivision wavelets are applied to two compression problems: the compression of a polyhedral model consisting of over 32,000 triangles, and compression of a piecewise linear representation of a color function defined on over one million points on the globe. The data for these examples were resampled from cylindrical sections onto a subdivided octahedron.

7.5.1 Geometric Data.

The input for the first example (shown in Figure 9(a)) is a polyhedral mesh consisting of 32,768 triangles whose vertices were resampled from laser range data originally provided through the courtesy of Cyberware, Inc. The triangulation was created by recursively subdividing an octahedron six times. The octahedron therefore serves as the domain mesh M^0, with the input triangulation considered as a parametric function $S(\mathbf{x}), \mathbf{x} \in M^0$ lying in $V^6(M^0)$. More precisely, if v_i^6 denotes the vertices of the input mesh, $S(\mathbf{x})$ can be written as

$$S(\mathbf{x}) = \Phi^6(\mathbf{x})\mathbf{V}^6, \qquad \mathbf{x} \in M^0$$

where the scaling functions $\Phi^6(\mathbf{x})$ are the (piecewise linear) functions defined through polyhedral subdivision.

The locally supported wavelet approximations $\psi_i^j(\mathbf{x})$ for this example are chosen to be supported on 2-discs. (The k-disc around a vertex v of a triangulation is defined to be the

set of all triangles whose vertices are reachable from v by following k or fewer edges of the triangulation.) The filter bank process outlined in Section 6.4 can be applied in linear time to rewrite $S(\mathbf{x})$ in the form

$$S(\mathbf{x}) = \Phi^0(\mathbf{x})\ \mathbf{V}^0 + \sum_{j=0}^{5} \Psi^j(\mathbf{x})\ \mathbf{W}^j.$$

The first term describes a base shape as the projection of $S(\mathbf{x})$ into $V^0(M^0)$, which in this case is an approximating octahedron with vertex positions given by the six rows of $\mathbf{V}^0$. For this data, the decomposition stage of the filter bank runs in about 14 seconds on an SGI Indigo^2 Extreme.

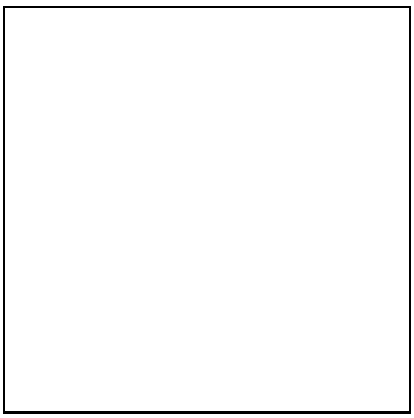
Approximations to the original mesh $S(\mathbf{x})$ can be easily obtained from the wavelet expansion using coefficients selected according to the threshold rule (see Section 7.2). The models in Figure 10(b), (d), and (f) are compressed to 1%, 13%, and 32%, respectively. Notice that thresholding causes the mesh to refine in areas of high detail, while leaving large triangles in areas of relatively low detail. An implementation of the entire decomposition and reconstruction process that produces Figure 10(d) runs in about 53 seconds from input to output.

Figure 10 also illustrates the use of wavelet approximations for automatic level-of-detail control in rendering. The original Spock polyhedron is shown in full resolution in Figure 9(a). Views of the full-resolution model from various distances are shown in the rest of Figure 9. When viewing the input polyhedron at these distances, it is inefficient and unnecessary to render all 32,000 triangles. The approximations shown in the left column of Figure 10 may instead be used without significantly degrading image quality.

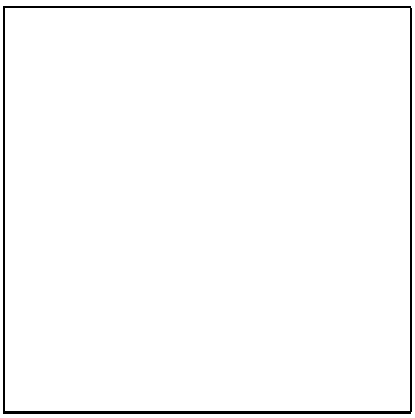
Suddenly switching between models of different detail in an animation often produces a noticeable jump. This problem is easily mended by using a wavelet expansion where the wavelet coefficients are treated as continuous functions of the viewing distance. This simple technique allows the object geometry to smoothly change its appearance as the viewing distance changes. This method has proved successful shown in a frame-by-frame animation we have produced.

7.5.2 Color Data.

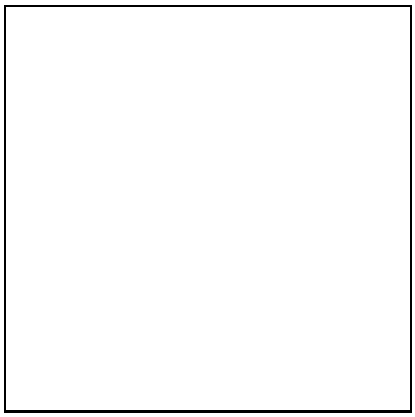
Subdivision wavelets may be applied to more general functions over surfaces than geometric data. Figure 12 demonstrates another application — that of compressing a function on the sphere. In this example, elevation and bathymetry data obtained from the U.S. National Geophysical Data Center was used to create a piecewise linear pseudocoloring of the sphere. The resulting color function was represented by 2,097,152 triangles and 1,048,578 vertices. The full-resolution pseudocoloring was too large to be rendered on an SGI Indigo^2 Extreme with 128 MB, and is therefore not shown in its entirety in Figure 12. Instead, Figure 11 shows a close-up of a region that is compressed in Figure 12. An appreciation for the density of the data can be obtained from Figure 11(b), where even at close range the mesh lines of the original uncompressed data are so dense that the image appears almost completely black.



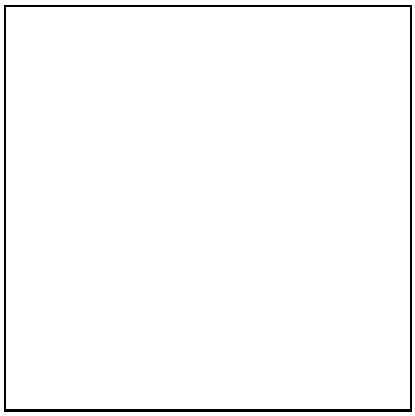
(a) The original Spock mesh.



(b) Distant view.

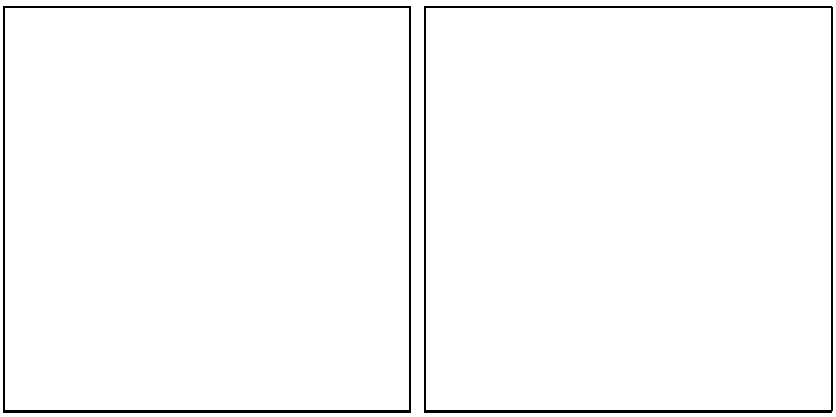


(c) Mid-range view.



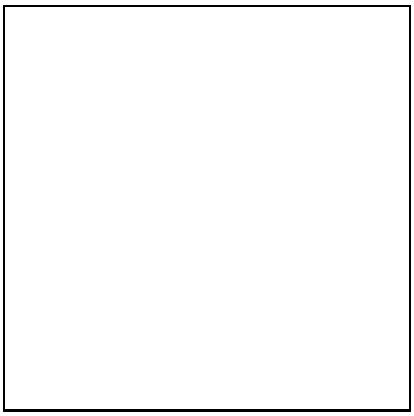
(d) Close-up view.

Figure 9: *The full-resolution Spock polyhedron. Left: the full mesh (16,386 points, 32,768 triangles). Right: Gouraud-shaded views of the full mesh at various distances.*

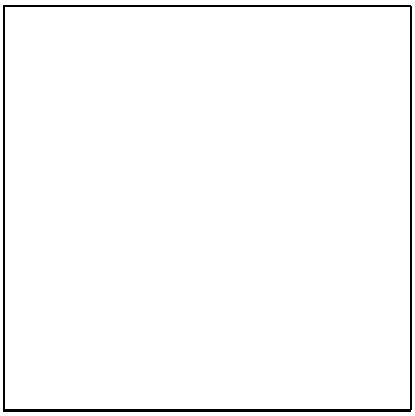


(a) Distant view of mesh in (b).

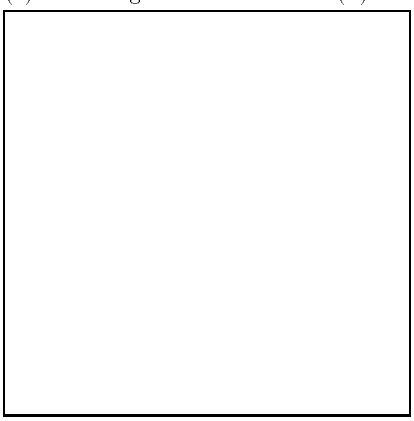
(b) 105 wavelets, 240 triangles.



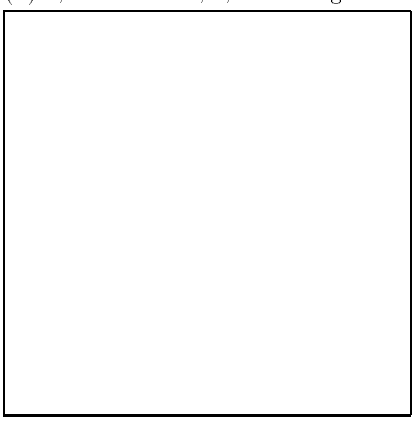
(c) Mid-range view of mesh in (d).



(d) 1,966 wavelets, 4,272 triangles.

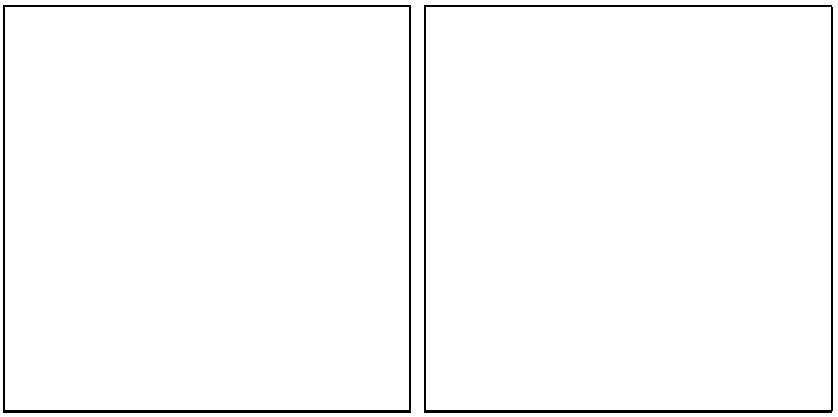


(e) Close-up view of mesh in (f).



(f) 5,054 wavelets, 10,704 triangles.

Figure 10: *Wavelet approximations of the Spock polyhedron. Left: Gouraud-shaded wavelet approximations. Right: Flat-shaded close-ups showing structure of approximations.*



(a) Full-resolution model. (b) Mesh for (a).

Figure 11: *Close-up of Earth data before compression*

The approximations shown in Figure 12(a)-(f) were produced using subdivision wavelet compression. Figure 12(a) shows a distant view of the Earth using an approximation of only 0.1% [the mesh is shown in (b)]. Likewise, Figures 12(c) and (d) show the result of compression to 2% for a medium-range view. At close range the compression to 16% in (e) is nearly indistinguishable from the full-resolution model in Figure 11(a). A comparison of the meshes shown in Figure 11(b) and Figure 12(f) reveals the striking degree of compression achieved in this case.

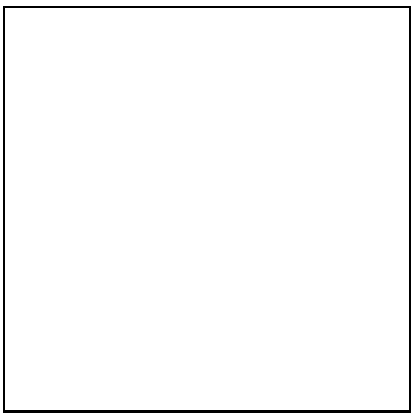
8 Smooth Surface Modeling.

This section describes how subdivision wavelets can be used to compress and edit smooth subdivision surfaces. Although our examples concentrate on the modeling of smooth surfaces based on the Dyn *et al.* butterfly scheme, the techniques described in this section may be applied to general subdivision surfaces.

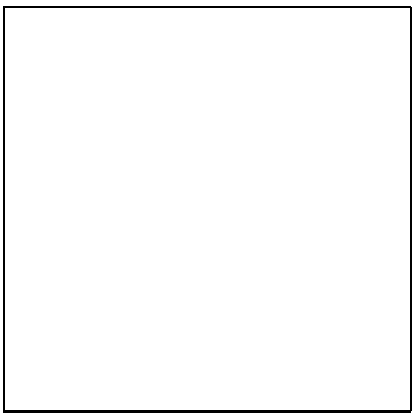
In Section 7, we saw examples of polyhedral compression using wavelets. In this section, we examine compression of surfaces defined by smooth subdivision rules.

8.1 Smooth Surface Compression.

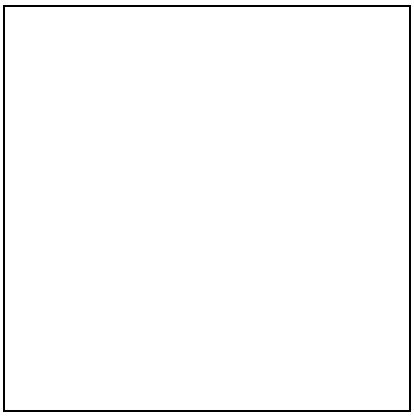
The result of smooth surface compression is shown in Figure 13. The surface in Figure 13(a) shows the level 6 Spock data set from Section 7.5 after it has been subdivided using the tangent-plane smooth butterfly subdivision rule. Figure 13(b) shows the result of compression using wavelets that were developed from the butterfly scheme. The surface of (a) has been compressed to a representation [depicted in (b)] that may be stored using only 16% of the original coefficients.



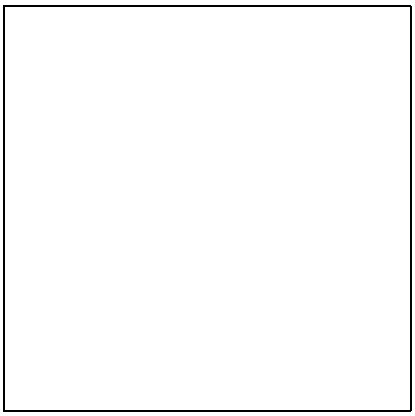
(a) Distant view of (b).



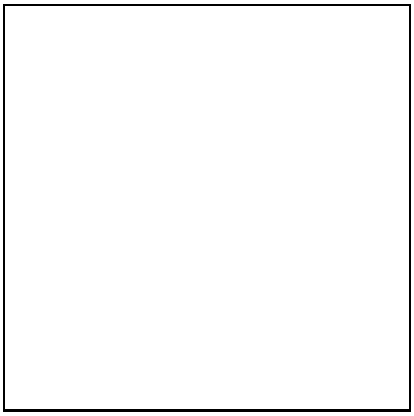
(b) Compression to 0.1%.



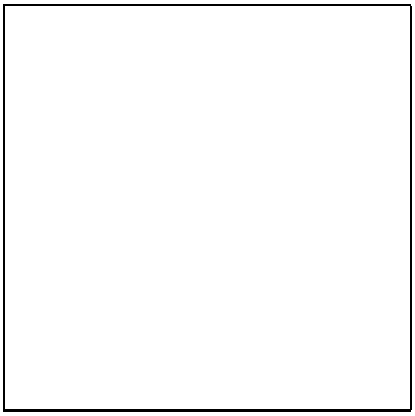
(c) Mid-range view of (d).



(d) Compression to 2%.

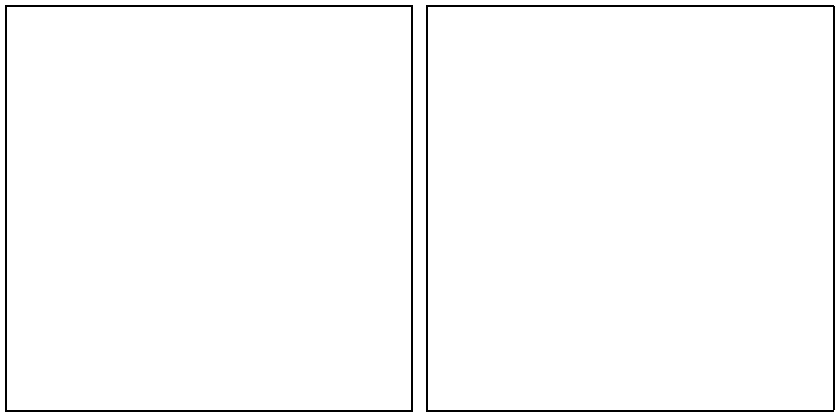


(e) Close-up view of (f).



(f) Compression to 16%.

Figure 12: *Approximating color as a function over the sphere.*



(a) The original smooth surface. (b) Its wavelet compression to 16%.

(c) A coarse-level change of (a). (d) A finer-level modification.

Figure 13: *Compression and multiresolution editing of a smooth surface.*

8.2 Multiresolution Editing.

Subdivision wavelets may also be used for editing shapes at multiple resolutions. Although we have not fully implemented multiresolution editing, Figures 13(c)–(d) show an example of editing the smooth shape seen in Figure 13(a).

Figure 13(c) shows the effect of changing a single scaling function coefficient on the level 0 base octahedron. Because finer-level vertices in the same region are defined relative to the coarser shape, they move along with the modification. However, the geometry in areas away from the front of the bust is not affected.

It is also possible to locally modify the shape at a finer level by changing the value of a wavelet coefficient at that level. The result of modifying a single level 3 wavelet coefficient is shown in Figure 13(d).

The changes shown in Figure 13(c)–(d) were created by simply modifying a single value in the wavelet representation. It is preferable to make these changes under a more powerful user interface. Finkelstein and Salesin [16] discuss interfaces that are more appropriate for wavelet editing of curves and surfaces.

9 Summary and Future Work.

In this paper, we have established a theoretical basis for applying multiresolution analysis to surfaces of arbitrary topological type. These results hold for any local, uniformly convergent, continuous, primal subdivision scheme, including polyhedral subdivision, the butterfly scheme [14], Loop's schem [26], and Catmull-Clark surfaces [3]. The results also hold for piecewise smooth subdivision as described in Hoppe *et al.* [23, 22], and for open surfaces possessing holes or boundary curves.

There are numerous areas for future work:

- *Weighted Inner Product.* Because the inner products of Section 5 are defined independent of the actual geometric size of the triangles, the least-squares accuracy of the resulting geometric approximations depends on how evenly spaced the input triangles are over the geometry of the input surface. Changing the inner product to depend on the true geometric size of these triangles has far-reaching implications for the resulting wavelet construction, and would appear to seriously hamper an efficient implementation.

  Another possibility is to modify the preprocessing phase in order to approximate the input with triangles that do not significantly vary in size. Such remeshing creates a close match between the inner product weights and the actual sizes of the triangles. This is the approach taken by Eck *et al.* [15].

- *Linear Time Sparse Matrix Inversion.* As explained in Section 6.4, linear-time reconstruction for general subdivision rules depends upon linear-time solution of general sparse linear systems. Hence, finding an $O(n)$ sparse linear system solver is of value for efficiently implementing subdivision wavelets for more general subdivision schemes, such as the G^1 methods of Loop and Catmull-Clark.

- *Locally Supported Orthogonal Wavelets.* The approximations developed in Section 6 are projections of the orthogonal wavelets into a local support space of arbitrary size. While the local support allows the resulting filters to be built in linear time, the functions are not truly orthogonal. Although this method has been satisfactory in practice, it is theoretically unsatisfying. An improvement would be to discover truly orthogonal functions with linear time filter construction, or to show that such functions cannot exist.

  Note, however, that even if locally supported wavelets can be shown to exist, their analysis filters will still be global. Thus, an analogue of the Quak-Weyrich trick for linear time decomposition [34] would also need to be developed for a linear time filter bank.

- *Preventing Self-Intersection.* Because reconstructed approximations may self-intersect, the basic compression technique does not guarantee preservation of the original topology in the ambient space. While this point is unimportant for such applications as animation previewing, or for simplifying surfaces rendered by radiosity, it may be important enough for other applications to warrant further study.

- *Simplifying the Topological Type.* The surface decomposition developed herein retains the topological type of the input surface. When the input is a relatively simple object with many small holes, it is more often desirable to decompose the input into a "topologically simpler" surface; that is, one with lower genus or fewer boundary curves. Getting this method to work under the current formulation would appear to require a form of subdivision that allows rules for changing the underlying surface topology.

10 Acknowledgments

We thank Tom Duchamp and Jean Schweitzer for productive discussion.

This work was made possible in part by Xerox Corp. and the National Science Foundation under grant no. CCR-8957323.

References

[1] Robert G. Bartle. *The Elements of Real Analysis.* John Wiley and Sons, 1964.

[2] Deborah Berman, Jason Bartell, and David Salesin. Multiresolution painting and compositing. *Computer Graphics Annual Conference Series*, pages 85–90, July 1994.

[3] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[4] Charles K. Chui. *Wavelet Analysis and its Applications*, volume 1. Academic Press, Inc., Boston, 1992.

[5] Charles K. Chui and Xianliang Shi. Wavelets and multiscale interpolation. In Tom Lyche and Larry L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II*. Academic Press, 1992.

[6] W. Dahmen, B. Kleeman, S. Prössdorf, and R. Schneider. A multiscale method for the double layer potential equation on a polyhedron. Technical Report 91, Institut für Geometrie und Practische Mathematik, Rheinisch-Westfälische Technische Hochshule Aachen, November 1993.

[7] Wolfgang Dahmen. Stability of multiscale transformations. To appear.

[8] Wolfgang Dahmen and Charles A. Micchelli. Using the refinement equation for evaluating integrals of wavelets. *SIAM Journal on Numerical Analysis*, 30(2):507–537, April 1993.

[9] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, Philadelphia, 1992.

[10] Tony DeRose, Tom Duchamp, and Jean Schweitzer. A convergence proof for piecewise smooth subdivision surfaces. In preparation, 1994.

[11] R. DeVore, B. Jawerth, and B. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746, March 1992.

[12] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[13] D.W.H. Doo. *A Recursive Subdivision Algorithm for Fitting Quadratic Surfaces to Irregular Polyhedrons*. PhD thesis, Brunel University, 1978.

[14] Nira Dyn, David Levin, and John Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.

[15] Matthias Eck, Tony DeRose, Tom Duchamp, Werner Stuetzle, Hugues Hoppe, and Michael Lounsbery. Multiresolution approximations of arbitrary polyhedra. submitted for publication.

[16] Adam Finkelstein and David Salesin. Multiresolution curves. *Computer Graphics Annual Conference Series*, pages 261–268, July 1994.

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

[18] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. Wavelength dependent reflectance functions. *Computer Graphics Annual Conference Series*, pages 213–220, July 1994.

[19] Steven Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics Annual Conference Series*, pages 221–230, August 1993.

[20] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Computer Graphics Annual Conference Series*, pages 35–44, August 1993.

[21] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, July 1991.

[22] Hugues Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, University of Washington, Seattle, Washington, June 1994. TR 94-06-01.

[23] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics Annual Conference Series*, pages 295–302, July 1994.

[24] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer Graphics Annual Conference Series*, pages 19–26, August 1993.

[25] Rong-Qing Jia and Charles A. Micchelli. Using the refinement equations for the construction of pre-wavelets II: Powers of two. In P. J. Laurent, A. LeMéhauté, and L. L. Schumaker, editors, *Curves and Surfaces*, pages 209–246. Academic Press, 1991.

[26] Charles T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[27] J. M. Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington, Seattle, WA, September 1994.

[28] Bradley J. Lucier. Wavelets and image compression. In Tom Lyche and Larry L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II*. Academic Press, 1992.

[29] 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.

[30] Stephane Mallat and Wen Liang Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, March 1992.

[31] David Meyers. Multiresolution tiling. In *Proceedings of Graphics Interface*, pages 25–32, May 1994.

[32] David Meyers. *Reconstruction of Surfaces from Planar Contours*. PhD thesis, University of Washington, Seattle, Washington, July 1994.

[33] A. P. Pentland. Fast solutions to physical equilibrium and interpolation problems. *Visual Computer*, 8(5-6):303–314, June 1992.

[34] Edwald Quak and Norman Weyrich. Decomposition and reconstruction algorithms for spline wavelets on a bounded interval. Technical Report 294, Department of Mathematics, Texas A&M University, April 1993.

[35] William J. Schroeder, Jonathan A. Zarge, and William E. Lorenson. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[36] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics Annual Conference Series*, pages 435–442, July 1994.

[37] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, 1992.