

© Copyright 1994
John Michael Lounsbery

Multiresolution Analysis for Surfaces of Arbitrary Topological Type

by

John Michael Lounsbery

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

1994

Approved by _____

(Chairperson of Supervisory Committee)

Program Authorized
to Offer Degree _____

Date _____

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI, 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Multiresolution Analysis for Surfaces of
Arbitrary Topological Type

by John Michael Lounsbery

Chairperson of Supervisory Committee: Professor Anthony D. DeRose
Department of
Computer Science and Engineering

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, hierarchical optimization, and numerical analysis.

In this dissertation, I 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.

The wavelets developed in this dissertation depend upon a generalization of the standard notion of multiresolution analysis. Rather than developing wavelets from scales and translates over regular domains, they are based on refinable scaling functions. Because these wavelets are built over subdivision surfaces, they may be used to model piecewise linear functions (such as polyhedral surfaces), tangent-plane smooth (C^1) functions, or piecewise smooth functions with discontinuities.

I present several applications of this work, including smooth level-of-detail control for graphics rendering, compression of geometric models, and animation previewing. The resulting algorithms are shown to run quite efficiently in most cases.

As shapes in computer graphics and modeling become increasingly complex, the techniques developed here provide a powerful and efficient means to extract and simplify essential detail, at the same time eliminating redundant information.

Table of Contents

List of Figures	v
List of Tables	vi
Chapter 1: Introduction	1
1.1 Applications.	1
1.2 Intuitive Application to Surfaces.	3
1.3 Dissertation Overview.	5
1.3.1 Outline.	5
1.3.2 Contributions.	6
1.4 Notation.	7
Chapter 2: Background	10
2.1 General Terms.	10
2.1.1 Topological Type.	10
2.1.2 Meshes and Polyhedra.	11
2.2 Wavelets and Multiresolution Analysis.	11
2.2.1 Multiresolution Analysis on the Real Line.	12
2.2.2 Scaling Function Duals.	13
2.2.3 Support.	14
2.2.4 Benefits of Wavelets.	14
2.2.5 Comparison with Fourier Analysis and Spherical Harmonics.	15
2.2.6 B-Spline Curve and Tensor-Product Wavelets.	15
2.3 Extending Wavelets to Surfaces: A Preview.	17
2.4 Explicit Patching Techniques.	19
2.5 Subdivision Surfaces.	19
2.5.1 Subdivision Rules: Split and Average.	20
2.5.2 Computing Properties of Subdivision Surfaces.	23

Chapter 3:	Multiresolution Analysis from Subdivision	26
3.1	Developing Nested Linear Spaces.	26
3.1.1	Refinable Basis Functions through Subdivision.	27
3.1.2	Nested Linear Spaces.	31
3.2	Inner Products over Subdivision Surfaces.	31
3.2.1	Definition.	31
3.2.2	Computation.	32
3.2.3	Practical Implementation.	39
3.3	Subdivision Wavelets.	39
3.3.1	The Construction.	40
3.3.2	Computation of Wavelets.	40
3.3.3	Locally Supported Approximations to the Wavelets.	41
3.3.4	A Filter Bank Algorithm.	43
3.4	Meshes with Boundary Edges.	45
3.5	Construction of Duals.	45
Chapter 4:	Wavelet Compression of Surfaces	47
4.1	Auxiliary Libraries Required for Compression.	48
4.2	Efficient Data Structures for Compression.	48
4.2.1	Representing the Subdivided Mesh.	49
4.2.2	Vertex Encodings.	51
4.2.3	Input Representation.	53
4.3	Polyhedral Implementation.	53
4.4	Coefficient Selection.	54
4.4.1	Normalization.	54
4.4.2	Selection Strategies.	55
4.5	Reconstruction Algorithms.	57
4.5.1	Naive Reconstruction	57
4.5.2	Rebuilding Parent Structure.	58
4.5.3	Triangulation.	60
4.5.4	Asymptotic Complexity.	64
4.6	Subdivision Connectivity.	64

4.6.1	The General Conversion Problem.	65
4.6.2	A Sphere-to-Octahedron Conversion Algorithm.	65
4.7	Polyhedral Examples.	66
4.7.1	Geometric Data.	67
4.7.2	Color Data.	70
4.7.3	Run Times.	72
Chapter 5:	Interactive Viewing	75
5.1	Animation Previewing.	75
5.2	Automatic Level-of-Detail Control.	76
5.3	Choosing a Coefficient Threshold.	77
5.4	Smoothly Blending between Approximations.	78
5.5	Incremental Reconstruction.	78
5.6	Precomputation.	79
5.6.1	Caching of Wavelet Values.	80
5.6.2	Full Precomputation.	80
5.7	Parallel Computation.	81
Chapter 6:	Smooth Surface Modeling	82
6.1	Motivation.	82
6.2	Smooth Surface Compression.	83
6.3	Efficiency.	83
6.4	Multiresolution Editing.	83
Chapter 7:	Comparison to Related Work	86
7.1	Modeling.	86
7.1.1	Muraki.	86
7.1.2	Finkelstein and Salesin.	86
7.1.3	Forsey and Bartels.	87
7.1.4	Vemuri and Radisavljevic.	88
7.2	Polyhedral Compression.	88
7.2.1	Turk.	90
7.2.2	Rossignac and Borrel.	91

7.2.3	Schroeder <i>et al.</i>	91
7.2.4	Hoppe <i>et al.</i>	92
7.2.5	Subdivision Wavelets.	92
7.2.6	Summary.	93
Chapter 8:	Summary and Future Work	94
8.1	Potential Applications.	95
8.1.1	Radiosity.	95
8.1.2	Multiresolution Editing.	96
8.1.3	Surface Optimization and Multigrid Methods.	96
8.1.4	Three-Dimensional Morphing.	97
8.2	Future Work.	99
8.2.1	Conversion to Subdivision Connectivity.	99
8.2.2	Weighted Inner Product.	99
8.2.3	Linear Time Sparse Matrix Inversion.	100
8.2.4	Locally Supported Orthogonal Wavelets.	100
8.2.5	Real-time Level-of-Detail Control.	100
8.2.6	Preventing Self-Intersection.	101
8.2.7	Simplifying the Topological Type.	101
8.2.8	Inner Product Classification	101
8.2.9	Computing Inner Products with the Duals	102
8.3	Conclusion.	102
	Bibliography	104

List of Figures

1.1	Decomposition of a polyhedral surface.	4
2.1	Refinability of B-spline basis functions.	18
2.2	Loop's subdivision scheme.	21
2.3	A single subdivision step.	22
2.4	Polyhedral subdivision of a tetrahedron and various associated filters.	24
2.5	The Dyn <i>et al.</i> butterfly subdivision rule.	25
3.1	The subdivision limiting process.	27
3.2	A hat function.	33
3.3	The polyhedral inner product mask.	34
3.4	Scaling around an extraordinary point.	36
3.5	Computing the inner product for piecewise linear curves.	37
3.6	A polyhedral wavelet.	41
4.1	Vertex classification on the subdivided tetrahedron.	49
4.2	Encodings for vertices.	52
4.3	Determining polyhedral wavelet coefficients.	54
4.4	Progress of the recursive <i>Expand()</i> procedure.	59
4.5	Progress of the recursive <i>Triangulate()</i> procedure.	62
4.6	Views of the full-resolution Spock polyhedron.	68
4.7	Wavelet approximations of the Spock polyhedron.	69
4.8	Close-up of Earth data before compression	70
4.9	Approximating color as a function over the sphere.	71
6.1	Compression and editing of a smooth surface.	84

List of Tables

1.1	Notation.	8
3.1	Synthesis filter construction times.	44
4.1	General Spock data.	73
4.2	Spock compression results, organized by coefficient threshold.	73
4.3	General Earth data.	74
4.4	Earth compression results, organized by coefficient threshold.	74
7.1	A comparison of polygonal compression methods.	93

Acknowledgments

I thank my mother and father for their constant encouragement and support, and for their help with editing and babysitting. I am very grateful to my dear wife Ivaly for her enduring love throughout the whole process, and most especially for her understanding during the intensity of this last year. I also am thankful to my children, Brendan, Dominic, and Lucia, for their sustaining cheer and for providing an ever increasing incentive to finish.

I thank my advisor Tony DeRose for introducing me to the wonders of the field of geometric design, for his patient interest and support throughout my PhD process, and for his impressive example of integrity and academic responsibility. His remarkable dedication to quality research and teaching is a strong inspiration. Special thanks must also go to Joe Warren for patiently guiding me along the early stages of this work, and for his encouragement along the way. I am also appreciative of Tom Duchamp's active interest in the development of this research.

In my years at the University of Washington, I have enjoyed working alongside many bright graphics students, including Steve Mann, Hugues Hoppe, Jean Schweitzer, Mike Salisbury, and Adam Finkelstein. They have contributed to a stimulating environment, and I have benefited from their friendship, technical help, and discussions.

Dedication

MATRI MEAE PATRIQUE MEO

Chapter 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 that is 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 [DeVore et al. 92].

Two-dimensional wavelets are important for a variety of applications, including image compression. They are generally constructed by forming tensor products of univariate wavelets [Daubechies 92], in much the same way that tensor-product B-spline surfaces are formed by products of univariate B-splines. Unfortunately, tensor-product constructions require 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 [Daubechies 92, Jia & Micchelli 91], but none of these methods are applicable to functions defined on more general topological surfaces, such as over the sphere.

In this dissertation, I show that by using techniques from subdivision surfaces, multiresolution analysis can be extended to functions defined on domains of arbitrary topological type. This generalization, which I term “subdivision wavelets,” dramatically expands the class of applications to which multiresolution analysis can be applied.

1.1 Applications.

Subdivision wavelets extend the benefits of multiresolution analysis to a wide range of new applications. These include:

- *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 for animation.* 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 reducing visible details. Moreover, it is possible to smoothly vary the level of detail, avoiding discontinuous jumps that occur as a result of sudden switching between separate models.
- *Compression of functions defined on surfaces.* The techniques described in this dissertation apply to general kinds of functions defined over surfaces. In addition to geometric applications, subdivision wavelets may be used to create compressed approximations of general surface functions with varying complexity and smoothness. For example, image or texture data, normal information, and temperature or air pressure information may all be compressed over surfaces of arbitrary topological type through the use of subdivision wavelets.
- *Multiresolution editing of surfaces.* Hierarchical B-splines, as introduced by Forsey and Bartels [Forsey & Bartels 88], 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. Subdivision wavelets provide an alternative to hierarchical B-splines, and are capable of representing tangent-plane 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 [Finkelstein & Salesin 94].

- *Global illumination.* Tensor-product wavelets have recently been used to rapidly solve physical simulation problems such as radiosity [Gortler et al. 93, Hanrahan et al. 91, Christensen et al. 94]. The use of wavelets in the more general setting of spherical radiance distributions, such as those of Gondek *et al.* [Gondek et al. 94], requires wavelets adapted to the sphere. This dissertation shows how such wavelets may be developed.
- *Polygon clustering for hierarchical radiosity.* An important method for accelerating hierarchical radiosity involves the clustering of groups of polygons into an appropriate approximation [Smits et al. 94]. 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 [Pentland 92] uses wavelet methods to implement multigrid optimization for surface interpolation over regular grids. Meyers [Meyers 94a, Meyers 94b] 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.
- *Three-dimensional morphing.* An object may continuously transform into another of the same topological type by smoothly blending between the wavelet representation of both. This dissertation lays a geometric basis for such a transformation.

More discussion of applications is provided in Chapters 4, 5, and 6 of this dissertation. Potential applications are discussed in Chapter 8.

1.2 Intuitive Application to Surfaces.

Although the mathematical underpinnings of multiresolution analysis of surfaces are rather involved, the resulting algorithms are quite simple. Before going into the details, let us first understand how the method can be applied to decompose the polyhedral object shown in Figure 1.1(a).

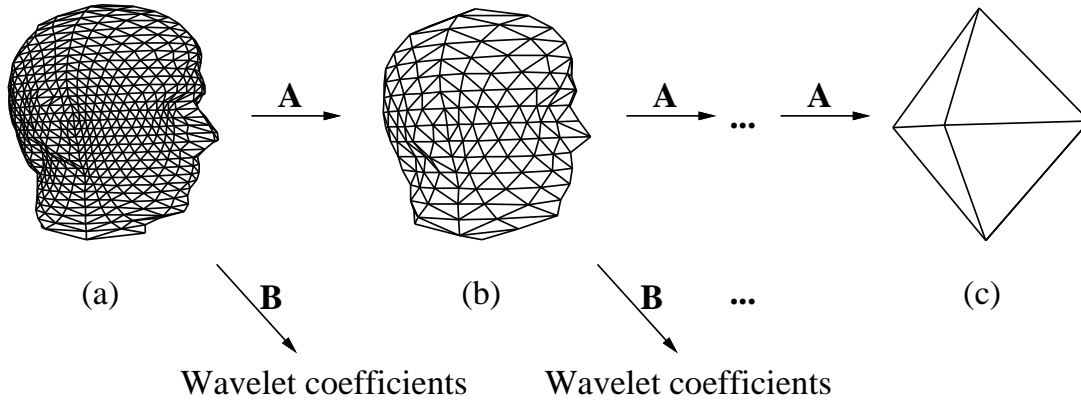


Figure 1.1: *Decomposition of a polyhedral surface.*

The main idea behind multiresolution analysis is the decomposition of a function (in this case, a polyhedron expressed as a parametric function on an octahedron) into a low-resolution part and a “detail” part. The low-resolution part of the polyhedron in Figure 1.1(a) is shown in Figure 1.1(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* (a filter that eliminates fine-level detail) 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.

1.3 *Dissertation Overview.*

This dissertation presents a theoretical foundation for developing multiresolution analysis for surfaces of arbitrary topological type. Because this theory is entirely motivated by its practical applications, this dissertation emphasizes theoretical aspects that have obvious uses in practice.

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 [Catmull & Clark 78, Halstead et al. 93], Loop [Loop 87] or Dyn et al. [Dyn et al. 90] (which produce tangent-plane-smooth G^1 subdivision surfaces); and the modifications by Hoppe *et al.* [Hoppe et al. 94, Hoppe 94] 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.

1.3.1 *Outline.*

Before going into the details of subdivision wavelet construction, it is first necessary to present background information. Chapter 2 gives some background on multiresolution analysis and subdivision surfaces, which are the two key building blocks of subdivision wavelets.

The primary theoretical contribution of this dissertation is presented in Chapter 3, which explains how to create multiresolution analysis over subdivision surfaces. This construction requires: (1) a development of refinability for scaling functions over subdivision surfaces; (2) a discussion of the inner products of the scaling functions; and (3) an efficient technique for implementing the resulting wavelets.

Chapter 4 discusses the details of applying subdivision wavelets to the compression of complex surface functions. The chapter includes a discussion of practical issues, including efficient data structures for manipulating subdivision wavelets.

Applications of subdivision wavelets for interactive viewing are discussed in Chapter 5. These include animation previewing and smooth level-of-detail control.

Chapter 6 shows examples of modeling smooth surfaces with subdivision wavelets.

In Chapter 7, subdivision wavelets are compared to other methods for the applications of surface modeling and polyhedral compression.

Finally, Chapter 8 summarizes this work and suggests interesting areas of future work in subdivision wavelets, including applications which have not yet been made concrete.

1.3.2 Contributions.

Specific contributions of this dissertation include:

- *Construction of refinable scaling functions over compact surfaces of arbitrary topological type.* Scaling functions are the most basic building blocks in multiresolution analysis. In Section 3.1, we show that scaling functions defined over subdivision surfaces are refinable, and hence may be used to construct wavelets.
- *Exact computation of inner products.* The evaluation of an inner product of the scaling functions is the next important step in the construction of wavelets. In Section 3.2, we show that even though the scaling functions are in general only defined implicitly as the limit of an iterative procedure, their inner product may be exactly integrated by setting up a linear system that is derived from subdivision refinement.
- *Construction of wavelets over surfaces of arbitrary topological type.* In Section 3.3, the inner products are used to construct wavelets that are defined over surfaces of arbitrary topological type.
- *Linear time filter banks for surfaces.* In Section 3.3.4, we develop efficient filter bank techniques based on subdivision wavelets. In particular, we develop a locally supported approximation to the wavelets that permits linear time decomposition and reconstruction for interpolatory subdivision schemes.

- *Efficient representation of subdivision surface detail.* A complex subdivision surface is inefficiently represented using standard storage techniques. In Section 4.2, we show how one may more compactly keep information at vertices of a subdivision surface.
- *Continuously varying approximations of detailed surface data.* Chapter 4 explains how subdivision wavelets may be applied to the specific problem of compressing surface data that has been defined through subdivision. This technique may be used to approximate polyhedral surfaces or tangent-plane smooth surfaces arising from subdivision, using L^2 and L^∞ norms. Smoothly varying between approximations at different levels of detail is described in Section 5.4.

Together with this largely theoretical development, actual results of applying subdivision wavelets are described in Section 4.7 and Chapter 6. Although these examples are limited to C^0 polyhedral surfaces and the tangent-plane smooth butterfly scheme, subdivision wavelets may also be applied to other kinds of tangent-plane smooth subdivision surfaces, piecewise-smooth subdivision surfaces [Hoppe et al. 94, Hoppe 94], and to open surfaces that possess boundary curves.

1.4 Notation.

The symbols used in this dissertation follow those described in Table 1.1.

Table 1.1: *Notation used in this dissertation.*

SYMBOL	DESCRIPTION	SECTION
\mathbb{R}	The set of real numbers	
$x, j, \text{etc.}$	Scalar variables	
$\mathbf{x}, \mathbf{p}, \text{etc.}$	Multidimensional variables, vectors and points	
$\mathbf{1}$	The identity matrix	
δ_{ij}	The Kronecker delta: 1 iff $i = j$, 0 otherwise	
M^j	j -th subdivision of base mesh M^0	2.5.1
\widehat{M}^j	The mesh split from M^{j-1} , without averaging	3.1.1
$\phi_k^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The k -th scaling function at level j	3.1.1
$\Phi^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The matrix of $\phi_k^j(\mathbf{x})$	3.1.1
\mathbf{V}^j	Matrix of scaling function coefficients that multiply $\Phi^j(\mathbf{x})$	2.5.1
$V^j, V^j(M^0)$	The span of the scalar-valued $\Phi^j(\mathbf{x})$, parametrized over the base mesh M^0	2.2.1, 3.1.2
$W^j, W^j(M^0)$	The orthogonal complement of V^j in V^{j+1}	2.2.1, 3.3.1
V_*^j	A space spanned by the local support around a scaling function, close to W^j , but still not orthogonal to V^j	3.3.1
$\mathcal{O}^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The portion of $\Phi^j(\mathbf{x})$ involving only the scaling functions appearing at level j that also appeared at level $j - 1$	3.1.1
$\mathcal{N}^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The portion of $\Phi^j(\mathbf{x})$ involving only the newly subdivided scaling functions appearing at level j	3.1.1
$\psi_k^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The k -th wavelet at level j	3.3.1
$\Psi^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R}^l)$ – The matrix of $\psi_k^j(\mathbf{x})$ for a fixed j , where l is the length of $\Psi^j(\mathbf{x})$	3.3.1
$\tilde{\phi}_k^j(\mathbf{x})$: $(\mathbb{R}^n \rightarrow \mathbb{R})$ – The k -th scaling function dual at level j	2.2.2, 3.5

Table 1.1 (continued)

SYMBOL	DESCRIPTION	SECTION
$\boldsymbol{\alpha}_i^j$	The vector of coefficients of ϕ_k^j used to create ψ_i^j	3.3.1
$\boldsymbol{\alpha}^j$	A matrix with columns made up of $\boldsymbol{\alpha}_i^j$	3.3.1
$\tilde{\boldsymbol{\beta}}_i^j$	The vector of coefficients of ϕ_k^j used to create $\tilde{\phi}_i^j$	3.5
$\tilde{\boldsymbol{\beta}}^j$	The matrix whose columns are made up of $\tilde{\boldsymbol{\beta}}_i^j$	3.5
$\langle f(\mathbf{x}), g(\mathbf{x}) \rangle$	Returns a scalar giving the inner product of $f(\mathbf{x})$ and $g(\mathbf{x})$	2.2.1, 3.2.1
\mathbf{O}_i^j	The vector giving the subdivision rule for the i -th old point at level $j + 1$, as multiples of coarse-level points at level j	3.1.1
\mathbf{O}^j	The subdivision matrix for old vertices, made up of rows of \mathbf{O}_i^j	3.1.1
\mathbf{N}_i^j	The vector giving the subdivision rule for the i -th newly subdivided point at level $j + 1$, as multiples of coarse-level points at level j	3.1.1
\mathbf{N}^j	The subdivision matrix for new vertices, made up of rows of \mathbf{N}_i^j	3.1.1
\mathbf{P}^j	The complete subdivision matrix, made from concatenating \mathbf{O}^j on top of \mathbf{N}^j	2.5.1, 3.1.1
\mathbf{I}^j	The inner product matrix at subdivision level j : $\mathbf{I}_{ik}^j := \langle \phi_i^j, \phi_k^j \rangle$	3.2.2

Chapter 2

BACKGROUND

Before proceeding with the details of the construction of wavelets over surfaces of arbitrary topological type, it is first necessary to develop some background material.

Section 2.1 begins with definitions of basic topological terms. In Section 2.2 we present the theory behind standard wavelets defined over the real line and the regular grid. The usual formulation of wavelets is firmly tied to parametrization over \mathbb{R}^n ; in order to extend wavelets to surfaces of arbitrary topological type, we first need to find a surface representation which gives refinability. These difficulties are explored in Section 2.3. One possible surface type, the explicit patching methods, are examined in Section 2.4, and shown to be unsatisfactory. The chapter concludes with a discussion of subdivision surfaces, which provide the necessary machinery for developing wavelets on smooth surfaces of arbitrary topological type.

2.1 *General Terms.*

2.1.1 *Topological Type.*

The *topological type* of a two-dimensional surface or domain refers to its genus, presence of boundary curves, etc. Any object of a particular topological type may be deformed into another object of the same topological type through a *homeomorphism*, an elastic transformation of the shape. Thus, the outer surfaces of a coffee cup and a doughnut are of the same topological type because either surface may be stretched into the other. However, a sphere and a coffee cup are not of the same topological type, because a coffee cup has an additional handle.

2.1.2 Meshes and Polyhedra.

A *mesh* $M = (V, F)$ is a set of vertices V and a set of faces F . Each *face* $f \in F$ is an ordered set of $n > 2$ vertices, $f = (v_0, v_1, \dots, v_{n-1})$, where $v_i \in V$, and $v_i \neq v_j$ for $i \neq j$. A directed *edge* is associated between each successive pair of face vertices v_i and $v_{(i+1) \bmod n}$. If a directed edge e appears in a mesh but the opposing directed edge does not appear, then the mesh is said to have a *boundary* at e .

A representation of a mesh may store data at the vertices. When the representation is used to store 3D geometry, the vertex field contains data in \mathbb{R}^3 . Weiler [Weiler 85] discusses various methods for the representation of meshes; a data structure more appropriate for subdivision surfaces is presented in Section 4.2 of this dissertation.

A mesh may represent a surface of any topological type, including sheets, spheres and Klein bottles. Holes may be represented by a mesh with distinguished edges. All faces are triangles in a *triangular* mesh.

A *polyhedron* is a piecewise linear surface. For our purposes, polyhedra will usually represent surfaces in three dimensions. Polyhedra may be implemented using meshes, each face of which represents a planar polygon.

2.2 Wavelets and Multiresolution Analysis.

Wavelets have roots in approximation theory, signal processing, and physics. The formal development of multiresolution analysis as a rigorous theory dates to Mallat in 1989 [Mallat 89], although related pyramidal techniques for image analysis [Burt & Adelson 83] precede his work. In the brief time since their development, wavelets have found use in signal analysis [Mallat 89, Rioul & Vetterli 91], image processing [DeVore et al. 92, Mallat & Hwang 92], physics [Arneodo et al. 94], and numerical analysis [Beylkin et al. 91]. More recently, wavelets have been applied to a wide range of computer graphics problems, including curve modeling [Finkelstein & Salesin 94], image editing [Berman et al. 94], improved global illumination [Schröder et al. 93, Gortler et al. 93, Christensen et al. 94], optimization for surface interpolation [Pentland 92], creating surfaces from contours [Meyers 94a, Meyers 94b], and animation control [Liu et al. 94]. An excellent tutorial on the use of wavelets in computer graphics is presented by Stollnitz *et al.* [Stollnitz et al. 94].

2.2.1 Multiresolution Analysis on the Real Line.

Multiresolution analysis as formulated by Mallat [Mallat 89] 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 $V^0 \subset V^1 \subset V^2 \subset \dots$ 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

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

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 3.3.4, the analysis and synthesis filters are determined by considering various ways of changing bases between scaling functions and wavelets.

Thus, multiresolution analysis requires a nested sequence of linear spaces V^j . Over the real line, this is 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 expressed as

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

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

The scaling functions spanning V^j are denoted by $\phi_i^j(x)$. In a similar fashion, the wavelets $\psi_i^j(x)$ are ordinarily defined to be functions that span the orthogonal complement space W^j . Like the scaling functions, the $\psi_i^j(x)$ are also generated by dilations and translations of a single function $\psi(x)$, with the additional property that they are defined to be orthogonal to the $\phi_i^j(x)$. More precisely, every basis function $\psi_i^j(x)$ of W^j is orthogonal to every basis function of V^j under the chosen inner product.

The standard inner product for functions on the real line is defined to be

$$\langle f, g \rangle := \int_{-\infty}^{\infty} f(x) g(x) dx,$$

with $f, g \in V^j$. This definition requires that both functions f and g are integrable over their entire domain.

Additionally, a function f in the spaces V^j inherits properties from the component basis functions. In general, f will only be smooth if the scaling functions $\phi_i^j(x)$ are also smooth.

To summarize, a multiresolution analysis requires the following properties:

- *A nested sequence of linear spaces V^j .* This property occurs for any sequence of refinable scaling functions. It is used to establish detail at various resolutions.
- *An inner product defined over the spaces V^j .* The inner product is used to measure orthogonality between functions in the nested spaces. It is based on integration, and hence requires that the scaling functions be integrable.
- *Smoothness (optional).* For many computer graphics and modeling applications, it may be desirable that the $\phi_i^j(x)$ are smooth.

2.2.2 Scaling Function Duals.

Given arbitrary scaling functions $\phi_i^j(x) \in V^j$, one may construct functions $\tilde{\phi}_k^j(x) \in V^j$ such that

$$\langle \phi_i^j(x), \tilde{\phi}_k^j(x) \rangle = \delta_{ik}.$$

This function $\tilde{\phi}_k^j(x)$ is called the *dual* of $\phi_k^j(x)$.

When $f(x)$ is defined as

$$f(x) = \sum_i v_i^j \phi_i^j(x),$$

it can be shown that

$$v_i^j = \langle f(x), \tilde{\phi}_i^j(x) \rangle. \quad (2.1)$$

The dual is therefore useful for finding the coefficients v_i^j of the scaling functions $\phi_i^j(x)$, given an arbitrary function $f(x)$.

2.2.3 Support.

The *support* of a function in \mathbb{R}^n is the range of parameter values for which the function is nonzero. A support is *local* if it lies within a constant distance of some parameter value. It is *global* if it is potentially unbounded.

The support of a wavelet has an important effect on its practical efficiency. Wavelets with local support usually result in linear time algorithms, while those with global support require $O(n^2)$ processing time.

2.2.4 Benefits of Wavelets.

Wavelets have many benefits that make them useful in practice. Orthogonality ensures that the coarser-level approximations are the least-squares best approximations to the finer function. Moreover, wavelet approximations are efficiently generated by simply applying the low-pass filter and *downsampling* (throwing away every other coefficient). When the wavelet's support is local, the resulting algorithm can usually be constructed to run in a very efficient $O(n)$ time in the number of initial coefficients.

For functions that naturally appear in the real world, including most signals and images, the coarser-level least-squares approximation generated by the filter bank is a good approximation of the original function. As a result, the missing detail represented by the wavelet coefficients is negligible for many values. For completely random data, such as a white noise signal with information appearing randomly at every level of detail, one would not expect to find this property.

This property makes wavelets naturally suited for *lossy* compression (compression that allows an imperfect reconstruction of the original data, with the balance between fidelity and approximation size controlled by a single parameter). By varying the threshold at which finer-level detail is considered significant enough to preserve, one may construct a highly efficient lossy compression algorithm based on wavelets.

2.2.5 *Comparison with Fourier Analysis and Spherical Harmonics.*

Multiresolution analysis shares similarities with Fourier analysis. The Fourier transform may also be used to extract frequency information from a discrete set of samples, and the Danielson-Lanczos Lemma [Press et al. 89] for the Fast Fourier Transform bears a strong resemblance to the filter bank process used in multiresolution analysis.

A significant difference is that wavelet analysis gives information not only about the frequency of the sampled signal, but also about location. This additional knowledge is invaluable for the analysis of *non-stationary signals* (signals whose properties may change over time), because it may be used to pinpoint the position of important high-frequency data. Additionally, although the Fast Fourier Transform decomposes a signal in time $O(n \log n)$, a filter bank can usually be constructed to run in linear time.

Spherical harmonics extend Fourier analysis to functions on spherical domains. Spherical harmonics have been used in computer graphics by Sillion *et al.* [Sillion et al. 91] for modeling bidirectional reflectance distribution functions (BRDFs) in global illumination solutions. In this dissertation, methods for developing multiresolution analysis on arbitrary topological types include spherical domains as a special case. On spherical domains, subdivision wavelets enjoy advantages over spherical harmonics in much the same way that standard wavelets provide advantages over Fourier analysis.

2.2.6 *B-Spline Curve and Tensor-Product Wavelets.*

Wavelets are well understood for B-spline curves, and Chui [Chui 92a, Chui 92b] treats the subject exhaustively.

For B-spline wavelets, translations and dilations of the B-spline basis functions [Bartels et al. 87] are used as scaling functions. These translations and dilations are sufficient to set up the nested sequence of linear spaces V^j required for a multiresolution analysis over the

infinite real line. Mallat's original B-spline wavelets [Mallat 89] are constructed to be mutually orthogonal, but have global support and exponential decay. Chui's B-spline wavelets are specially constructed to have minimal support, though at the cost of giving up mutual orthogonality. Quak and Weyrich [Quak & Weyrich 93] make use of a clever technique that allows them to develop linear time reconstruction filters for B-spline wavelets.

When wavelets are used to represent vector-valued functions, such as curves and surfaces, each coordinate in the range space may be described by a separate scalar-valued function. Thus, a planar curve, which is a function mapping from the real line to \mathbb{R}^2 , may be independently decomposed in both coordinates. Thus, a single wavelet decomposition step for a planar curve in V^{j+1} produces a collection of pairs in \mathbb{R}^2 giving an approximation to the curve in V^j together with a collection of pairs in \mathbb{R}^2 describing the missing detail W^j . Both coordinates in a pair are weighted when choosing coefficients to add back. We use this technique of independent coordinates to develop wavelet analysis of surfaces.

Multiresolution editing of B-spline curves and tensor-product surfaces is presented by Finkelstein and Salesin [Finkelstein & Salesin 94]. Their implementation is not based on simple translates and dilates, because the scaling functions and wavelets near the ends of the curve must be defined specially.

The extension of B-spline wavelets to tensor-product surfaces parametrized over the infinite grid is straightforward. For tensor-product B-spline surfaces, scaling functions may again be defined as translations and dilations of a single function. One may develop wavelets over bounded rectangular domains by forming a tensor product of bounded interval B-spline wavelets.

When developing objects in the real world, it is desirable to have the capability to model any shape that it is possible to create. Unfortunately, tensor-product patches have problems achieving this goal.

A tensor-product patch is parametrized over a rectangular piece of the plane. However, it is not possible to parametrize a surface of arbitrary topological type over a rectangular domain without introducing degeneracies, such as at the poles of a sphere. Nondegenerate tensor-product patches may only model flat rectangular sheets, cylinders, and tori, and are unable to represent shapes that are topologically equivalent to spheres, 2-holed tori, and others.

Surfaces of arbitrary topological type are often modeled with tensor-product B-splines

by introducing degeneracies. However, when these degeneracies are introduced there is the danger of violating standard geometric assumptions. This requires much more care when writing a modeling system. As an example, the trimmed surfaces that result by cutting a section from a rectangular patch typically require the solution of surface-surface intersection, which is, in general, a very complex and time-consuming problem.

2.3 Extending Wavelets to Surfaces: A Preview.

In this dissertation, we extend techniques like those used to construct wavelets for B-spline curves and tensor-product surfaces to domains of arbitrary topological type. This extension is based on a generalization of the usual definition of scaling functions and wavelets.

B-spline curves and tensor-product surfaces over the infinite plane have domains that are *spatially invariant*. Intuitively, spatial invariance means that every place in the domain looks like every other place. Spatial invariance allows translations and dilations of the same scaling function to be defined at arbitrary places in the domain, an important property used to construct the nested V^j spaces.

However, surfaces of arbitrary topological type are not spatially invariant, which means that one cannot set up scaling functions to span V^j that are translations and dilations of a single function. Chapter 3 discusses a more general way of looking at multiresolution analysis that is based on refinability.

Refinability generalizes dilation and translation, and is essential for multiresolution analysis. Refinability means that a coarse-level scaling function may be expressed as a linear combination of finer-level functions. Refinability allows the filter bank to run in reverse and allows the detail to be concisely represented in the wavelet basis.

An example of refinability for cubic B-spline basis functions is shown in Figure 2.1. Although this example shows the refinability of a function in terms of its finer-level dilations, the methods of this dissertation are developed from the more general refinability of a function in terms of functions that are not always simple dilations. For B-splines, there is rich theory to relate a coarse-level B-spline basis function to a combination of finer-level basis functions. Examples are the Oslo algorithm for knot insertion [Lyche et al. 80], Boehm's knot insertion algorithm [Boehm 80], and blossoming [Ramshaw 87].

Another property that is used to construct wavelets from B-splines is integrability. Be-

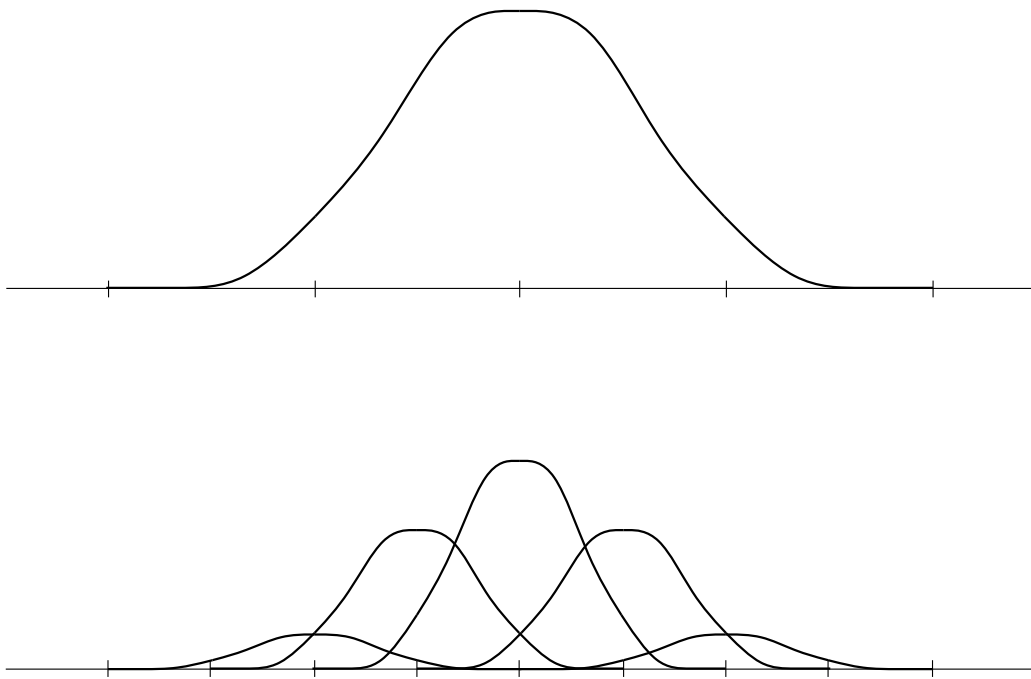


Figure 2.1: *Refinability of B-spline basis functions. The coarse-level basis function on the top is equivalent to the sum of the finer-level basis functions on the bottom.*

cause they are piecewise polynomials, it is straightforward to integrate B-splines to compute the standard inner product that is required for orthogonality. If we substitute another surface form for B-splines, we will need to ensure that it may be integrated to produce inner products.

2.4 *Explicit Patching Techniques.*

As was discussed in Section 2.2.6, tensor-product B-splines are incapable of representing surfaces of arbitrary topological type without degeneracies. There are two alternatives for modeling arbitrary topological types without degeneracies: explicit patching methods and subdivision surfaces.

Currently, the most common way to represent tangent-plane smooth surfaces of arbitrary topological type is to build a collection of explicit parametric patches. These patches are nearly always polynomial or rational, and require considerable care to ensure that they meet with tangent-plane continuity — both along their common boundaries and around vertices. There exist a multitude of methods; a survey of their use in triangular interpolation is provided by Mann *et al.* [Mann et al. 92, Mann 92] and Lounsbery *et al.* [Lounsbery et al. 92].

Although explicit patching methods are capable of representing smooth surfaces of arbitrary topological type, and they are integrable, they lack the property of refinability. Without refinability, it is difficult to see how to develop an effective multiresolution representation.

2.5 *Subdivision Surfaces.*

Subdivision surfaces are an alternative method for constructing smooth surfaces of arbitrary topological type. We will see in Section 3.1.1 that it is possible to develop refinable scaling functions on subdivision surfaces. Thus, they are suitable primitives for constructing multiresolution analysis over surfaces of arbitrary topological type. Subdivision surfaces are discussed at length throughout the rest of this chapter.

The study of subdivision surfaces traces back to 1978, when papers by Doo and Sabin [Doo 78, Doo & Sabin 78] and by Catmull and Clark [Catmull & Clark 78] developed subdivision procedures for defining tangent-plane smooth surfaces. Although subdivision

procedures are simple to implement, the idea failed at first to catch on because there were no convenient methods of analyzing the surfaces, which are only defined implicitly as the limit of an infinite process.

More recently, subdivision surfaces have enjoyed a resurgence of interest. Papers by Reif [Reif 92, Reif 93], Cavaretta *et al.* [Cavaretta et al. 91], Dyn *et al.* [Dyn et al. 93], and Halstead *et al.* [Halstead et al. 93] have provided convenient tools for analyzing and implementing subdivision surfaces. As a result, their use has become more practical.

2.5.1 Subdivision Rules: Split and Average.

Intuitively speaking, subdivision surfaces are defined by iteratively refining a control mesh M^0 so that the sequence of increasingly faceted meshes M^1, M^2, \dots converges to some limit surface $S = M^\infty$, as is shown in Figure 2.2. 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 nonsquare matrix of constants \mathbf{P}^j such that

$$\mathbf{V}^{j+1} = \mathbf{P}^j \mathbf{V}^j. \quad (2.2)$$

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.

Subdivision schemes can be categorized as *primal* or *dual*. A subdivision scheme is primal if the faces of the mesh are refined into subfaces by the refinement procedure. Catmull-Clark subdivision [Catmull & Clark 78, Halstead et al. 93] is a primal scheme based on subdivision of quadrilateral faces. It is notable for reproducing bicubic B-spline surfaces in areas away from *extraordinary points* (any original vertex of valence other than six, where the *valence* is defined to be the number of edges incident to a vertex). Polyhedral subdivision, the butterfly scheme, and Loop's method [Loop 87] 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 [Doo 78, Doo & Sabin 78] is a dual scheme based on quadrilaterals. For simplicity, we shall restrict the discussion in this dissertation to primal triangular sub-

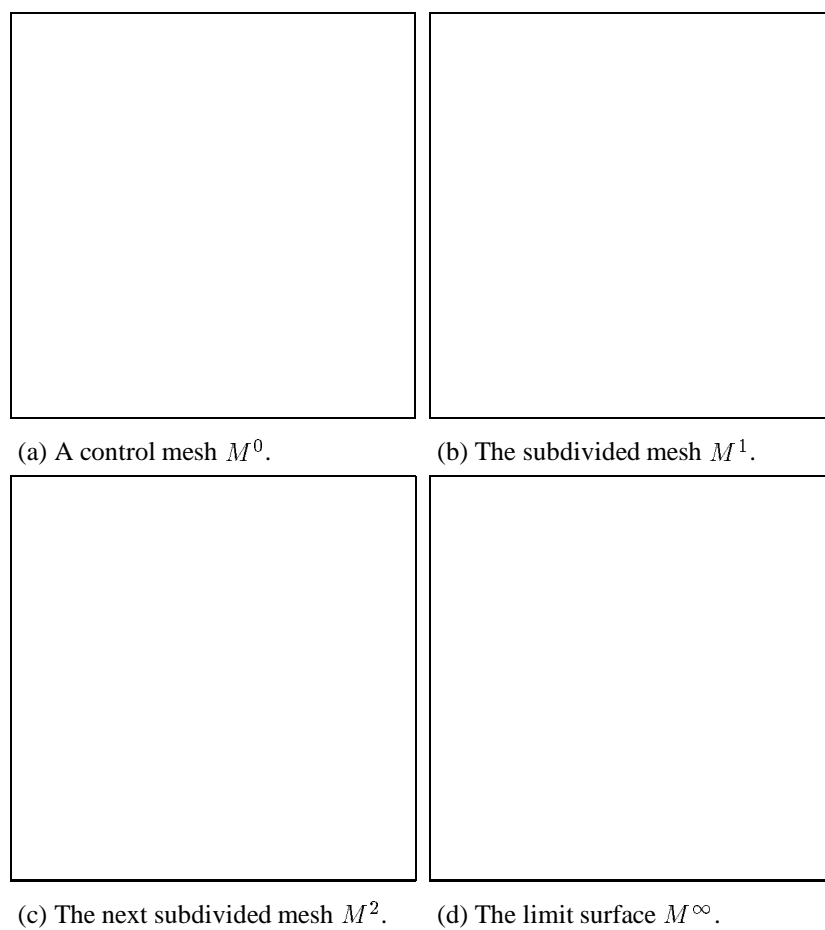


Figure 2.2: *Loop's subdivision scheme applied to a stellated dodecahedron.*

division schemes, although these results hold more generally for any uniformly convergent primal subdivision scheme.

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 2.3. 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.

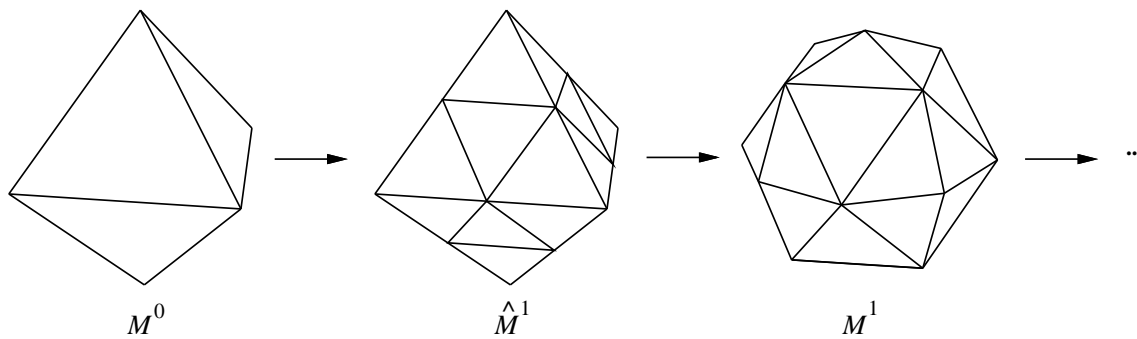


Figure 2.3: A single subdivision step on the octahedron. M^0 is first split into \widehat{M}^1 . The vertices on \widehat{M}^1 are then repositioned using a local linear averaging rule. After one complete subdivision step, the result is M^1 .

The simplest example of such a scheme is polyhedral subdivision. Its splitting step is the same as that of other primal schemes, but its averaging step is simply the identity — that is, no vertices are repositioned. Thus, given a polyhedron M^0 with triangular faces, a new polyhedron M^1 is built directly by splitting each triangular face of M^0 into four subfaces as in Figure 2.4. The matrix \mathbf{P}^0 characterizing the first polyhedral subdivision step on the tetrahedron is also shown in Figure 2.4. 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 of the vertices of M^j that correspond 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 extraordinary points

become increasingly isolated in an otherwise regular tiling of the mesh.

Polyhedral subdivision converges to the original polyhedron covering M^0 , that is, to a C^0 surface. Other schemes have also been developed that converge to tangent-plane smooth limit surfaces that either approximate or interpolate the vertices of M^0 .

The Dyn, Levin and Gregory butterfly scheme [Dyn et al. 90] is a subdivision method that converges to a tangent-plane smooth surface. Butterfly subdivision interpolates the vertices of the mesh, and thus, as is shown in Section 3.3.4, is especially practical for the implementation of wavelets on smooth surfaces. The butterfly scheme uses a simple rule for subdivision, depicted graphically in Figure 2.5. In the figure, w is a tension parameter. When w is zero, the rule exactly reproduces linear subdivision. (The lepidopteran shape of the coarse-level structure influencing \mathbf{q} lends the butterfly scheme its name.)

2.5.2 Computing Properties of Subdivision Surfaces.

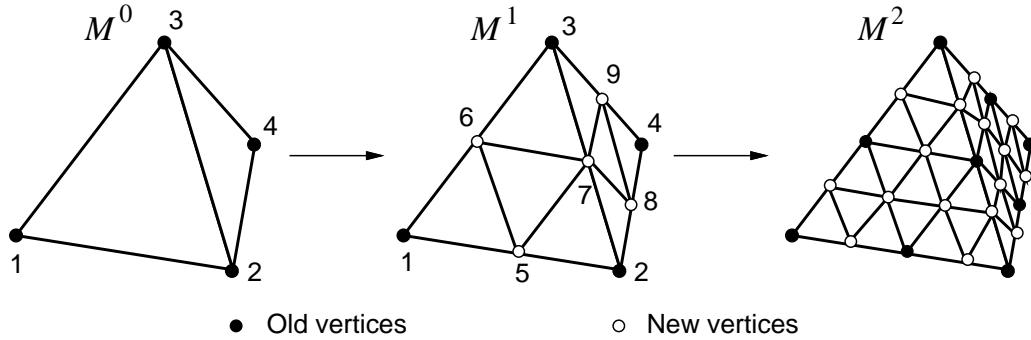
Subdivision surfaces are easy to implement, and can model shapes of arbitrary topological type. Additionally, tangent-plane smooth subdivision rules, such as those of Loop or Catmull and Clark allow meshes of arbitrary topological type to converge to a tangent-plane smooth surface in the limit.

Hoppe *et al.* [Hoppe et al. 94, Hoppe 94] make subdivision even more versatile for modeling. They develop specialized subdivision rules that allow sharp edges and other types of discontinuities commonly found in real-world objects.

An early difficulty of computing with subdivision surfaces was their implicit definition as the limit of an infinite sequence of linear averaging operations. In particular, there is no explicit closed formula for exactly evaluating the surface at an arbitrary point. The lack of an explicit expression for the evaluation of subdivision surfaces makes it difficult to analyze their behavior in the limit, and made early experiences with these surfaces unsatisfactory for many applications. Many of these problems have been recently overcome.

Halstead *et al.* [Halstead et al. 93] apply techniques from Eigenanalysis to examine the behavior of the Catmull-Clark scheme in the limit. Their methods also apply to other kinds of subdivision procedures based on local linear averaging.

Based on an analysis of the Eigenstructure of the local subdivision matrix (a matrix giving the subdivision rule for a local neighborhood around a point), Halstead *et al.* derive



$$\mathbf{P}^0 = \begin{pmatrix} \mathbf{O}^0 \\ \mathbf{N}^0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix} \quad \mathbf{Q}^0 = \begin{pmatrix} -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & -\frac{3}{8} \\ -\frac{3}{8} & \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & \frac{1}{8} & -\frac{3}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & -\frac{3}{8} & -\frac{3}{8} & -\frac{3}{8} \\ \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} & -\frac{1}{8} \\ \frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{5}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & \frac{5}{8} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{A}^0 \\ \mathbf{B}^0 \end{pmatrix} = \begin{pmatrix} \frac{7}{16} & -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} \\ -\frac{1}{16} & \frac{7}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{16} & \frac{7}{16} & -\frac{1}{16} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & \frac{7}{16} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{3}{8} \\ \hline -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{I}^0 = \begin{pmatrix} 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 1 \end{pmatrix} \quad \mathbf{a}^0 = \begin{pmatrix} \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} \\ \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{3}{8} \end{pmatrix}$$

Figure 2.4: Polyhedral subdivision of a tetrahedron and various associated filters

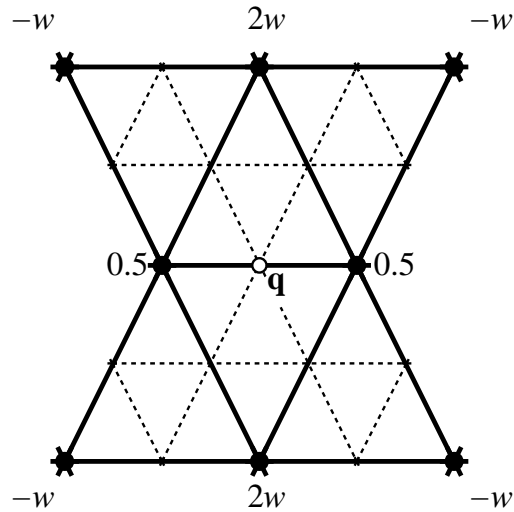


Figure 2.5: The Dyn *et al.* butterfly subdivision rule. The new point \mathbf{q} is derived from an affine combination of points in its coarser-level neighborhood, multiplied by the indicated scalar weights. w is a shape parameter.

a separate local averaging rule that gives the point to which any vertex v on a mesh M^j converges on the limit surface. They also derive a formulation for the normal at the same location. Their results demonstrate that one may evaluate positions and normals at a dense set of points on a subdivision surface as efficiently as for B-spline surfaces.

More surprisingly, Halstead *et al.* also show that even though the limit of a subdivision surface is not given in a closed form, a fairness functional evaluated over the limit surface can nonetheless be computed exactly. The techniques developed in Section 3.2 that allow one to integrate the scaling functions defined over subdivision surfaces are reminiscent of this result.

Although they can be somewhat tricky to analyze, subdivision surfaces provide a versatile and efficient tool for modeling surfaces of arbitrary topological type. In the next chapter, subdivision surfaces are also shown to have the refinability property essential for constructing a multiresolution analysis.

Chapter 3

MULTIRESOLUTION ANALYSIS FROM SUBDIVISION

The central contribution of this dissertation is the construction of wavelets over surfaces of arbitrary topological type. In this chapter, I provide the details of this construction.

A fundamental ingredient for multiresolution analysis is a sequence of nested linear spaces. In Section 3.1, nested spaces are developed for surfaces of arbitrary topological type. These spaces are defined implicitly by demonstrating the existence of refinable scaling functions on subdivision surfaces. In Section 3.2, I show how these scaling functions may be efficiently and exactly integrated in order to compute the inner product functions necessary for orthogonality. Section 3.3 addresses the development of wavelets computed from the inner product and discusses locally supported approximations to the wavelets that are more appropriate for practical use. These methods may be used in general to construct wavelets from any uniformly convergent local and continuous subdivision procedure.

3.1 Developing Nested Linear Spaces.

The usual formulation of wavelets over regular domains sets up a series of nested spaces formed 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 define 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, etc. Subdivision wavelets are based on 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 [Daubechies 92], 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.

3.1.1 Refinable Basis Functions through Subdivision.

Subdivision can be used to define a collection of refinable basis functions. This is done by first showing that subdivision surfaces can be parametrized by a function $S(\mathbf{x})$, where \mathbf{x} ranges over M^0 . One may then show that the parametrization can be used to define the scaling functions. Parametrizing the scaling functions over a domain M^0 of arbitrary topological type differs sharply from the 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 3.1). 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.

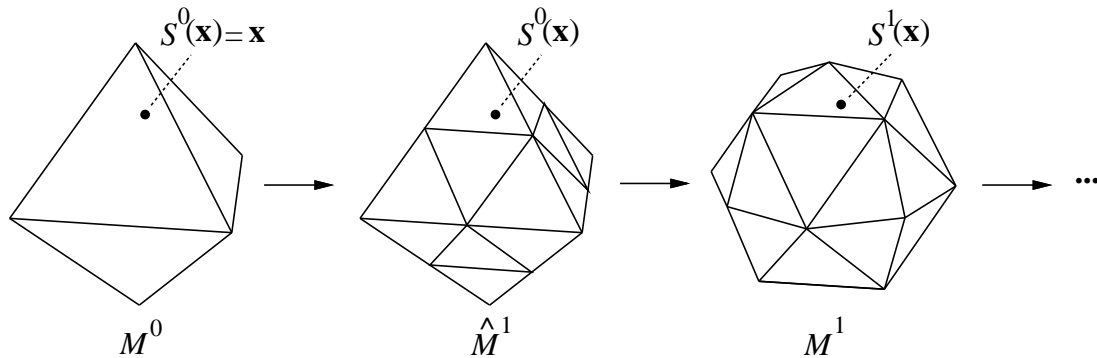


Figure 3.1: *The subdivision limiting process.*

More precisely, the parametrization $S(\mathbf{x})$ can be established 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. $S(\mathbf{x}) := \lim_{s \rightarrow \infty} S^s(\mathbf{x})$.

Duchamp and Schweitzer [Duchamp & Schweitzer 94] show that the parametrization $S(\mathbf{x})$ exists for any local and uniformly convergent continuous subdivision procedure. Next we see that $S(\mathbf{x})$ 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 $k = 1, \dots, s$, 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 (nonsquare) matrices $\mathbf{P}^0, \dots, \mathbf{P}^{k-1}$ 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

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

As a simple corollary to Lemma 1, note that

$$\Phi^{s \leftarrow j}(\mathbf{x}) = \Phi^{s \leftarrow j+1}(\mathbf{x}) \mathbf{P}^j. \quad (3.1)$$

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

$$S(\mathbf{x}) = \sum_i v_i^j \phi_i^j(\mathbf{x}). \quad (3.2)$$

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

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

It is shown in Duchamp and Schweitzer [Duchamp & Schweitzer 94] that $S(\mathbf{x})$ exists for any local and uniformly convergent continuous subdivision procedure and for any choice of control points \mathbf{V}^j . In particular, $S(\mathbf{x})$ must converge if all entries of \mathbf{V}^j are chosen to be the origin, except for the i -th entry of \mathbf{V}^j . This implies that in the limit, the i -th entry in $\Phi^{s \leftarrow j}$ must also exist and be continuous. Therefore, continuity of matrix multiplication allows us to rewrite $S(\mathbf{x})$ as:

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

The desired result is obtained by taking

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

and then expanding the matrix product $\Phi^{s \leftarrow j}(\mathbf{x}) \mathbf{V}^j$ in summation notation. \square

It is convenient to rewrite Equation 3.2 in matrix form as

$$S(\mathbf{x}) = \Phi^j(\mathbf{x}) \mathbf{V}^j, \quad (3.3)$$

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 2.2. Equation 3.3 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, the scaling functions ϕ_i^j are continuous, hence they are also integrable [Bartle 64].

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 3.1 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

$$\Phi^j(\mathbf{x}) = \Phi^{j+1}(\mathbf{x})\mathbf{P}^j. \quad (3.4)$$

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

For primal subdivision schemes, it is convenient to write Equation 3.4 in block matrix form by writing $\Phi^{j+1}(\mathbf{x})$ as

$$\Phi^{j+1}(\mathbf{x}) = (\mathcal{O}^{j+1}(\mathbf{x}) \ \mathcal{N}^{j+1}(\mathbf{x})). \quad (3.5)$$

In this equation, $\mathcal{O}^{j+1}(\mathbf{x})$ consists of all scaling functions $\phi_i^{j+1}(\mathbf{x})$ associated with the old vertices of M^j (the black vertices in Figure 2.4) 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 2.4). Equation 3.4 can now be expressed in block matrix form:

$$\Phi^j(\mathbf{x}) = (\mathcal{O}^{j+1}(\mathbf{x}) \ \mathcal{N}^{j+1}(\mathbf{x})) \begin{pmatrix} \mathbf{O}^j \\ \mathbf{N}^j \end{pmatrix}, \quad (3.6)$$

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.

3.1.2 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 3.4 implies that these spaces are indeed nested; that is,

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

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

3.2 Inner Products over Subdivision Surfaces.

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.

3.2.1 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 3.1. 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

$$\langle f, g \rangle := \int_{\mathbf{x} \in M^0} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}, \quad (3.7)$$

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 .

This 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 definition 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 4.7.

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. This point is discussed further in Section 5.6.

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.

3.2.2 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 . (The hat function at a vertex \mathbf{p} in a mesh is the piecewise linear function which is 1 at \mathbf{p} , blends smoothly to 0 at the neighbors of \mathbf{p} , and is

0 for all other vertices. An example is shown in Figure 3.2.) When functions f and g are combinations of piecewise linear scaling functions, it is fairly simple to directly compute the integral of Equation 3.7.

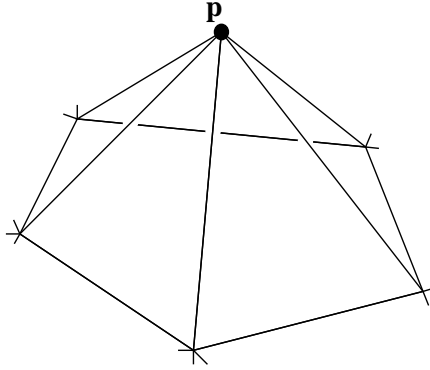


Figure 3.2: A piecewise linear hat function for a vertex \mathbf{p} of valence 5.

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. The approach presented here for computing integrals of wavelets based on their refinement equations is similar to independent work by Dahmen and Micchelli [Dahmen & Micchelli 93]. Their method, however, is restricted to integration over the uniform (spatially invariant) setting.

For any ϕ_i^j constructed from a continuous, uniformly convergent, and local subdivision procedure, one may compute $\langle f, g \rangle$ exactly whenever f and g are given as expansions in ϕ_i^j :

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

Bilinearity 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 2.4.

When the scaling functions in $\Phi^j(\mathbf{x})$ 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 3.3, where the central weight is simply the valence of the central vertex.

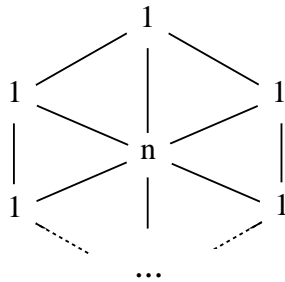


Figure 3.3: *The polyhedral inner product mask.*

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 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

$$\mathbf{I}^j = \int_{\mathbf{x} \in M^0} (\Phi^j(\mathbf{x}))^T \Phi^j(\mathbf{x}) d\mathbf{x}, \quad (3.8)$$

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 3.4 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 an i' such that

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

where $\sigma_i : M^0 \rightarrow M^0$ 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 3.4. 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

$$\mathbf{I}^{j'} = (\mathbf{P}^{j'})^T \mathbf{I}^{j'+1} \mathbf{P}^{j'} \quad (3.9)$$

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. Once an absolute scale for the homogeneous system is chosen, the system can be 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 a nonhomogeneous one.

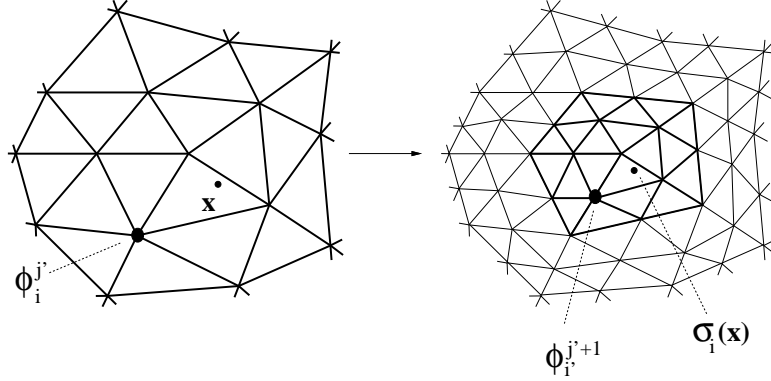


Figure 3.4: The diagram on the left depicts the triangulation of M^0 produced after j' subdivision steps; that is, after j' recursive midpoint splits. The diagram on the right corresponds to $j' + 1$ subdivision steps. $\phi_i^{j'}$ denotes the scaling function for the i -th vertex of $M^{j'}$, and $\phi_i^{j'+1}$ 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.

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 $\mathbf{I}^{j'-1}, \mathbf{I}^{j'-2}, \dots, \mathbf{I}^0$ can be successively determined via Equation 3.9.

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 3.5. 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:

$$\phi_{2i}^j(x) = \frac{1}{2}\phi_{i-1}^{j-1}(x) + \phi_i^{j-1}(x) + \frac{1}{2}\phi_{i+1}^{j-1}(x). \quad (3.10)$$

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 symmetric to 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 $\frac{1}{4}$.

In general, we are not so fortunate as to be able to always find the inner product through exact integration. 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.

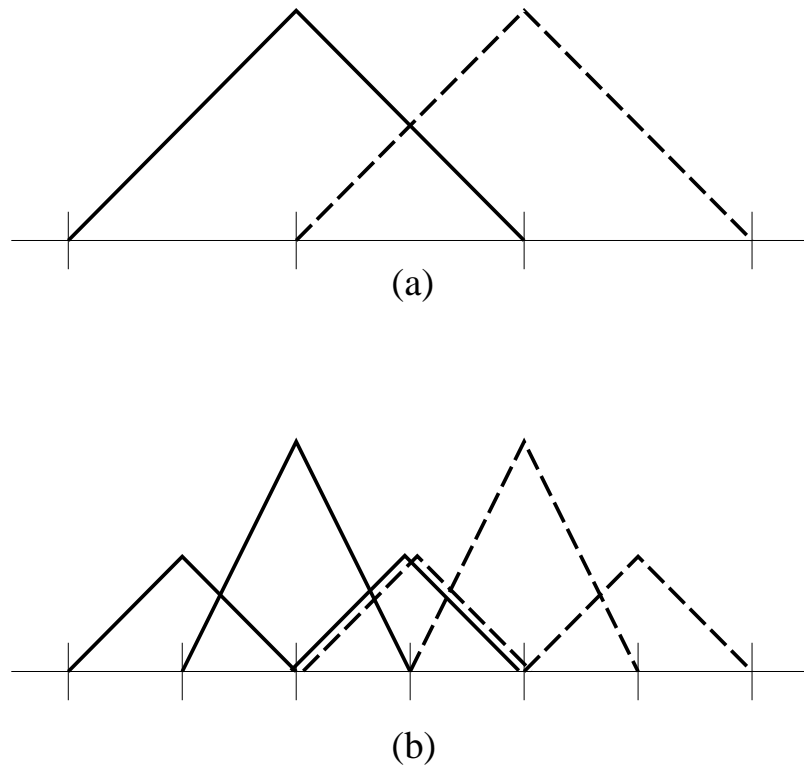


Figure 3.5: *Computing the inner product for the two piecewise linear curves shown in (a). After refinement, the problem reduces to computing the nine combinations of inner products between a solid function and a dashed function in (b).*

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

$$\begin{aligned} x_1 &:= \langle \phi_{2i}^j(x), \phi_{2i}^j(x) \rangle \\ x_2 &:= \langle \phi_{2i}^j(x), \phi_{2(i+1)}^j(x) \rangle. \end{aligned} \quad (3.11)$$

Figure 3.5(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 3.11 is shown as the inner product of the solid function with the dashed-line function.

The next step is to refine with Equation 3.10:

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

Bilinearity of inner products allows this to be expanded to:

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

The result after refinement is shown in Figure 3.5(b). Through refinability, Equation 3.11 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:

$$x_2 = \frac{1}{2}\langle \phi_i^{j-1}(x), \phi_{i+1}^{j-1}(x) \rangle + \frac{1}{4}\langle \phi_{i+1}^{j-1}(x), \phi_{i+1}^{j-1}(x) \rangle + \frac{1}{2}\langle \phi_{i+1}^{j-1}(x), \phi_{i+2}^{j-1}(x) \rangle. \quad (3.12)$$

The inner products in the first and last terms of Equation 3.12 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 3.12 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.

3.2.3 Practical Implementation.

The above description establishes a theoretical framework for computing inner products of refinable scaling functions. In practice, it is more computationally efficient to reduce the size of the linear system by detecting repeated cases on each level. For example, in a sufficiently subdivided neighborhood, all scaling functions centered over an ordinary point immediately adjacent to a non-boundary extraordinary point of valence n are symmetric and of the same shape. They may therefore be treated equally for the purposes of computing inner products.

Detecting and consolidating these repeated cases can greatly diminish the number of distinct inner products solved in the linear system. However, it can become rather complex to manage an efficient system to classify scaling functions according to their shape. Although we have done some preliminary work towards classifying inner product cases, we leave the completion of this problem for future work.

3.3 Subdivision Wavelets.

We have established nested linear spaces and an inner product relative to any uniformly convergent primal subdivision rule. We are now in a position to construct wavelets, that is, a set of functions $\Psi^j(\mathbf{x}) = (\psi_1^j(\mathbf{x}), \psi_2^j(\mathbf{x}), \dots)$ 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 [Chui 92a], 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.

3.3.1 The Construction.

Our subdivision wavelet 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 $\Phi^j(\mathbf{x})$ 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, including polyhedral subdivision and the butterfly method, have this property.¹ Given a function $S^{j+1}(\mathbf{x})$ in $V^{j+1}(M^0)$ 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 method 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:

$$\mathcal{N}^{j+1}(\mathbf{x}) = \Psi^j(\mathbf{x}) + \Phi^j(\mathbf{x}) \boldsymbol{\alpha}^j. \quad (3.13)$$

Figure 3.6 is a plot of one such wavelet for the case of polyhedral subdivision. If $S^{j+1}(\mathbf{x})$ 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.

3.3.2 Computation of Wavelets.

The coefficients $\boldsymbol{\alpha}^j$ are solutions to the linear system formed by taking the inner product of each side of Equation 3.13 with $\Phi^j(\mathbf{x})$:

$$\begin{aligned} \langle \Phi^j(\mathbf{x}), \Phi^j(\mathbf{x}) \rangle \boldsymbol{\alpha}^j &= \langle \Phi^j(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle \\ &= (\mathbf{P}^j)^T \langle \Phi^{j+1}(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle \end{aligned} \quad (3.14)$$

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 denoted by $\langle \Phi^j(\mathbf{x}), \Phi^j(\mathbf{x}) \rangle$ is therefore simply \mathbf{P}^j , and the matrix $\langle \Phi^{j+1}(\mathbf{x}), \mathcal{N}^{j+1}(\mathbf{x}) \rangle$ is a submatrix

¹ 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.

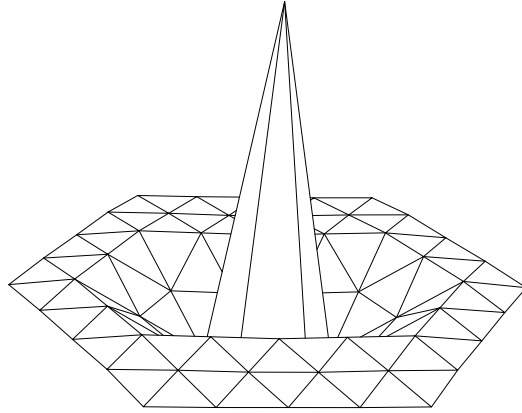


Figure 3.6: A polyhedral wavelet centered on a regular vertex of valence 6.

of \mathbf{I}^{j+1} that consists of the columns that correspond to members of $\mathcal{N}^{j+1}(\mathbf{x})$. The matrix $\boldsymbol{\alpha}^0$ for the example tetrahedron appears in Figure 2.4.

This system of equations may be solved for the coefficients $\boldsymbol{\alpha}^j$. Following Equation 3.13, the values $\boldsymbol{\alpha}^j$ are sufficient to construct the wavelets centered around the vertex at the center of each scaling function in $\mathcal{N}^{j+1}(\mathbf{x})$.

3.3.3 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 3.14 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 [Chui 92a] is able to develop unique locally supported wavelets for B-spline curves. Mallat [Mallat 89] 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 de-

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 3.14 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 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 α^j set to zero. The remaining nonzero coefficients can be found by solving a smaller, local variant of Equation 3.14. By allowing more of the coefficients of α^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 4.7 and 6.2 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.

3.3.4 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

$$\left(\Phi^j(\mathbf{x}) \ \Psi^j(\mathbf{x}) \right) = \Phi^{j+1}(\mathbf{x}) \left(\mathbf{P}^j \ \mathbf{Q}^j \right), \quad (3.15)$$

and the analysis filters are obtained from the inverse relation

$$\begin{pmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{pmatrix} = \left(\mathbf{P}^j \ \mathbf{Q}^j \right)^{-1}. \quad (3.16)$$

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

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

It then follows from Equation 3.13 that the synthesis filters can be written in block form as

$$\left(\mathbf{P}^j \ \mathbf{Q}^j \right) = \begin{pmatrix} \mathbf{O}^j & -\mathbf{O}^j \boldsymbol{\alpha}^j \\ \mathbf{N}^j & \mathbf{1} - \mathbf{N}^j \boldsymbol{\alpha}^j \end{pmatrix}, \quad (3.17)$$

where $\mathbf{1}$ denotes the identity matrix. The analysis filters are obtained from Equation 3.16. (Examples for both are shown for the tetrahedron in Figure 2.4.)

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 3.16 need not be sparse. For interpolating subdivision schemes such as polyhedral subdivision and the G^1 “butterfly” scheme of Dyn *et al.* [Dyn et al. 90], the situation is much improved. Such interpolating schemes have the property that \mathbf{O}^j is the identity matrix. Equation 3.17 in this case is greatly simplified; the resulting filters are

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

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 3.16, 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 3.1 shows what we currently know about the time required to develop reconstruction filters for various subdivision methods.

Table 3.1: *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

$$S^{j+1}(\mathbf{x}) = \sum_i v_i^{j+1} \phi_i^{j+1}(\mathbf{x}) \quad (3.18)$$

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 3.3, and let \mathbf{W}^j denote the corresponding matrix of wavelet coefficients w_i^j . We can rewrite Equation 3.18 in matrix form and substitute the definition of the analysis filters. Thus:

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

therefore,

$$\mathbf{V}^j = \mathbf{A}^j \mathbf{V}^{j+1} \quad \mathbf{W}^j = \mathbf{B}^j \mathbf{V}^{j+1}.$$

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:

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

3.4 Meshes with Boundary Edges.

Subdivision wavelets may be constructed for surfaces whose base mesh has boundaries or holes, but for such surfaces one must compute inner products specially adapted to the boundary neighborhoods. These inner products may be computed using the techniques of Section 3.2.

Subdivision rules capable of producing piecewise smooth surfaces with boundaries or sharp edges are described by Hoppe *et al.* [Hoppe et al. 94, Hoppe 94].

3.5 Construction of Duals.

From Section 2.2.2, we recall that the scaling function duals $\tilde{\phi}_i^j(\mathbf{x}) \in V^j$ must satisfy the relation

$$\langle \phi_k^j(\mathbf{x}), \tilde{\phi}_i^j(\mathbf{x}) \rangle = \delta_{ik}.$$

We can store the coefficients that represent $\tilde{\phi}_i^j(\mathbf{x})$ in the i -th column of the matrix β^j :

$$\tilde{\phi}^j(\mathbf{x}) = \Phi^j \beta^j.$$

In general, the entries of β^j may be found by solving the linear system

$$\langle \Phi^j, \Phi^j \rangle \beta^j = \mathbf{1}.$$

Because $\langle \Phi^j, \Phi^j \rangle$ is exactly the inner product matrix \mathbf{I}^j , the dual matrix β^j is simply $(\mathbf{I}^j)^{-1}$.

The duals are useful for approximating an arbitrary function $f(\mathbf{x})$ as a linear combination of level j scaling functions. Once the appropriate points are chosen for sampling $f(\mathbf{x})$, Equation 2.1 may be used to determine the scaling function coefficients at the sample points. The problem of where to sample is discussed in Section 4.6.

Equation 2.1 requires evaluating the inner product of $f(\mathbf{x})$ and the dual. For a general $f(\mathbf{x})$, the only way we know to compute this value is by numerical integration. When $f(\mathbf{x})$ is a subdivision surface, it may be that the inner product is computable by a more satisfying method, perhaps using methods like those presented in Section 3.2. Another problem is that because the inverse of the inner product matrix is dense, the duals are generally globally supported, meaning that the scaling function coefficients cannot be computed in linear time. We leave both these practical concerns for future work.

Chapter 4

WAVELET COMPRESSION OF SURFACES

Multiresolution analysis is widely used for data compression applications. Mallat [Mallat 89] and DeVore *et al.* [DeVore et al. 92], among others, use wavelet techniques for efficient image compression. Finkelstein and Salesin [Finkelstein & Salesin 94] develop a wavelet-based method for approximating B-spline curves within an L^∞ tolerance. Other examples of wavelet-based compression techniques abound [Chui & Shi 92, Lucier 92, Mallat & Hwang 92, Berman et al. 94].

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 broken down into a coarse-level approximation v^0 together with wavelet coefficients w^0, \dots, 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, \dots, 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 chapter, we will examine the details of extending this general technique to the compression of complex surface data. In Section 4.2 we discuss an appropriate data structure for compressing a large surface representation. The common special case of polyhedral decomposition is treated in Section 4.3. Next, Section 4.4 discusses various rules for selecting wavelet coefficients. The intricacies of the reconstruction algorithm are presented in Section 4.5, and Section 4.6 discusses the preprocessing step sometimes necessary for converting general input into a form appropriate for compression. Finally, Section 4.7 illustrates the results of compression for two complex polyhedral data sets.

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

4.1 *Auxiliary Libraries Required for Compression.*

The techniques of Chapter 3 allow the compression of a complex surface in linear time. A sparse matrix package is useful for implementing vector-matrix multiplication in time linear in the input size, although, as is shown in Section 4.3, it is not strictly necessary for compression of polyhedral surfaces. A linear equation solver is also needed to find the wavelet coefficients α^j of Equation 3.14 for each level j . Because each system solved only involves a local neighborhood around a wavelet, it is possible to set up the linear systems in such a way as to avoid the need for a general sparse matrix solver. All examples in this dissertation were generated with the *spofa()* and *sposl()* routines from LINPACK [Dongarra et al. 79].

4.2 *Efficient Data Structures for Compression.*

When compressing large data sets, memory consumption is an important consideration. Data sets such as those depicted in Section 4.7.2 contain as input over one million color triplets. A data set of this size requires care to be implemented on a standard workstation. Using the techniques described in this section, it is possible to process the enormous Earth data set using an SGI Indigo² Extreme with 128 MB of memory.

Another important consideration is the time required for operations on the subdivision mesh. The subdivision wavelet process involves scaling function information at a range of levels, from the finest functions at input level j to the coarsest at level 0. For an arbitrary vertex at some level in the hierarchy, it is important to efficiently determine the neighbors, parents, level, valence, and a variety of other information. The choice of data structure crucially affects the speed at which this information may be accessed.

When choosing data structures, a naive first choice might be to store the complete data using a standard winged-edge structure [Weiler 85], but that approach is too expensive

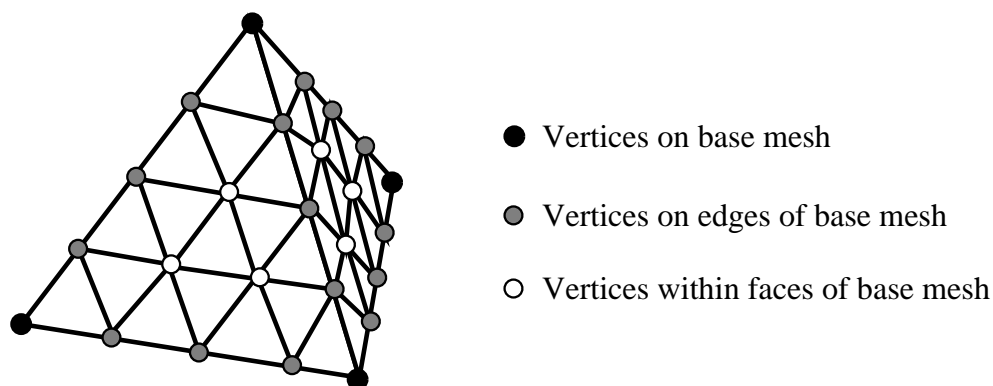


Figure 4.1: *The subdivided tetrahedron, with vertices classified according to their relative location on the base mesh.*

to comprehensively maintain for larger input sets. For example, the Earth data set has 2,097,152 faces, 1,048,578 vertices, and 3,145,728 edges — far too many to fit into even 128 MB if implemented with a full winged-edge data structure. Additionally, a comprehensive mesh representation does not easily allow speedy traversal up and down the subdivision hierarchy.

4.2.1 Representing the Subdivided Mesh.

The great degree of regularity found in a subdivided mesh can be exploited to yield a much more compact and efficient representation than is possible with an arbitrary mesh. Therefore, we use an explicit mesh structure only for the base mesh, which can be of arbitrary connectivity. The subdivision points are then stored on features of the base mesh. This representation combines flexibility of topology with an efficient representation at each level of subdivision.

All vertices on the subdivision mesh may be classified and stored according to their location with respect to the base mesh. A small subset of subdivision vertices corresponds directly to the vertices on the underlying base mesh. A larger group of vertices appears along the edges connecting vertices in the original base mesh. A still larger group consists of vertices on the subdivided mesh lying wholly within the faces of the original base mesh. See Figure 4.1.

It is possible to store all vertices from a subdivided mesh such that they may be accessed

in constant time from a feature on the base mesh. The handful of vertices associated with the base mesh vertices are stored one per base mesh vertex. The vertices along the base mesh edges may be stored in a simple array associated with each edge. The vertices at the interiors of the faces may also be stored in an array attached to the face, then ordered and indexed (using an indexing method such as that used to order control points on a Bézier patch [Farin 93]) in constant time by means of the unique encoding that will be described in Section 4.2.2.

Using this storage method, all vertices at every level are accessible in virtually constant time; the only possible nonconstant operation is that of finding the relevant feature on the typically small base mesh. This operation can be sped up to essentially constant time through the use of a hash table.

This representation is compact and efficient, regardless of the subdivision depth. The space overhead is negligible, requiring just a pointer to the array of data connected with each feature of the base mesh. All other information for a vertex, such as its depth, neighbors, and parents may be inferred from the unique encoding assigned to each vertex on the subdivided mesh.

This structure provides a framework on which to store different kinds of data. For greatest space efficiency, it is useful to run filter bank decomposition as an in-place operation over this structure. The original input (in the form of scaling function coefficients at the finest level) may be read in and stored on the finest-level vertices. As the decomposition process continues, most of this fine-level data may be replaced with the corresponding wavelet coefficient that represents the detail necessary to recreate the original fine-level point. The vertices corresponding to the coarser-level points at each level are not used within their level, but are copied up to the next coarser level in preparation for the next stage of the filter bank.

While economical, this representation is not the most compact possible. A wavelet representation should be capable of storing n data points using exactly n scaling function and wavelet coefficients. The above representation is within a constant factor of the maximum, however – it builds slots for $\frac{4n}{3}$ points. The wasted space goes to the allocation for coarser-level vertices within each level. Because these vertices are duplicated by information at their respective coarser level, such storage is not strictly needed.

4.2.2 Vertex Encodings.

An important key to using this structure is that every vertex be uniquely encoded at every level of the subdivision. We use a form of barycentric indexing that is similar to that used for encoding control points in a triangular Bézier patch [Farin 93].

Each vertex at level l of the subdivision is given a unique encoding \mathbf{abcijk} , where $i + j + k \equiv 2^l$. This encoding says that the vertex has barycentric position ijk within face \mathbf{abc} of the base mesh: its position in the face is weighted i times by vertex \mathbf{a} , j times by vertex \mathbf{b} , and k times by vertex \mathbf{c} . If any of i , j , or k is of weight zero, then the corresponding vertex is irrelevant, and can be represented by any appropriate placeholder.

For uniqueness, it is important to ensure that: (1) any vertex within a face has a unique encoding; (2) vertices on an edge shared by two faces have identical encodings within either face; and (3) vertices on the base mesh vertices are given the same encoding — regardless of the face from which they are accessed. This is trivially obtained by sorting the vertex names within the encoding (with the provision that a placeholder vertex sorts to the lowest value). Thus, all vertices on the edge shared by face \mathbf{abc} and a face \mathbf{acd} will be of the form $\mathbf{xac0ij}$, where \mathbf{x} is a placeholder for the vertex with zero weight. A more complete example illustrating the vertex encodings within two neighboring faces at subdivision level 2 is shown in Figure 4.2.

Using simple algorithms, it is possible to determine important information about a vertex from its encoding, without the need for additional storage. First, the encoding gives the subdivision level: at level l , the sum of i , j , and k is always 2^l . Next, it tells the type of vertex: if two of i , j , and k are zero, then the vertex must be associated with a vertex on the base mesh. Similarly, if only one is zero, then it must be along an edge in the base mesh. Otherwise, it is a vertex on a patch interior. It is also possible to determine other information about the vertex, such as whether it is newly introduced at this level (in which case i , j , or k is odd), and what its same-level neighbors or coarser-level parents are.

The complete set of encodings is not explicitly stored — individual encodings are used only to quickly obtain information about a vertex. Small collections of vertex encodings are temporarily generated during implementation, such as when a neighborhood of vertices around a wavelet is modified during reconstruction. Additionally, it is useful to write iterators that make use of the encodings to cycle one by one through all vertices at a particular

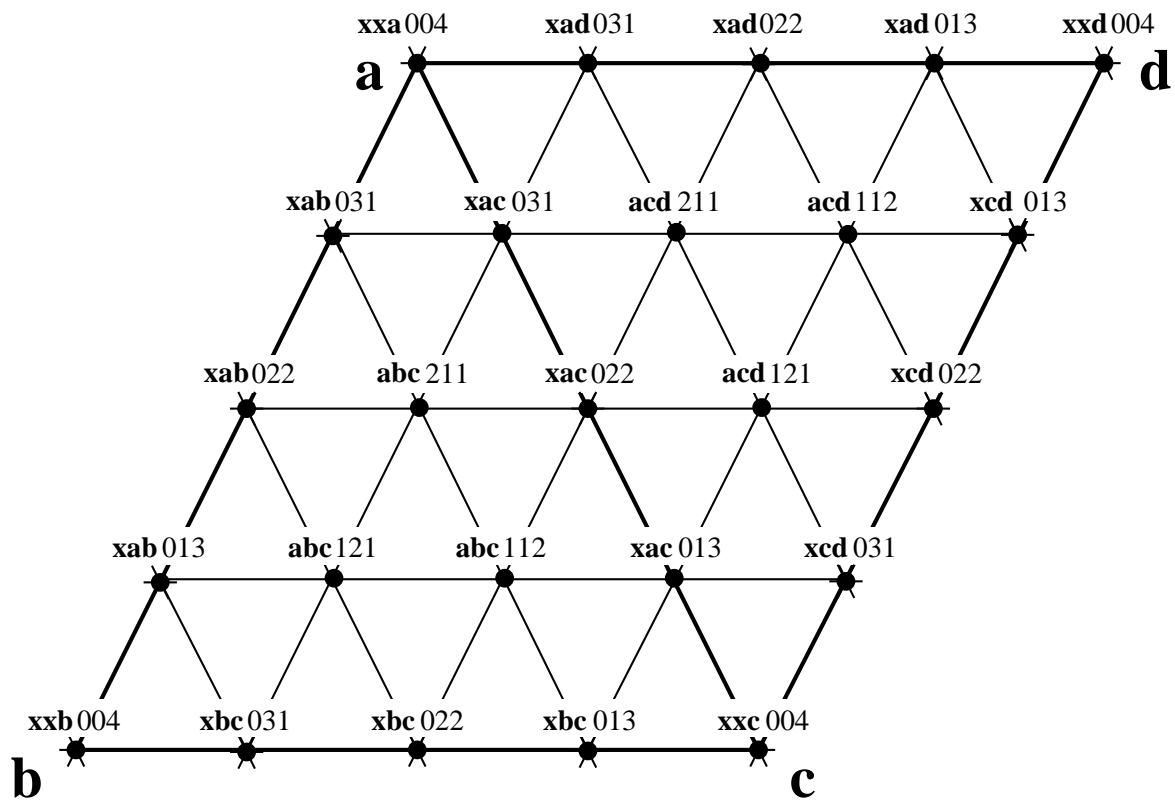


Figure 4.2: Encodings for vertices in faces abc and acd at subdivision level 2.

level.

4.2.3 *Input Representation.*

The input we read is a representation of the coarse-level base mesh, together with the collection of points living at subdivision level j . Each fine-level information tuple is read in along with its identifying vertex encoding. This provides enough information to store the scaling function coefficients at the finest level j , in preparation for the filter bank process.

4.3 *Polyhedral Implementation.*

Many applications of subdivision wavelets involve the special case of piecewise linear subdivision — an example is polyhedral compression. The techniques outlined in Chapter 3 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 may be gleaned from the matrix \mathbf{B} that represents the filter for computing the wavelet coefficients of the polyhedral surface in Figure 2.4, the wavelet coefficient \mathbf{w} associated with a vertex \mathbf{v} (where \mathbf{v} has been subdivided from the coarser-level vertices \mathbf{p} and \mathbf{q}) may be derived by the simple rule:

$$\mathbf{w} = \mathbf{v} - \frac{1}{2}(\mathbf{p} - \mathbf{q}). \quad (4.1)$$

This equation is shown geometrically in Figure 4.3. It leads to a very simple algorithm for computing the wavelet coefficients — without the need for implementing sparse matrix multiplication.

Determining all level $j - 1$ wavelet coefficients w^{j-1} using Equation 4.1 is the first half of the level j to level $j - 1$ decomposition process. It is still necessary to derive the filter

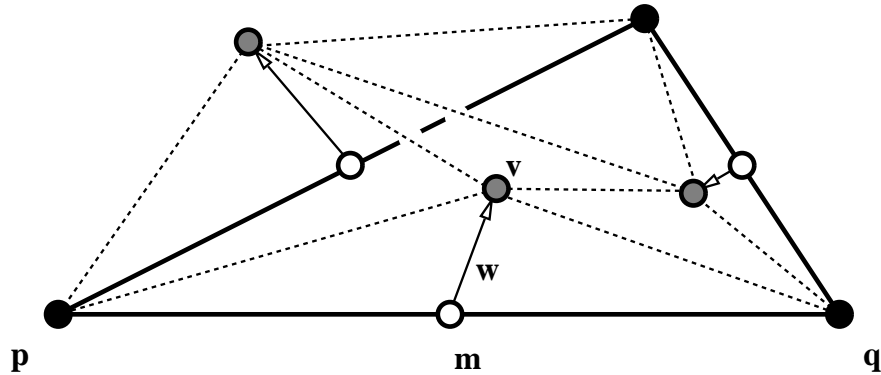


Figure 4.3: *Determining the wavelet coefficient \mathbf{w} around vertex \mathbf{v} , with parents \mathbf{p} and \mathbf{q} . The midpoint between \mathbf{p} and \mathbf{q} is \mathbf{m} , and the resulting wavelet coefficient is the difference \mathbf{w} between \mathbf{m} and \mathbf{v} .*

\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 Chapter 3, but specially adapted for linear subdivision. In particular, the inner product mask for the linear case is shown in Figure 3.3. This mask is especially easy to determine for polyhedra, because the central value is merely the valence of the vertex.

4.4 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.

4.4.1 Normalization.

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 dissertation use selection based upon this L^2 normalization of the wavelet coefficients.

In some applications, it may be useful to use a different normalization that creates a better visual impression than least-squares approximation. For example, in image compression, the user is more likely to notice fine detail in a picture than a wide-ranging brightness change with greater least-squares error. Similarly, coefficients used to approximate a detailed geometric model may be better spent on recreating sharp corners and fine-level texture than on closely approximating less noticeable gross shape. Exploring other normalization strategies may be an interesting area for future research.

4.4.2 Selection Strategies.

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

- *Threshold testing.* Perhaps the simplest selection technique, and one which can be implemented in linear time in the number of coefficients, 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 4.7. 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 with their fellow wavelets on the same level. Hence, this property cannot be shown for them. However, they are (close to) orthogonal to wavelets at different levels, and an adequate approximation to least-squares progressive refinement is still possible using this same simple technique.

- *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.

Any subdivision rule that allows the size of the wavelets to be bounded (one such rule is polyhedral subdivision) allows L^∞ reconstruction. When the scaling function generated by the subdivision rule lies in the convex hull of its control points, L^∞ may be implemented by recording at each vertex on the surface the distance of the current approximation from the original function. Starting from the complete input, wavelet coefficients at various positions around the surface may then be progressively removed, until it is no longer possible to remove any without violating the L^∞ condition. The ideal algorithm would remove the maximum number of such vertices that is possible, but such an algorithm is related to bin-packing, a problem known to be NP-hard [Garey & Johnson 79]. One approach, reminiscent of progressive refinement selection, is a greedy algorithm: first sort the coefficients by size, then repeatedly attempt to remove from the surface the wavelet coefficient with the least impact. Such a technique runs in time $O(n \log n)$, because sorting is again required.

- *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.

4.5 *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 a selected subset of these finer-level functions.

4.5.1 *Naive Reconstruction*

In some settings of compression, it may suffice to simply write out the base level approximation together with a list of the selected wavelets and their values. More usually, however, a more explicit construction of the approximation is required. The following naive algorithm builds this approximation A from the base approximation v^0 and the wavelet coefficients w^i at each level from 0 to $j - 1$:

```

procedure NaiveReconstruct ()
  begin
     $A := v^0$ ;
    for ( $i := 0$ ;  $i < (j-1)$ ;  $i++$ )
      begin
        Subdivide  $A$  according to the appropriate subdivision rule;
        ForEach ( $w_j^i \in w^i$ )
          Add  $w_j^i$  back to  $A$ ;
        end;
        Triangulate the result;
      end;
  end;

```

Although the naive algorithm correctly builds an approximation from the reduced set of wavelet coefficients, the resulting approximation contains exactly as many triangles as the original input. This is because the naive algorithm subdivides at every position over the constructed surface at every level, regardless of whether additional detail is ever added to that position.

When compressing polyhedral surfaces, it is usually more convenient to derive an approximation with fewer triangles. For polyhedral compression, we can improve the naive algorithm by only subdividing the surface in regions around the selected wavelets.

We begin our improved reconstruction, as before, with a copy of the base mesh. Suppose we wish to add to our reconstruction a scaled wavelet ψ_i^j centered around a vertex \mathbf{v} at level j . The construction presented in Section 3.3.1 builds ψ_i^j as a sum of basis functions at levels j and $j - 1$. Adding back ψ_i^j therefore still requires modifying the relevant scaling function coefficients at these levels.

4.5.2 Rebuilding Parent Structure.

A complication arises when the scaling functions that must be modified have not yet been represented in the reconstruction. For example, the first wavelet added to our reconstruction must involve the modification of at least one finer-level scaling function that has not yet been built into the representation. In these cases, it is necessary to first build in the coefficients at the affected finer-level scaling functions by refining the underlying coarser-level “parent” function. That is, the coefficients at the finer-level functions are derived from

subdivision of the coefficients at the parent. If the value of the coefficient at the parent has not yet been computed, it must be recursively expanded in the same way. This expansion is implemented using a recursive *Expand()* procedure. The progress of this procedure through three subdivision levels is indicated in Figure 4.4.

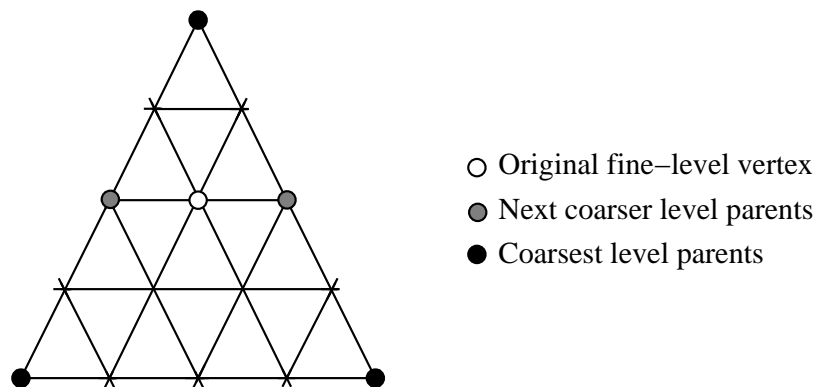


Figure 4.4: *Progress of the recursive $Expand()$ procedure for piecewise linear subdivision. Beginning with the original fine-level vertex (shown in white), all resulting expanded vertices are shown here.*

The *Expand()* procedure is written in the following pseudocode:

procedure Expand (**vertex** v)

begin

if (v is marked as expanded)

return;

if (v is a coarser-level vertex living on a finer level)

begin

vup := coarser-level version of v;

Expand(vup);

mark v as expanded;

return;

end;

(Collect the subdivision rule for v:)

vp := array of parents of v;

vw := array of weights of vp;

subsize := number of elements in vp;

(The weights vw encode the subdivision rule that gives v as an affine combination of its parents vp)

for (i = 0; i < subsize; i++)

begin

(First, recursively expand the parents, if necessary)

Expand(vp[i]);

(Then derive v from them by the subdivision rule)

position at v += vw[i] * vp[i];

end;

mark v as expanded;

end;

4.5.3 Triangulation.

The above reconstruction is sufficient 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.

The approximation generated above may be triangulated with a top-down recursive procedure that is initially called for each triangle in the base mesh. As illustrated in Figure 4.5, the procedure tests the finer-level *subvertices* on each edge of its input triangle. If any of these subvertices were expanded during the *Expand()* routine, the *Triangulate()* procedure may be called on each of the four subtriangles.

When a triangle with an expanded subvertex is split into four subtriangles, it is not always true that all three subvertices that are to be connected by triangulation have already been expanded to contain position information. In polyhedral reconstruction, the position of the subvertex on the edge may be derived from the midpoint of the two vertices on either end of the edge.

The version of *Triangulate()* given here splits each triangle with an expanded subvertex into four subtriangles, even if only a single subvertex is expanded. Although it is possible to split into fewer subtriangles in these cases, the above procedure does not introduce edges where they do not exist in the input. Such a triangulation does not always produce the minimal number of faces possible, but it does provide coherence for smoothly varying between different levels of detail (see Section 5.4). Triangulations with fewer faces may be produced if this property is sacrificed.

Additionally, the above algorithm does not create a strict mesh, because it creates T-vertices at the midpoints of certain edges. The algorithm may be easily modified to avoid T-vertices, and to produce a true mesh. However, there is a tradeoff. Producing a compressed proper mesh is only possible by introducing new edges that do not exist in the input.

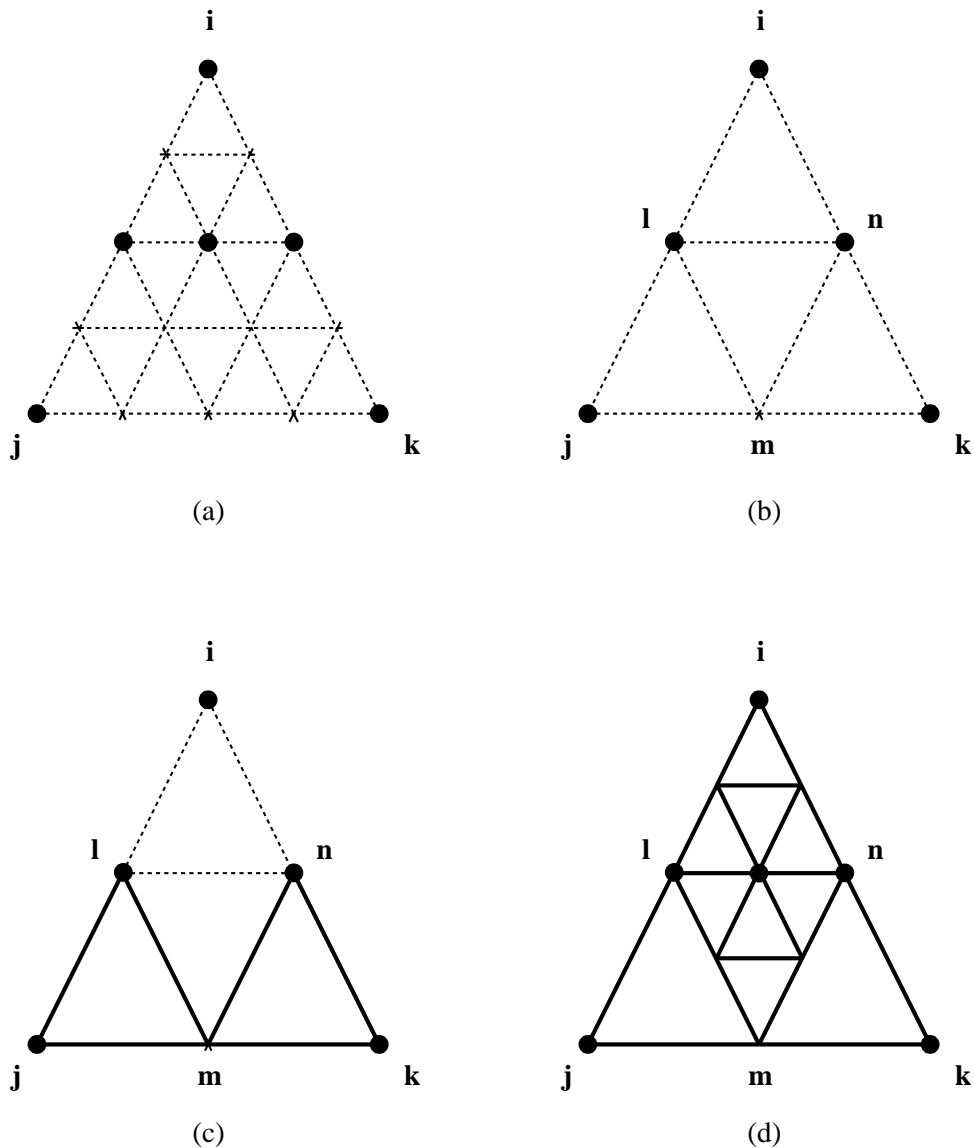


Figure 4.5: Progress of the recursive `Triangulate()` procedure. (a) The input neighborhood, with the expanded vertices in black. (b) The recursive procedure begins with the triangles vertices i , j , and k . The subvertices are l , m , and n . (c) Because the subvertices l and n are expanded, ijk is split, and `Triangulate()` is recursively called on the four subtriangles. Subtriangles jlm and kmn may be trivially output, because they have no subvertices on their edges. The position of m is set to be the midpoint of j and k . (d) The final triangulation. The presence of a subvertex on the edge between l and n means that ilm and lkn are further subdivided.

The *Triangulate()* procedure described here may be implemented using the following pseudocode:

```

procedure Triangulate (vertex vi, vj, vk; integer depth)
  begin
    if (depth = maximum subdivision level)
      begin
        WriteTriangle(vi, vj, vk);
      return;
    end;

    (vl, vm, vn) := subvertices of (vi, vj, vk);
    if (none of vl, vm, or vn are marked as expanded)
      begin
        WriteTriangle(vi, vj, vk);
      return;
    end;

    else if (only vl is marked as expanded)
      begin
        Expand(vm);
        Expand(vn);
      end;
    :
    (Similar process for other cases of exactly one subvertex expanded)
    :
    else if (only vl and vm are marked as expanded)
      Expand(vn);
    :
    (Similar process for other cases of exactly two subvertices expanded)
    :
    (Any case of 1, 2, or 3 subvertices marked as expanded reaches this point)

    (Recursively triangulate each of the 4 subtriangles)
    Triangulate(vi, vl, vm, depth + 1);
    Triangulate(vl, vj, vn, depth + 1);
    Triangulate(vm, vn, vk, depth + 1);
    Triangulate(vl, vn, vm, depth + 1);
  end;

```

Triangulate() is only appropriate for polyhedral reconstruction. Reconstruction of general subdivision surfaces should use another method for creating a surface from the expanded points. The tangent-plane smooth examples in Chapter 6 were produced with naive reconstruction.

4.5.4 Asymptotic Complexity.

Compression of an input of size n based on reconstruction from c coefficients always requires at least $O(c)$ time, because each coefficient must be added to the base mesh. The time required to add a single coefficient varies. In most cases, only a constant-size local neighborhood around the coefficient's vertex needs to be modified. In some cases, however, it may be necessary to reconstruct the underlying parent structure first. These cases require time proportional to the subdivision depth j , where $j \approx \log_4 n$. These cases are rare, however, because the locality that usually occurs in real-world data implies that detail coefficients tend to be clustered together. This means that a coefficient added later in the process is highly likely to be added to a region whose parent structure has already been computed.

This suggests that reconstruction time is at worst $O(j c)$, but more usually much closer to $O(c)$. These estimates are borne out by the empirical evidence of run times for reconstruction that is presented in Section 4.7.3.

4.6 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.

4.6.1 *The General Conversion Problem.*

Given some input representation, such as a polyhedral surface or a collection of points, the preprocessing phase must first turn the input into a function $f(\mathbf{x})$ parametrized over some simple mesh M^0 . This involves associating each point with a position on M^0 .

Once $f(\mathbf{x})$ has been determined, the next step is to project $f(\mathbf{x})$ into $V^j(M^0)$, creating an approximation $\tilde{f}(\mathbf{x})$. More precisely,

$$f(\mathbf{x}) \approx \Phi^j(\mathbf{x})\mathbf{V}^j = \tilde{f}(\mathbf{x}),$$

where $\Phi^j(\mathbf{x})$ spans the space of functions $V^j(M^0)$, and \mathbf{V}^j is the set of scaling function coefficients that describe the approximation $\tilde{f}(\mathbf{x})$. The \mathbf{V}^j may be determined using the scaling function duals $\tilde{\phi}_i^j(\mathbf{x})$ from Section 3.5.

For some inputs, building $f(\mathbf{x})$ is straightforward. When the input is cylindrical data from a topological sphere, it may be approximated with values at vertices of a subdivided octahedron, as is shown in the next section.

Developing an algorithm to build $\tilde{f}(\mathbf{x})$ from a general input representation is an important area of research that is currently under investigation. The present lack of such a preprocessing algorithm for the general case in certain applications should not detract from the theoretical importance of subdivision wavelets, nor from their practical impact in applications where the construction of such input is straightforward.

4.6.2 *A Sphere-to-Octahedron Conversion Algorithm.*

For specific cases, the preprocessing is not difficult. In this section, we describe an algorithm to approximate spherical or cylindrical range data with data on a subdivided octahedron. The algorithm may be extended to convert cylindrically scanned data from any object whose topology is that of a sphere.

To illustrate, the original data for Figure 4.7 (page 69) was taken from Cyberware laser range data arranged in rings of points sampled from a bust, and Figure 4.9 comes from data sampled at latitude and longitude positions around the globe. In each case, the data may be segmented into octants corresponding to each face of the base octahedron.

The subdivision depth j controls how closely the original function $f(\mathbf{x})$ may be approximated. The Spock and Earth data sets were approximated using subdivided octahedra of depths 6 and 9, respectively. At each increased level of subdivision, the maximum edge length between vertices in M^j is only half that of M^{j-1} . As j increases, the approximation $\tilde{f}(\mathbf{x})$ quickly converges to $f(\mathbf{x})$.

The process of approximating a spherical function with vertices on an octahedron subdivided j times requires two steps. First, one must assign latitude and longitude to points on the subdivided octahedron M^j . Next, use the latitude and longitude to look up the value of $f(\mathbf{x})$ at that position. This value is used as the scaling function coefficient.

It is fairly straightforward to map points on the subdivided octahedron to spherical latitude and longitude coordinates. The base octahedron, representing the mesh M^0 at subdivision level 0, has six vertices. Two of these map directly to the north and south poles (at latitudes 90° and -90° , respectively). The other four vertices are evenly spread around the equator (latitude 0°), at longitudes 0° , 90° , 180° , and 270° . The spherical coordinates of a vertex on a subdivided mesh may be determined recursively; for example, a vertex of M^1 has latitude and longitude values halfway between the values at two vertices of M^0 . Alternatively, the points may instead be computed by direct geometric projection of the point on the octahedron onto a latitude and longitude position on the sphere.

If the spherical data is originally provided as a C^0 continuous function, the only remaining step is to cycle through all vertex positions in \mathbf{V}^j , derive their coordinates on the sphere, and assign them the value of $f(\mathbf{x})$ at these coordinates.

If the input is not continuous, but is only provided as a discrete set of points on the sphere (as in Figure 4.9), it is necessary to first construct a C^0 bilinear interpolant [Farin 93] parametrized by latitude and longitude.

4.7 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 using the technique of Section 4.6.2.

4.7.1 Geometric Data.

The input for the first example (shown in Figure 4.6(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

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

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 3.3.4 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 Indigo² 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 4.4.2). The models in Figure 4.7(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 unoptimized implementation of the entire decomposition and reconstruction process that produces Figure 4.7(d) runs in about 53 seconds from input to output.

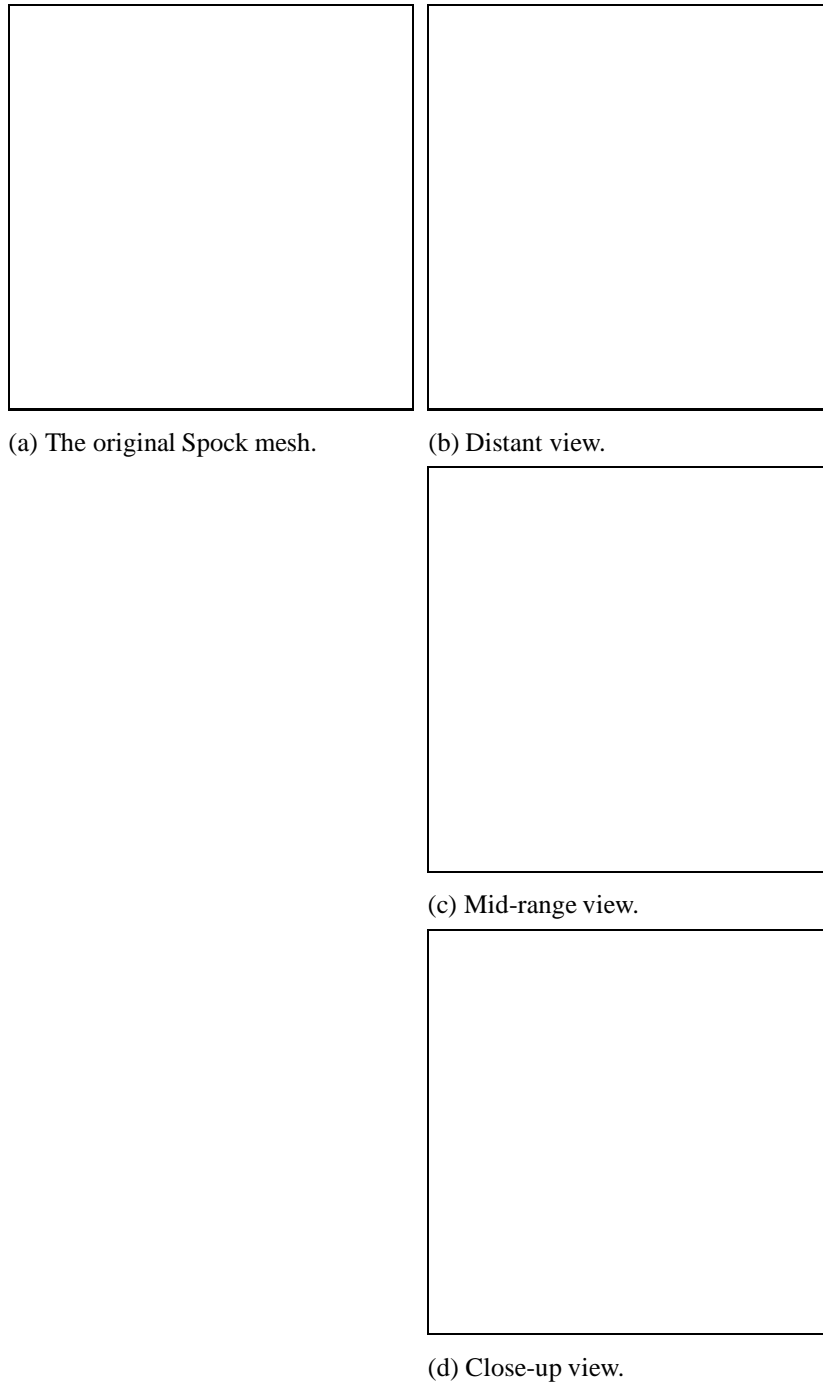


Figure 4.6: *The full-resolution Spock polyhedron. Left: the full mesh (16,386 points, 32,768 triangles). Right: Gouraud-shaded views of the full mesh at various distances.*

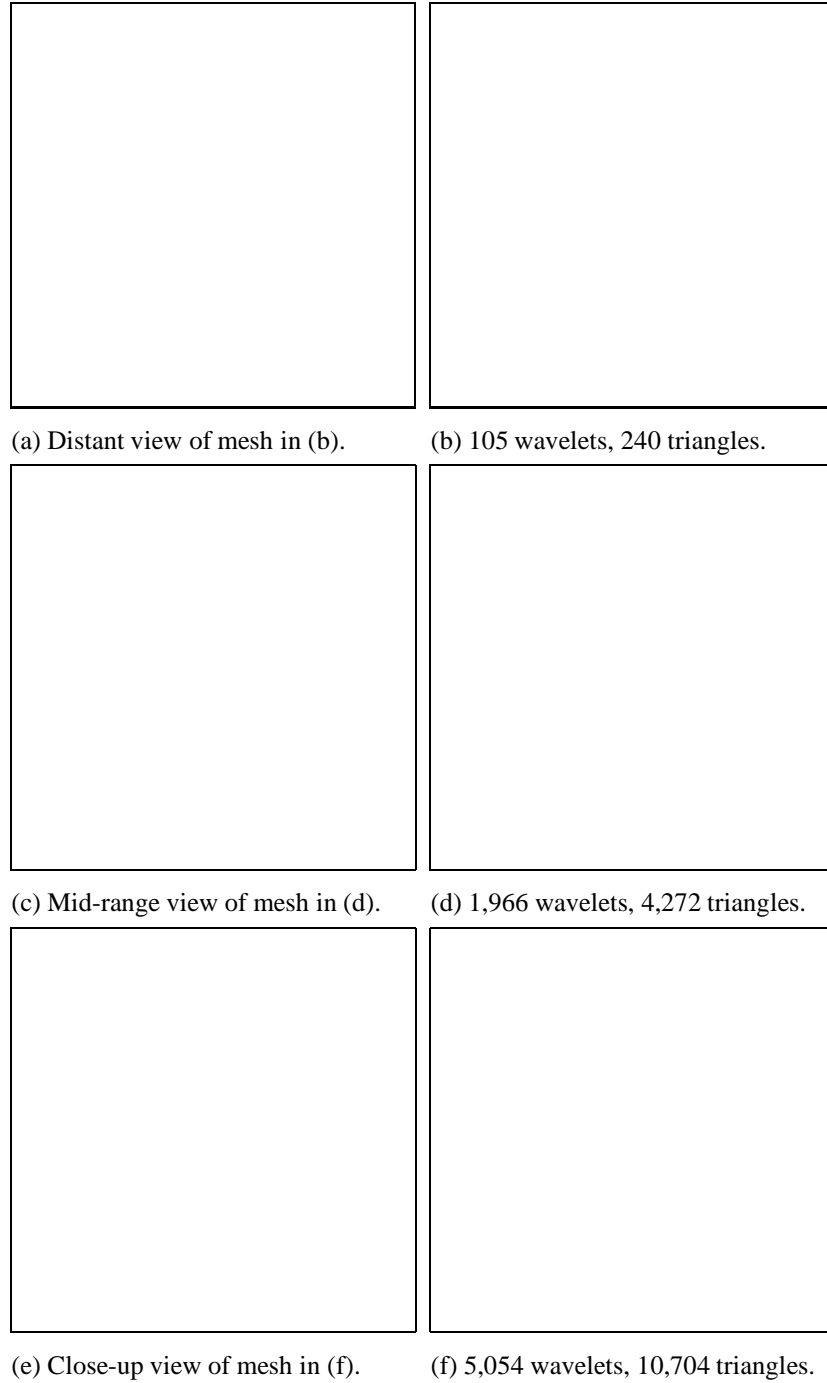


Figure 4.7: *Wavelet approximations of the Spock polyhedron. Left: Gouraud-shaded wavelet approximations. Right: Flat-shaded close-ups showing structure of approximations.*

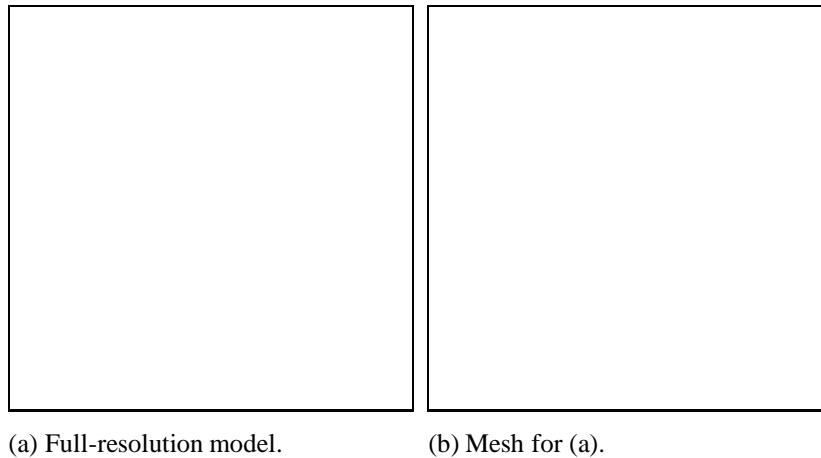


Figure 4.8: *Close-up of Earth data before compression*

Figure 4.7 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 4.6(a). Views of the full-resolution model from various distances are shown in the rest of Figure 4.6. 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 4.7 may instead be used without significantly degrading image quality. Further discussion of level-of-detail applications is given in Section 5.2.

4.7.2 *Color Data.*

Subdivision wavelets may be applied to more general functions over surfaces than geometric data. Figure 4.9 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 Indigo² Extreme with 128 MB, and is therefore not shown in its entirety in Figure 4.9. Instead, Figure 4.8 shows a close-up of a region that is compressed in Figure 4.9. An appreciation for the density of the data can be obtained from Figure 4.8(b), where even at close range the mesh lines of the original uncompressed data are so dense that the image appears almost

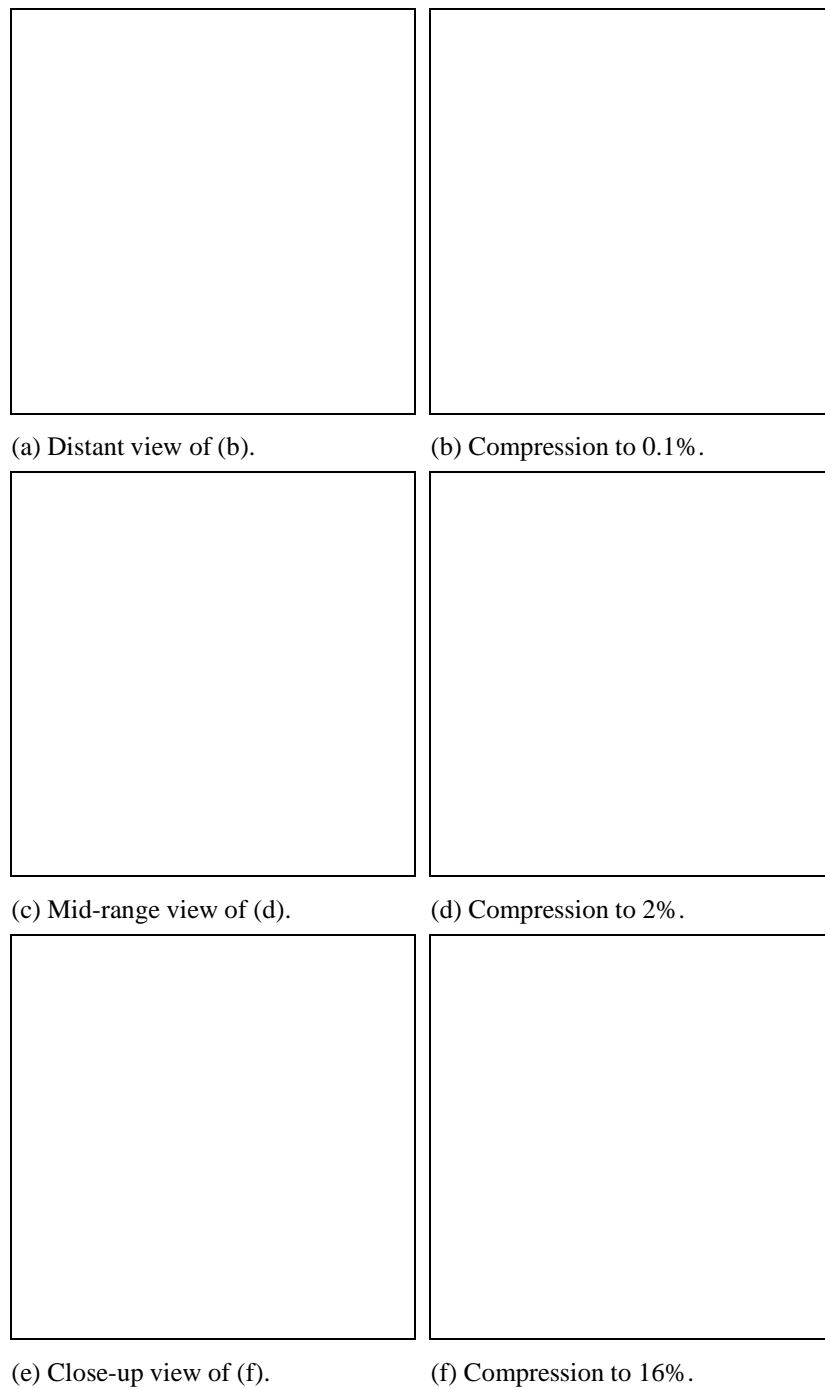


Figure 4.9: *Approximating color as a function over the sphere.*

completely black.

The approximations shown in Figure 4.9(a)-(f) were produced using subdivision wavelet compression. Figure 4.9(a) shows a distant view of the Earth using an approximation of only 0.1% [the mesh is shown in (b)]. Likewise, Figures 4.9(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 4.8(a). A comparison of the meshes shown in Figure 4.8(b) and Figure 4.9(f) reveals the striking degree of compression achieved in this case.

4.7.3 Run Times.

Example run times for the Spock and Earth data sets appear Tables 4.1 and 4.2. All times reflect compression done on an SGI Indigo² Extreme with a 100 MHz MIPS R4000 CPU and 128 MB of physical memory.

The first two lines of Table 4.1 give information about the size of the Spock input. *Decomposition time* gives the filter bank time needed for any compression of the Spock data, and *Selection time* is the time required to select wavelet coefficients by thresholding.

Table 4.2 is broken down to compare side by side reconstruction times for different coefficient thresholds. The thresholds chosen were used to generate the compressions shown in Figure 4.7(b), (d), and (f). In the table, *Coefficients selected* tells how many coefficients were selected using the indicated coefficient threshold. Information about the size of each reconstruction is given in *Triangles in reconstruction* and *Fraction of original*. *Reconstruction time* includes time for adding back wavelets and recursive expansion, but not for triangulation. Reconstruction and decomposition times are given separately. When separate compressions of the same data set are run, decomposition needs to be run only once, but selection and reconstruction need to be run for each compression. The times for coefficient selection and for triangulation are much less than for decomposition and reconstruction; their contribution is included in *Total processing time*, which gives the beginning-to-end time for a single decomposition and reconstruction session. This value is the sum of decomposition, selection, and reconstruction time, plus all additional operations, including memory management and disk I/O. (Disk I/O is particularly time-consuming for the file of Earth input, which takes up more than 51 MB of ASCII data before compression.)

The run times shown in all tables reflect polyhedral compression with caching of values for the regular case, but without full precomputation (see Section 5.6). Because decomposition and reconstruction require repeated solution of linear systems while the wavelets are computed, full precomputation of α^j for each level j is expected to significantly decrease the run times given in the tables.

Tables 4.3 and 4.4 give information about the Earth data set, using the same format as the Spock tables.

Table 4.1: *Statistics common to all compressions of the Spock data. Time is in seconds.*

INPUT VERTICES	16,386
INPUT TRIANGLES	32,768
SUBDIVISION DEPTH	6
DECOMPOSITION TIME	14
SELECTION TIME	1

Table 4.2: *Spock compression results, organized by coefficient threshold. Time is in seconds.*

FIGURE	4.7(b)	4.7(d)	4.7(f)
COEFFICIENT THRESHOLD	2.0	0.06	0.017
COEFFICIENTS SELECTED	107	1,966	5,054
TRIANGLES IN RECONSTRUCTION	240	4,272	10,704
FRACTION OF ORIGINAL	0.007	0.13	0.33
RECONSTRUCTION TIME	4	29	63
TOTAL PROCESSING TIME	25	53	87

Table 4.3: *Statistics common to all compressions of the Earth data. Time is in seconds.*

INPUT VERTICES	1,048,578
INPUT TRIANGLES	2,097,152
SUBDIVISION DEPTH	9
DECOMPOSITION TIME	588
SELECTION TIME	50

Table 4.4: *Earth compression results, organized by coefficient threshold. Time is in seconds.*

FIGURE	4.9(b)	4.9(d)	4.9(f)
COEFFICIENT THRESHOLD	0.02	0.002	0.0005
COEFFICIENTS SELECTED	741	15,101	138,321
TRIANGLES IN RECONSTRUCTION	1,968	39,408	328,058
FRACTION OF ORIGINAL	0.001	0.019	0.156
RECONSTRUCTION TIME	75	310	1,230
TOTAL PROCESSING TIME	1,315	1,754	2,904

Chapter 5

INTERACTIVE VIEWING

A high-quality animation typically includes highly complex and detailed models, each made of thousands or millions of polygons. Such detail may be necessary for viewing the models close up, but is extraneous for other views. This extraneous detail comes with a price: the time for rendering a single frame of the animation increases with the number of polygons that must be processed. This cost is particularly acute in animation due to the large number of frames that must be computed. At 30 frames a second, 1800 separate frames must be rendered for each minute of animation.

In this chapter, I describe how subdivision wavelets may be applied to two important interactive viewing problems. Section 5.1 discusses the use of wavelet approximations for animation previewing. The rest of the chapter discusses automatically controlling the level of detail presented to a user in an animated view of a static model. This process is outlined in Section 5.2.

5.1 Animation Previewing.

The process of designing a computer animation can be long and tedious. The designer typically goes through many iterations, including changing the model, viewing an animation of the result, spotting errors, incorporating new ideas, and reiterating. The high-quality models used for production-quality animation typically have considerable detail. By providing a low-quality approximation of the model that is suitable for previewing, wavelet approximation can greatly speed up the design process.

Because high quality is not a great concern for animations at the design stage, it is possible to preview in real time even the most complex models with only a rough approximation. If the animator still wishes to view a small area of the model with high detail, special attention may be paid to that region using selection by location (Section 4.4.2). The automatic

level-of-detail techniques presented in the rest of this chapter, while potentially useful for previewing, are not strictly necessary for the limited quality demands of animation previewing.

5.2 Automatic Level-of-Detail Control.

In an interactive view such as an animation or a view within a virtual environment, a user's view of a model changes over time. When a complex model is very far away, it is highly inefficient to render every detail; instead, it should be possible to capture only the essential qualities of the model that are important to the viewer. As the object gets closer, more detail may be added. This process is known as *level-of-detail* control.

Level of detail has been used in flight simulators for many years [Newhard & Nicol 84]. More recently, Funkhouser and Séquin [Funkhouser & Séquin 93] use level-of-detail control for real-time walkthroughs through architectural models. They use simplified models for objects that are distant, and detailed models for close-up views. However, their progression of models at various levels of detail is not automatically generated from a single detailed representation. Instead, each model of different detail must be separately created.

We will now address techniques for automatically implementing level of detail, based on subdivision wavelets.

The basic idea is rather simple. A threshold $\epsilon(0)$ is selected based on the viewer's distance to the model. The threshold $\epsilon(0)$ is then used to construct an approximation $A(0)$, as described in Section 4.4.2. A method for selecting an appropriate $\epsilon(0)$, based on the view distance, is described in Section 5.3.

As the object moves closer, a new, smaller threshold $\epsilon(1)$ is chosen, and a new, more detailed approximation $A(1)$ must be constructed. At this point, a complication is introduced: the close approximation $A(1)$ has details that are absent on $A(0)$. If $A(1)$ were to suddenly replace $A(0)$ in the view, the added features would discontinuously pop into place, making the substitution noticeable by the viewer.

The popping caused by the sudden addition of features can be avoided by smoothly blending between the approximations $A(0)$ and $A(1)$. Section 5.4 explains a simple technique to implement smooth blending of approximations built using subdivision wavelets.

In an interactive setting, it is important to create the approximations as quickly as pos-

sible. Thus far, we have only seen how to create an approximation by adding back an entire subset of wavelet coefficients to the base mesh. In an animation, successive approximations are likely to have many coefficients in common. In this case, it is possible to speed up each step in a sequence of approximations by only modifying the wavelet coefficients that have changed between steps. See Section 5.5.

Sections 5.6 and 5.7 contain more general methods for speeding up approximations, based on precomputation of values and parallel computation.

5.3 *Choosing a Coefficient Threshold.*

Suppose we use the threshold testing method (see Section 4.4.2) to choose coefficients in an animation. At a large distance d_0 , all but the largest coefficients are unimportant for constructing the approximation, so it is possible to use a large threshold $\epsilon(0)$ for choosing which coefficients to discard. Up close, at distance $d_1 \ll d_0$, the threshold $\epsilon(1) \ll \epsilon(0)$ must be much smaller in order to add back all but the smallest coefficients. But how are these values to be chosen along the range between $\epsilon(0)$ and $\epsilon(1)$?

The absolute size of the threshold ϵ that is appropriate for a given distance depends on many factors: the absolute size of the model being animated, screen resolution, the viewing angle, etc. Hence, it is difficult to give a precise rule that works for all cases. However, once appropriate threshold values $\epsilon(0)$ and $\epsilon(1)$ have been chosen for the near and far extremes of distance, the following heuristic has proven useful for choosing an appropriate threshold $\epsilon(d)$ for distances d , where $(d_0 < d < d_1)$.

We construct $\epsilon(d)$ such that its logarithm varies linearly between $\log(\epsilon(0))$ and $\log(\epsilon(1))$ along the distance between d_0 and d_1 . Formally,

$$\log(\epsilon(d)) = t * \log(\epsilon(1)) + (1 - t) * \log(\epsilon(0)),$$

where

$$t := \frac{d - d_0}{d_1 - d_0}.$$

This heuristic is based on an empirically derived relation between the size of the L^2 normalized coefficients and the visible detail on a reconstructed approximation. We have not yet derived a sound theoretical justification for it. The heuristic has worked well in

animations that smoothly blend between a wide range of approximations at a wide range of distances.

5.4 *Smoothly Blending between Approximations.*

Sudden switching between models of different detail in an animation may produce 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. With this simple technique, the object geometry can smoothly change its appearance as the viewing distance changes.

Consider two multiresolution approximations: $A(0)$ is shown at time t_0 , and a subsequent finer-resolution approximation $A(1)$ is depicted at time t_1 . The set of coefficients C present in $A(1)$ but not in $A(0)$ may be smoothly blended over the period of time between t_0 and t_1 using a linear interpolation of the new wavelet coefficients. More precisely, the approximation $A(t)$ presented at time t , ($t_0 < t \leq t_1$) scales each coefficient c_i of C by

$$\frac{(t - t_0)}{(t_1 - t_0)}.$$

The result is a smooth linear blend between $A(0)$ and $A(1)$. Note that a triangulation of the intermediate approximation $A(t)$ has exactly the connectivity of $A(1)$ along the entire blend; hence, there is a discontinuous jump in the connectivity of the underlying mesh used to represent the geometry as the time moves past t_0 . However, the actual geometric effect of the transition is truly gradual, and can be shown to the viewer free of any noticeable jumps.

Using subdivision wavelets for smooth blending has proven successful in the production of frame-by-frame animations of complex models. In these animations, both color and geometry vary smoothly as the distance from the eye to the model varies.

5.5 *Incremental Reconstruction.*

Let us consider the case of zooming in on an object in an animation. When the threshold rule of Section 5.3 is used, the sequence $\epsilon(t)$ of threshold values decreases monotonically as the distance to the object decreases and more detail is reconstructed. As $\epsilon(t)$ decreases,

there is a corresponding increase in the number of wavelet coefficients used to create an approximation $A(t)$. In other words, the wavelet coefficients $C(t_i)$ included in the approximation $A(t_i)$ at time t_i are always a subset of the wavelet coefficients $C(t_j)$ included in $A(t_j)$, the next approximation in the sequence, when $t_i < t_j$.

Recall from Section 4.5.4 that reconstruction time is essentially linear in the number of wavelet coefficients added. This implies that instead of constructing the approximation $A(t_j)$ from scratch — by adding the entire set of coefficients to the base mesh at each time step t — one may greatly speed the reconstruction at time t_j by adding to $A(t_i)$ only the set $C(t_i, t_j) = C(t_j) - C(t_i)$ of coefficients that are in $C(t_j)$ but not in $C(t_i)$. We call this process *incremental reconstruction*; it decreases the time needed for reconstruction for time steps in the range $[t_i, t_j]$ to a function of $|C(t_i, t_j)|$, down from $|C(t_j)|$.

Incremental reconstruction may be efficiently implemented if all wavelet coefficients have first been sorted by their size. It is then a simple matter to determine $C(t_i, t_j)$ from the sorted array of coefficients. Although this requires a preprocessing time of $O(n \log n)$, where n is the total number of wavelet coefficients, it leads to more efficient processing later, when time is precious.

5.6 Precomputation.

A slow step in reconstruction is the computation of inner products and the repeated solution of linear systems. It is particularly important to reduce this computation for interactive applications. The area-independent formulation of the inner product given in Equation 3.7 allows all shapes that share a base mesh M^0 to share inner product and α^j values for a given subdivision rule. Precomputation can allow a greatly accelerated reconstruction process by removing the need for computing inner products and wavelets during execution time.

There are two main methods of precomputation: *caching* and *full precomputation*. With caching, inner products and wavelets are computed on the fly, but repeated cases are detected and a cached value is returned, which avoids full computation. Full precomputation refers to storing all computationally complex values to disk at some single earlier processing time, and then reading them into the relevant matrices before the time of reconstruction.

5.6.1 *Caching of Wavelet Values.*

As the subdivision process continues, vertex connectivity becomes more and more regular. With local subdivision rules, the few extraordinary points on the base mesh become rare exceptions to the usual regular neighborhoods. A simple method of precomputation that we have implemented is to cache the wavelet values built over vertices in the regular case.

A procedure may be constructed that determines from its encoding whether the wavelet for a vertex matches the regular case. Encodings for wavelets far enough away from the extraordinary points at the corners (the particular distance depends on the size of the chosen wavelet support) always have the same structure, and hence, the same wavelet form. The values for this regular case may be stored after the first solution has been determined, and may be returned trivially for each subsequent instance of the regular case.

Although it is possible to detect and cache other repeated cases in the neighborhood of extraordinary points, the results may not make the additional programming complexity worthwhile.

5.6.2 *Full Precomputation.*

For interactive applications, precomputation should be carried to its extreme for maximum efficiency. All relevant matrices required for decomposition and reconstruction may be read from precomputed values.

In particular, all nonzero inner product values and all possible wavelets (represented by the α^j matrices) may be computed for a specific mesh configuration and stored in a table; when a specific wavelet is required, it may be found in the table. Using precomputation, no linear systems ever need to be solved during the course of an approximation.

Much faster computation results from reading all important matrices from a file before processing the input. For a particular subdivision rule and a particular base mesh, all important matrix values may be precomputed and written to a file. Because all matrices involved in the wavelet process are sparse, reading these values can be very efficient. When implementing wavelets for a smooth subdivision rule (such as the butterfly method), precomputing the \mathbf{I}^j and α^j matrices for each level j results in a huge savings in efficiency.

5.7 *Parallel Computation.*

The essential wavelet operations of decomposition and reconstruction are capable of being run in parallel; it is possible to split decomposition and reconstruction into separate processes on separate machines, or to run them at different times on the same machine.

Running the filter bank in parallel may be desired for improved efficiency or for memory reasons. When the input data is too large to fit into memory during filter bank decomposition, it may be split into smaller pieces (with some overlap) that are run separately. The results may be combined as necessary during reconstruction.

Parallel computation depends on locally supported wavelets. Wavelets of local support permit decomposition and reconstruction to be expressed as local operations over the vertices at each level.

Chapter 6

SMOOTH SURFACE MODELING

This chapter describes how subdivision wavelets may 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 rule, the techniques described in this chapter may be applied to general subdivision surfaces.

6.1 Motivation.

In Chapter 4, we saw examples of polyhedral compression using wavelets. In this chapter, we examine compression of surfaces defined by smooth subdivision rules. Because tangent-plane smooth subdivision rules capable of modeling curved surfaces are significantly more complex to implement than polyhedral subdivision, the question may arise: instead of compressing the smooth surface, why not simply polygonalize it, and compress its polyhedral representation?

The reason is that a smooth surface consisting of curved pieces may be much more efficiently stored using a curved basis than by using piecewise linear basis functions. In particular, a highly curved region on the surface may require many polygons to accurately approximate, but only a very few curved basis functions. This is shown conclusively in the work of Hoppe *et al.* [Hoppe et al. 94, Hoppe 94]. The second stage of their surface reconstruction algorithm produces a polyhedral surface that is optimized so as to closely fit sampled point data [Hoppe et al. 93]. In the following stage, they use as their surface primitives curved subdivision surfaces with controlled continuity. These allow an even more accurate fit to the original data, with less storage overhead.

While curved surfaces may be best fit with tangent-plane smooth wavelets, not every compression application involving smooth surfaces is best handled with exclusively smooth basis functions. When the ultimate aim is reducing the number of polygons, such as when

rendering smooth surfaces on graphics hardware, wavelet compression in the smooth basis makes little sense — polygons will need to be displayed anyway, so it is better do the compression in the form that will actually be displayed.

6.2 *Smooth Surface Compression.*

The result of smooth surface compression is shown in Figure 6.1. The surface in Figure 6.1(a) shows the level 6 Spock data set from Section 4.7 after it has been subdivided using the tangent-plane smooth butterfly subdivision rule. Figure 6.1(b) shows the result of compression using wavelets that were developed from the butterfly rule. The surface of (a) has been compressed to a representation [depicted in (b)] that may be stored using only 16% of the original coefficients.

6.3 *Efficiency.*

Implementing wavelets for smooth subdivision rules is considerably slower than for simple polyhedral subdivision. In general, the additional complexity is due to the larger support of the smooth basis functions when compared to the linear hat functions. This added support greatly complicates the function interrelationships on the same level.

In actual implementation, the precomputation techniques of Section 5.6.2 prove especially valuable. Once the α^j and \mathbf{I}^j values for each subdivision level j have been read from disk, the execution time may proceed much more quickly.

6.4 *Multiresolution Editing.*

Subdivision wavelets may also be used for editing shapes at multiple resolutions. Although we have not fully implemented multiresolution editing, Figures 6.1(c)–(d) show an example of editing the smooth shape seen in Figure 6.1(a).

Figure 6.1(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.

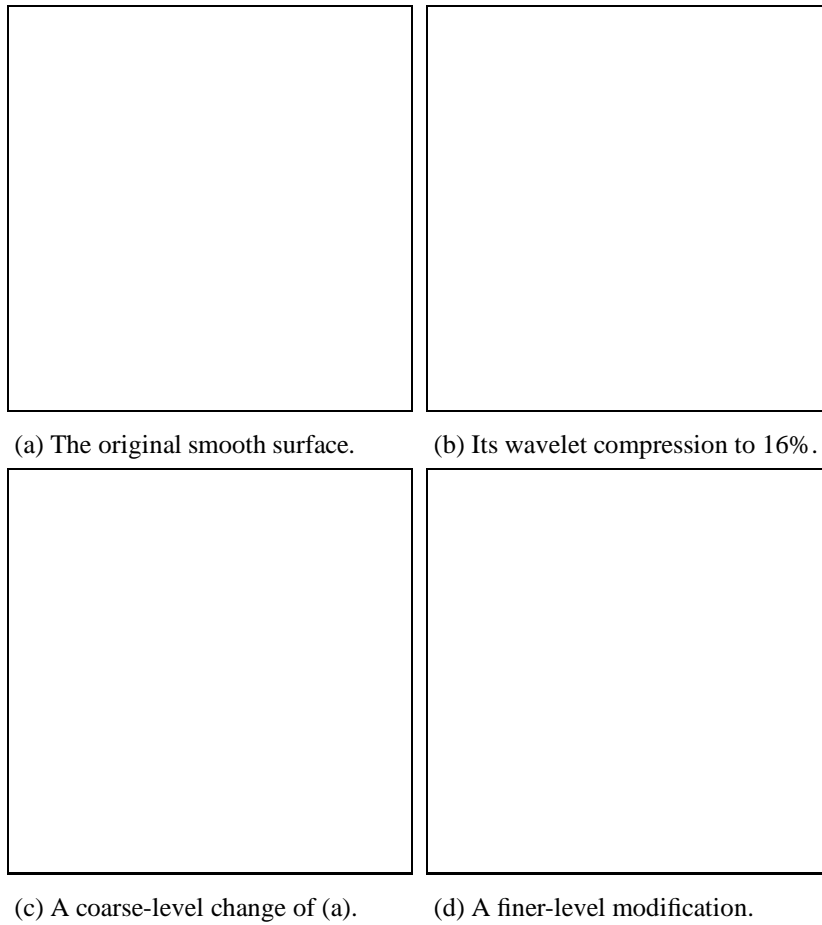


Figure 6.1: *Compression and multiresolution editing of a smooth surface.*

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 6.1(d).

The changes shown in Figure 6.1(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 [Finkelstein & Salesin 94] discuss interfaces that are more appropriate for wavelet editing of curves and surfaces.

Chapter 7

COMPARISON TO RELATED WORK

In this chapter, we compare subdivision wavelets to other techniques for modeling and polyhedral compression.

7.1 *Modeling.*

7.1.1 *Muraki.*

Muraki [Muraki 93] uses wavelets parametrized over \mathbb{R}^3 to approximate volume data in 3D. He finds them useful for efficient approximation of volume data at various levels of detail. Muraki's scaling functions are based on Blinn's *blobby model* [Blinn 82] for algebraic surface modeling.

Muraki models volumes that are parametrized over the regular \mathbb{R}^3 grid. Therefore, his method is likely to require a very large number of coefficients to accurately represent a surface, and is unable to explicitly control topological type.

7.1.2 *Finkelstein and Salesin.*

Finkelstein and Salesin [Finkelstein & Salesin 94] develop a wavelet-based method for multiresolution editing of B-spline curves. Using filter bank techniques, a fine-level B-spline curve is decomposed into its approximation at various levels of detail. They allow the user to modify the curve at arbitrary widths by modifying the underlying wavelet representation of the curve.

Their technique is useful for "top-down" design, where increasingly finer levels of detail are added to a rough initial shape. It also can be used for "bottom-up" design, where an initial detailed shape can be given a gross overall modification that does not change the fine detail. Moreover, in addition to providing a means for editing at the discrete stages of

the wavelet hierarchy, they also develop an interface for modifying at intermediate levels. The result is a versatile method for curve editing that allows modification of the curve at arbitrary levels of detail.

B-spline curve ideas are straightforward to carry over to tensor-product surfaces — a brief example is given in the paper of Finkelstein and Salesin. Although their method works well for modeling tensor-product patches, tensor-product representations, unlike subdivision wavelet, are inadequate for modeling arbitrary topological types.

7.1.3 *Forsey and Bartels.*

Hierarchical B-splines are an earlier method for modeling tensor-product B-spline surfaces at multiple resolutions. First developed in 1988 by Forsey and Bartels [Forsey & Bartels 88], hierarchical B-splines establish fundamental paradigms for editing surfaces at multiple levels of detail. The essential idea is to provide finer-level control over selected rectangular regions of a B-spline surface. In an area chosen by the user, the underlying B-spline patch may be edited by modifying finer-level control points generated by B-spline refinement. The user's changes to the finer-level points are defined as an offset from the original unmodified surface in the region.

Hierarchical B-splines have important advantages: first, they are quite convenient for efficiently modeling complex objects that have detail at various levels. Second, because the detail only needs to be stored where the user defines it, hierarchical B-splines provide an economical means of representing a complex surface. And third, because fine-level detail follows along when its underlying skeleton is modified, they also provide a convenient method for modeling figures in computer animation.

Although a powerful modeling technique, hierarchical B-splines do have some disadvantages. Originally developed for tensor-product B-spline patches, they are unable to model arbitrary topological types.

Another potential problem with hierarchical B-splines is their lack of uniqueness. A single coarse-level B-spline basis function is exactly equivalent to a linear combination of finer-level basis functions. This implies that, in contrast to a wavelet representation, any hierarchical B-spline shape may be represented in multiple ways.

Nonuniqueness of representation may cause problems in certain special situations. First,

if a coarse-level shape is needlessly modeled as a combination of finer-level shapes, there is danger of inefficient representation. Additionally, a result of this redundancy is that identical shapes can have very different underlying representations — which may lead to different results when the user tries to modify what appears to be two instances of the same shape.

While they are most obviously useful for top-down modeling, hierarchical B-splines are more complicated to integrate into a system that automatically extracts a compact hierarchical B-spline representation from a more detailed description of a complex shape. Forsey and Bartels [Forsey & Bartels 91] and Forsey and Wang [Forsey & Wang 93] have developed methods for extracting this information; however, unlike wavelets, automatic extraction is not an obvious byproduct of the technique.

The subdivision wavelets presented herein allow hierarchical modeling that overcomes these difficulties. Subdivision surfaces allow the modeling of surfaces with arbitrary topological type. Wavelets provide a natural means for efficiently and uniquely decomposing a complex shape into a concise representation.

7.1.4 Vemuri and Radisavljevic.

Vemuri and Radisavljevic [Vemuri & Radisavljevic 94] develop a wavelet-based technique for representing shapes built from deformable superquadrics [Terzopoulos & Metaxas 91]. Their method is capable of modeling topological cylinders at various resolutions, and they also describe a technique for fitting shapes to sampled medical data. Wavelets give them the advantages of a unique representation and the ability to smoothly blend between representations at different levels of detail; they find this useful for multigrid optimization. Because it is based on dilates and translates of functions over the cylindrical domain, their method is only able to model shapes of limited topological type.

7.2 Polyhedral Compression.

Replacing complex models with simpler approximations is a good way to speed up rendering and modeling applications. The importance of the polyhedral compression problem is evidenced by the growing body of work that attempts to solve it. This section surveys notable solutions to the problem and compares them to the compression methods based on subdivision wavelets that are described in Chapter 4. Polyhedral compression in the con-

text of its rendering applications is also surveyed by Heckbert and Garland [Heckbert & Garland 94].

The following criteria are used in my comparison of polyhedral compression algorithms:

1. *Curvature match.* Quality is an important consideration when assessing a compression scheme. Although it is difficult to measure a variety of published methods with a given quality metric, it is possible to visually examine how well a method adapts to changes in curvature.

In a concise representation, important features should be kept, but areas of little variation require little extra information. Thus, on a polyhedral model, areas of high curvature require many polygonal facets, while large flat regions with little curvature can be represented with many fewer polygons. Hence, a simple way to judge the effectiveness of an approximation is to see how well the approximation adapts itself to changing curvature over the input model. Because the curvature at a point on the surface is not simply a scalar, but varies according to the direction, correctly adjusting to curvature can be especially difficult to achieve.

This criterion states whether the given method adapts to regions of varying curvature. As presented, this judgment is still somewhat subjective, but appears to be a useful standard for comparing approximation quality.

2. *Closeness measure.* For many applications, it is helpful to control the approximation according to a measure of how closely the result matches the original surface. In general, the most useful measure is an L^∞ norm, specifying an upper bound for the distance of any point on the approximation from its corresponding point on the input surface. The L^2 norm, measuring the sum of the squares of these distances, may also be useful. This criterion simply states whether a closeness measure is supplied for the given algorithm.
3. *Specifies compression.* For other applications, it is useful to specify in advance exactly how much compression to achieve in the approximation. For example, it may be known that an animation runs in real time only when no more than a fixed number

of polygons are processed per time step. This point says whether it is possible to specify the compression size in advance.

4. *Smooth blends.* If the approximation is to be used for animation, sudden changes of detail in the model (such as during a sequence in which the camera zooms into the object) are especially noticeable. It is therefore important for this application that there be a mechanism that allows gradual blending between approximations at various complexities.
5. *Preserves topology.* For most compression applications, it is preferable to modify the topological type of the coarse-level approximation into a simpler form. For example, any object viewed from far enough away on a computer screen can be represented very simply, with only a few pixels. At this level, it is needlessly inefficient to preserve a complex topological type in the approximation. In other applications, such as applying a texture map over the shape, it may be preferable that the topological type be preserved. This criterion simply states whether the algorithm preserves the topology in an approximation.
6. *Speed.* Finally, for interactive applications, polyhedral approximation should run quickly.

7.2.1 Turk.

Turk [Turk 91, Turk 92] develops an early method in which the user first determines the number of vertices to use in the approximation. Turk's algorithm places this number of vertices over the original polyhedral surface, and then uses repelling forces to spread them out.

When the repelling force is uniform over the mesh, the result is a surface with evenly spaced vertices, regardless of variations in the local curvature. In an attempt to overcome this drawback, Turk varies the repelling forces according to curvature information on the original surface. When actual curvature is not known, Turk's algorithm attempts to estimate it. For many data sets with sudden curvature changes, such as brain data with folds, this curvature estimation fails. Additionally, his paper treats curvature as a scalar value at a

point, when in fact it varies according to direction. (Turk notes that directional curvature is an area for future research.)

Although the result of Turk's simplification is guaranteed to be a mesh with vertices on the original surface, he provides no means of guaranteeing how far the approximation is from the original data at other points.

7.2.2 *Rossignac and Borrel.*

Rossignac and Borrel [Rossignac & Borrel 93] describe a simple and highly efficient method for compressing polyhedra. Their technique dices the ambient space into smaller subregions. Multiple vertices in a subregion are coalesced into a single vertex using a weighted approximation, such as their centroid. The coalesced vertices are then reconnected with their neighbors to form a collection of faces.

Because the connectivity of the approximation mesh derives only from geometric proximity of the subregions, it does not preserve the topological type of the original shape. For most applications, this is a significant advantage. Rossignac and Borrel give the only technique surveyed here that enjoys this property.

The method as described is highly dependent upon the gridding chosen in the partitioning. Hence, it is not translation- or rotation-invariant. By setting the size of the partitioning region, the user may roughly control the accuracy and fineness of detail appearing in the approximation. The paper additionally describes a method for blending between approximations at different levels of detail.

Because the detail in the approximation depends solely upon the size of the predetermined regions, the resulting approximation is unable to adapt itself to regions where the curvature changes significantly. The result is an approximation in which finer-level detail tends to be blurred, obscuring potentially important features of higher curvature.

The estimated run time for this technique is the best of all methods surveyed in this section.

7.2.3 *Schroeder et al.*

Schroeder *et al.* [Schroeder et al. 92] provide an iterative mechanism for reducing mesh complexity. Their algorithm, which they term "decimation," repeatedly passes over the

mesh, looking for areas that may be simplified. Because their technique evaluates the effect of a simplification by comparing between successive approximations, there is no way to measure how far the end result may deviate from the original data.

The meshing figures in the Schroeder *et al.* paper appear to demonstrate that the scheme adapts to changes in curvature, but the method does not treat the problem of blending between approximations. No information concerning run time has been found.

7.2.4 Hoppe *et al.*

Hoppe *et al.* [Hoppe *et al.* 93, Hoppe *et al.* 94, Hoppe 94] develop methods for compressing both piecewise linear polyhedral surfaces as well as meshes that represent subdivision surfaces. They approximate the original surface by computing a minimization that balances both the simplicity of the representation and an L^2 distance measure from the original surface.

The approximation provided by Hoppe *et al.*'s algorithm appears to have exceptional quality, better than any other method surveyed in this section. The optimization of their distance metric produces a surface with very good adaption to the local curvature. This quality has a high cost in efficiency, however. The scheme runs in hours, and is not yet applicable for interactive use.

There is no mechanism for blending between models with different detail, nor does the method allow one to directly specify in advance how much to compress the data.

7.2.5 Subdivision Wavelets.

We have presented in this dissertation a method for polygonal compression that is based on wavelets defined over subdivision surfaces. (This method is described in more detail in Chapter 4.) The quality of the output is not as outstanding as that of Hoppe *et al.*, but appears to compare favorably with the results of Turk, Schroeder *et al.*, and Rossignac & Borrel. In particular, a simple implementation of the algorithm produces polyhedral approximations that are more densely triangulated in areas of higher curvature.

In general, wavelet techniques allow simple implementation of an L^2 measure of closeness. For polyhedral data, it is also possible to control the closeness of the approximation within an L^∞ norm. Alternatively, an approximation may be developed that is both a good

fit, and whose size is within a constant bound of a predetermined compression factor. The method of compression based on subdivision wavelets may also be used to blend between different levels of detail, and runs quite efficiently in comparison to most other compression methods.

One important requirement for subdivision wavelets is that the input have subdivision connectivity, as discussed in Section 4.6. In general, this requires a preprocessing step before actually compressing an arbitrary polyhedron.

7.2.6 Summary.

Table 7.1 compares the methods surveyed according to the above criteria. The run times presented are for compressing an initial mesh with around 30,000 triangles to about 15% of its original size on an SGI Indigo² Extreme. (These times are estimated for the technique of Rossignac and Borrel, given the run times for smaller data, and assuming a linear-time algorithm.)

Table 7.1: A comparison of polygonal compression methods.

Method	Curvature match	Closeness measure	Specifies compression	Smooth blends	Preserves topology	Speed (sec.)
Turk	Poor	None	Yes	Yes	Yes	300–600
Rossignac & Borrel	Poor	L^∞	No	Yes	No	20–30
Schroeder <i>et al.</i>	Good	None	Yes	No	Yes	?
Hoppe <i>et al.</i>	Excellent	L^2	No	No	Yes	4800–7200
Subdivision wavelets	Good	L^∞, L^2	Yes	Yes	Yes	50–55

Other polygonal compression methods are not surveyed here, due to lack of information about their quality and execution speed. These include the methods of Kalvin and Taylor [Kalvin & Taylor 93], DeHaemer and Zyda [DeHaemer & Zyda 91], and Hamann [Hamann 94].

Chapter 8

SUMMARY AND FUTURE WORK

In this dissertation, we have established a theoretical basis for applying multiresolution analysis to surfaces of arbitrary topological type. These results hold for any uniformly convergent continuous and local primal subdivision scheme, including polyhedral subdivision, the butterfly method [Dyn et al. 90], Loop's method [Loop 87], and Catmull-Clark surfaces [Catmull & Clark 78]. The results also hold for piecewise smooth subdivision as described in Hoppe *et al.* [Hoppe et al. 94, Hoppe 94], and for open surfaces that possess holes or boundary curves.

Subdivision surfaces play an important role in this theory, and appear to be an increasingly attractive method for modeling complex surfaces. Subdivision surfaces are chosen as the building blocks for these wavelets because they allow the construction of smooth surfaces of arbitrary topological type, and because, as was demonstrated in Section 3.1.1, they are refinable.

Besides refinability, the other key ingredient needed for multiresolution analysis is integrability. Although subdivision surfaces do not possess a closed form for evaluation, we show in Section 3.2 that one may nevertheless compute inner product integrals on them exactly, as the solutions to a linear system.

These results were tied together in the construction of unique wavelets over subdivision surfaces. Although the construction of Section 3.3.2 results in wavelets that are orthogonal, they are impractical due to their infinite support. In Section 3.3.3 we provide a locally supported approximation that, unlike simple truncation, allows invertible decomposition. The resulting functions allow linear time decomposition and reconstruction for many kinds of subdivision rules, including polyhedral subdivision and the tangent-plane smooth butterfly method of Dyn *et al.* They are also shown to be perfectly adequate for efficient wavelet compression of complex data, as is shown by the examples of Sections 4.7 and 6.2.

Algorithms to implement subdivision wavelets are not overly complicated. The details of these algorithms as they apply to surface compression and smooth level-of-detail control

are presented in Chapters 4 and 5. The practical issues of squeezing memory for a large representation and of efficiently running wavelet compression in interactive viewing are also presented.

8.1 *Potential Applications.*

In previous chapters, we examined the details of implementing subdivision wavelets for shape compression, function approximation, animation previewing, and for smoothly varying the level of detail in animation. Further applications of wavelets over surfaces require more study. This section suggests promising applications of subdivision wavelets that have not yet been implemented.

8.1.1 *Radiosity.*

Hanrahan *et al.* [Hanrahan et al. 91] present a hierarchical method for radiosity. Their technique assumes that a scene is represented by large polygonal patches that may be broken up into smaller elements of constant brightness. They describe a hierarchical algorithm to determine how to break up the larger patches into smaller pieces. The result is an algorithm that efficiently concentrates computation in regions of greatest change.

While their technique works well for choosing where to decompose the large regions into smaller elements, Hanrahan *et al.* do not have a method for creating larger patches from a *cluster* of smaller patches. Hence, the efficiency of their technique is dependent upon the number of coarse-level patches in the input scene. If the scene initially has many patches, their method cannot significantly improve the run time.

Subdivision wavelets can be used to cluster a large collection of small patches on an object of arbitrary topological type into a coarse-level approximation. This can be used as a preprocessing step to the algorithm of Hanrahan *et al.* Additionally, the wavelet coefficients produced in the course of this approximation may be used to guide hierarchical radiosity as it searches for places to split the patches.

Bidirectional reflectance distribution functions (BRDFs) are used to model light reflectance in global illumination. BRDFs are defined over a hemispherical domain, but are usually modeled with either an approximation parametrized over the grid [Christensen et al.

94] or with spherical harmonics [Sillion et al. 91]. Using subdivision wavelets, it is possible to approximate BRDFs with wavelets truly parametrized over a hemispherical domain. This can lead to improved computation for radiance distribution.

8.1.2 *Multiresolution Editing.*

The techniques of Finkelstein and Salesin [Finkelstein & Salesin 94] demonstrate the usefulness of wavelets for editing B-spline curves and tensor-product B-spline surfaces. Their method may be adapted to provide multiresolution editing for surfaces of arbitrary topological type. Depending on the subdivision rules employed, it is possible to edit polyhedra, smooth subdivision surfaces, surfaces with boundaries, and smooth subdivision surfaces with sharp corners and “darts,” as in Hoppe *et al.* [Hoppe et al. 94, Hoppe 94].

The examples of Section 6.2 show that subdivision wavelets have the potential for multiresolution editing of surfaces of arbitrary topological type. However, this implementation lacks an acceptable user interface.

Following Forsey and Bartels [Forsey & Bartels 88] and Finkelstein and Salesin, it may be desirable to represent the surface geometry as a function of the surface normal. As demonstrated in Forsey and Bartels, this can be a convenient way to represent surface detail as an offset from the underlying coarser representation.

8.1.3 *Surface Optimization and Multigrid Methods.*

Because of the complexity of the function being minimized, nonlinear optimization with nonlinear constraints requires iterative techniques to converge to a solution. Searching for the n parameters with a minimum function value is equivalent to searching n -dimensional space. The surface optimization implemented by Moreton and Séquin [Moreton 92, Moreton & Séquin 92] searches to find the optimal setting of hundreds of free variables; their modified conjugate-gradient method requires hours or days to converge for relatively simple surface shapes.

An iterative optimization technique begins with an initial guess at the minimum, then attempts to move closer by constructing a series of approximations around the current guess. If the guess is close enough to the actual minimum, the local approximation used in the iterative technique is a good approximation of the true neighborhood around the minimum.

Hence, an important principle in iterative nonlinear optimization is to begin a search with a good initial guess.

In a multigrid approach, the functional being minimized is first optimized over a low-dimension approximation to the problem space. When this minimum is located, more detail is added to the search — using the previous level’s minimum as a starting point for the next iteration. Several important properties are required in order for this approach to work. First, the lower-level representation must be a good approximation to the more complex function; and second, there must be a way to directly map a solution point in the simpler space to a starting point in the more complex space by adding the additional detail in a coherent fashion.

The subdivision wavelets developed in this dissertation contain these properties, and hence would appear to be strong candidates for implementing multigrid techniques for complex nonlinear surface optimization over surfaces of arbitrary topological type. Pentland [Pentland 92] shows that wavelets permit multigrid techniques to be extended for surface interpolation and optimization problems over regular grid domains. Meyers [Meyers 94a, Meyers 94b] uses wavelet approximations of curves to implement a multigrid technique for area minimization of contour tilings [Meyers et al. 92]. The resulting technique runs in $O(n \log n)$ time, and produces results that are comparable to the standard $O(n^2 \log n)$ global optimization algorithm of Fuchs *et al.* [Fuchs et al. 77]. A similar speedup of one-dimensional optimization using wavelet-based multigrid techniques is reported by Gortler *et al.* [Gortler et al. 93].

Outside of the wavelet literature, Yserentant [Yserentant 86] does subdivision on non-regular grids in order to develop multigrid optimization techniques over arbitrary surface domains. He develops a nonorthogonal hierarchical basis for this work, and shows that the resulting multigrid algorithms run in time close to that of more traditional multigrid techniques over regular domains.

8.1.4 *Three-Dimensional Morphing.*

Morphing, or transforming a three-dimensional object into another of similar topological type, is another area to which subdivision wavelets may be applied. Although we do not provide a specific implementation of wavelet-based morphing, many key elements of mor-

phing are provided by subdivision wavelets, including:

- A parametrization that establishes a correspondence between two objects of similar topological type.
- The ability (through the filter bank process) to extract important distinguishing features on each object for removal and readdition.
- A mechanism for smoothly blending in a change.

Kent *et al.* [Kent et al. 92] describe a technique for homeomorphic transformation between two shapes parametrized over the sphere. In the first stage, a spherical parametrization is developed over each object, creating a correspondence between them.

Creating a correspondence between two shapes is a problem similar in difficulty to the remeshing problem described in Section 4.6 of this dissertation. The difference is that Kent *et al.* restrict their objects to be topologically equivalent to a sphere — because parametrizing over other topological types is a difficult problem. In Section 3.1.1 we explain how such a parametrization may be performed. Subdivision wavelets allow the effects of Kent *et al.* to be extended to permit transformations between topologically similar shapes of arbitrary topological type, not just topological spheres.

Hughes [Hughes 92] develops a technique for smoothly transforming between volumetric shapes of any topological type, parametrized over the regular \mathbb{R}^3 domain. His technique is to first compute a Fourier analysis of the shapes. The high-frequency data on the first shape is smoothly removed, and an interpolation is built between the low-frequency features of the first shape and the target shape. Finally, the high-frequency features of the second shape are smoothly added in.

As already discussed in Section 2.2.5, Fourier analysis has features in common with multiresolution analysis, and a similar technique could be used for transforming between shapes represented with wavelets. In the first stage, the fine detail (corresponding to high-frequency data) could be removed. Next, an interpolation can be built between the coarser-level shapes.

A restriction when using wavelets to effect a transformation is that only shapes of similar topological type can be morphed. In contrast, Hughes' method allows a transformation

between shapes of varying topological type; however, his method applies only to volume data, not surface data over irregular domains.

8.2 *Future Work.*

Although this dissertation lays out much of the theoretical foundation for applying wavelets to surfaces of arbitrary topological type, there are still many areas of theoretical and practical interest that should be explored further.

8.2.1 *Conversion to Subdivision Connectivity.*

As explained in Section 4.6, the current theory is practical only for input meshes whose connectivity results from recursively subdividing a simple mesh. We are currently investigating resampling algorithms that approximate a mesh of arbitrary topological type and connectivity with a mesh of equivalent topological type and more appropriate subdivision connectivity.

Given an arbitrary mesh of triangles, the object of such resampling is to produce a greatly simplified mesh of equivalent topological type together with a representation of its subdivision mesh that has a value for each subvertex that approximates a point on the original surface.

One such algorithm (proposed by Tom Duchamp) proceeds by repeatedly coalescing faces, and keeping track of where the original vertices are with respect to the coalesced face. The algorithm proceeds until no more faces may be coalesced without changing the topological type. Each face on the simplified mesh may be subdivided to an appropriate depth, and values may be assigned to its subvertices based on the values of the closest vertices in the original mesh to the respective parameter value.

This algorithm has not yet been implemented, but analysis suggests an $O(n \log n)$ run time in the number of input vertices.

8.2.2 *Weighted Inner Product.*

Because the inner products of Section 3.2 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 less satisfying (but more easily implemented) possibility would be to modify the preprocessing phase in order to approximate the input with triangles that do not significantly vary in size. This would create a close match between the inner product weights and the actual sizes of the triangles.

8.2.3 *Linear Time Sparse Matrix Inversion.*

As explained in Section 3.3.4, 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.

8.2.4 *Locally Supported Orthogonal Wavelets.*

The approximations developed in Section 3.3 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.

Furthermore, 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 [Quak & Weyrich 93] would also need to be developed for a linear time filter bank.

8.2.5 *Real-time Level-of-Detail Control.*

The methods of Chapter 5 are useful for developing frame-by-frame animation with level-of-detail control. An application with a much wider potential for games and film is truly

real-time animation of complex 3D models, with level of detail provided by subdivision wavelet techniques.

The unoptimized methods outlined in this dissertation require at least several seconds per frame to animate the 32,000 polygon Spock bust at SVHS resolution. This factor implies that the methods are still at least 100 times too slow for real-time level of detail without a significant change in the algorithm. Because careful optimizations, such as those discussed in Sections 5.5 and 5.6, are likely to reduce the processing time, and because this is an application with broad potential for the future of interactive 3D viewing, further research into making level of detail work in real time is a direction with a good potential for significant payoff.

8.2.6 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.

8.2.7 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.

8.2.8 Inner Product Classification

As explained in Section 3.2.3, the detection and caching of repeated inner product cases can greatly reduce the size of the linear system that must be solved to compute the inner product values. The difficulty of parametrizing around extraordinary points makes classifying these

cases difficult. An efficient classification method would greatly help the solution of inner products.

8.2.9 *Computing Inner Products with the Duals*

In Section 3.5, we gave a construction for the scaling function duals. Unfortunately, the construction we gave had two practical difficulties for computing scaling function coefficients. First, we know of no way to compute the necessary inner products except through numerical approximation. Second, the duals themselves are globally supported, which precludes their computation in linear time. In cases when efficient evaluation of Equation 2.1 is required, both these problems must be overcome.

8.3 *Conclusion.*

This dissertation brings the power of multiresolution analysis to functions defined on surfaces of arbitrary topological type. The subdivision wavelets developed herein have significant practical importance for the fast-growing field of 3D surface modeling and display.

Subdivision wavelets are most efficient for decomposing data generated from a static, known function. For data of this sort, it is possible to run wavelet decomposition in advance. When the input data is generated according to a dynamic function that changes unpredictably over time, it may be necessary to repeat the decomposition filter over the areas affected by the change. An example of an application that would require repeated processing is a 3D model that is interactively modified by a user.

A carefully constructed special-purpose algorithm may produce superior results for some applications proposed in this dissertation. However, the wide range of possible applications demonstrates the versatility of subdivision wavelets, and shows them to be a reusable tool that naturally and efficiently solves or accelerates a wide range of common problems in computer graphics and modeling. As an illustration of this versatility, a complex object of arbitrary topological 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 subdivision wavelet representation.

Subdivision wavelets inherit from more standard wavelet methods a high degree of efficiency in the central algorithms used to decompose and reconstruct approximations

over surfaces. The theory and techniques developed in this dissertation provide a versatile method with a demonstrated ability to accelerate a wide range of methods throughout the field of computer graphics and modeling.

Bibliography

- [Arneodo et al. 94] A. Arneodo, E. Bacry, and J.-F. Muzy. Solving the inverse fractal problem from wavelet analysis. *Europhysics letters*, 25(7):479–484, 1 March 1994.
- [Bartels et al. 87] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for use in Computer Graphics & Geometric Modeling*. Morgan Kaufmann, Los Altos, California, 1987.
- [Bartle 64] R. G. Bartle. *The Elements of Real Analysis*. John Wiley and Sons, 1964.
- [Berman et al. 94] D. Berman, J. Bartell, and D. Salesin. Multiresolution painting and compositing. *Computer Graphics Annual Conference Series*, pages 85–90, July 1994.
- [Beylkin et al. 91] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.
- [Blinn 82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.
- [Boehm 80] W. Boehm. Inserting new knots into B-spline curves. *Computer Aided Design*, 12(4):199–201, 1980.
- [Burt & Adelson 83] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983.
- [Catmull & Clark 78] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

- [Cavaretta et al. 91] A. S. Cavaretta, W. Dahmen, and C. A. Micchelli. Stationary subdivision. *Memoirs of the American Mathematical Society*, 435:1–186, 1991.
- [Christensen et al. 94] P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Wavelet radiance. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 287–302, Darmstadt, Germany, 1994.
- [Chui & Shi 92] C. K. Chui and X. Shi. Wavelets and multiscale interpolation. In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II*. Academic Press, 1992.
- [Chui 92a] C. K. Chui. *Wavelet Analysis and its Applications*, volume 1. Academic Press, Inc., Boston, 1992.
- [Chui 92b] C. K. Chui. *Wavelet Analysis and its Applications*, volume 2. Academic Press, Inc., Boston, 1992.
- [Dahmen & Micchelli 93] W. Dahmen and C. A. Micchelli. Using the refinement equation for evaluating integrals of wavelets. *SIAM Journal on Numerical Analysis*, 30(2):507–537, April 1993.
- [Daubechies 92] I. Daubechies. *Ten lectures on wavelets*. SIAM, Philadelphia, 1992.
- [DeHaemer & Zyda 91] M. J. DeHaemer and M. J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, 15(2):175–184, 1991.
- [DeVore et al. 92] R. DeVore, B. Jawerth, and B. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746, March 1992.
- [Dongarra et al. 79] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, 1979.

- [Doo & Sabin 78] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.
- [Doo 78] D. Doo. *A Recursive Subdivision Algorithm for Fitting Quadratic Surfaces to Irregular Polyhedrons*. PhD dissertation, Brunel University, 1978.
- [Duchamp & Schweitzer 94] T. Duchamp and J. Schweitzer. A convergence proof for subdivision surfaces. In preparation, 1994.
- [Dyn et al. 90] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.
- [Dyn et al. 93] N. Dyn, S. Hed, and D. Levin. Subdivision schemes for surface interpolation. Technical Report 8-93, Department of Applied Mathematics, Telaviv University, Telaviv, Israel, February 1993.
- [Farin 93] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, third edition, 1993.
- [Finkelstein & Salesin 94] A. Finkelstein and D. Salesin. Multiresolution curves. *Computer Graphics Annual Conference Series*, pages 261–268, July 1994.
- [Forsey & Bartels 88] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics*, 22(4):205–212, 1988.
- [Forsey & Bartels 91] D. R. Forsey and R. H. Bartels. Tensor products and hierarchical fitting. In *Volume 1610 Curves and Surfaces in Computer Vision and Graphics II*, pages 88–96. SPIE, 1991.
- [Forsey & Wang 93] D. Forsey and L. Wang. Multi-resolution surface approximation for animation. In *Proceedings of Graphics Interface*, 1993.

- [Fuchs et al. 77] H. Fuchs, Z. M. Kedem, and S. P. Uelson. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, October 1977.
- [Funkhouser & Séquin 93] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics Annual Conference Series*, pages 247–254, August 1993.
- [Garey & Johnson 79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
- [Gondek et al. 94] J. S. Gondek, G. W. Meyer, and J. G. Newman. Wavelength dependent reflectance functions. *Computer Graphics Annual Conference Series*, pages 213–220, July 1994.
- [Gortler et al. 93] S. Gortler, P. Schröder, M. F. Cohen, and P. Hanrahan. Wavelet radiosity. *Computer Graphics Annual Conference Series*, pages 221–230, August 1993.
- [Halstead et al. 93] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Computer Graphics Annual Conference Series*, pages 35–44, August 1993.
- [Hamann 94] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, April 1994.
- [Hanrahan et al. 91] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, July 1991.
- [Heckbert & Garland 94] P. S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Proceedings of Graphics Interface*, pages 43–50, 1994.
- [Hoppe 94] H. Hoppe. *Surface Reconstruction from Unorganized Points*. PhD dissertation, University of Washington, Seattle, Washington, June 1994. TR 94-06-01.

- [Hoppe et al. 93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics Annual Conference Series*, pages 19–26, August 1993.
- [Hoppe et al. 94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics Annual Conference Series*, pages 295–302, July 1994.
- [Hughes 92] J. F. Hughes. Scheduled Fourier volume morphing. *Computer Graphics*, 26(2):43–46, 1992.
- [Jia & Micchelli 91] R.-Q. Jia and C. 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.
- [Kalvin & Taylor 93] A. D. Kalvin and R. H. Taylor. SuperFaces: polyhedral approximation with bounded error. Research Report RC 19135 (82286), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, 1993.
- [Kent et al. 92] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. *Computer Graphics*, 26(2):47–54, 1992.
- [Liu et al. 94] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. *Computer Graphics Annual Conference Series*, pages 35–22, July 1994.
- [Loop 87] C. T. Loop. Smooth subdivision surfaces based on triangles. Master’s thesis, Department of Mathematics, University of Utah, August 1987.
- [Lounsbery et al. 92] M. Lounsbery, S. Mann, and T. DeRose. Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52, September 1992.

- [Lucier 92] B. J. Lucier. Wavelets and image compression. In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II*. Academic Press, 1992.
- [Lyche et al. 80] T. Lyche, E. Cohen, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, 1980.
- [Mallat & Hwang 92] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, March 1992.
- [Mallat 89] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.
- [Mann 92] S. Mann. *Surface Approximation Using Geometric Hermite Patches*. PhD dissertation, University of Washington, Seattle, WA, 1992.
- [Mann et al. 92] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan. A survey of parametric scattered data fitting using triangular interpolants. In H. Hagen, editor, *Curve and Surface Design*, pages 145–172. SIAM, 1992.
- [Meyers 94a] D. Meyers. Multiresolution tiling. In *Proceedings of Graphics Interface*, pages 25–32, May 1994.
- [Meyers 94b] D. Meyers. *Reconstruction of Surfaces from Planar Contours*. PhD dissertation, University of Washington, Seattle, Washington, July 1994.
- [Meyers et al. 92] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, July 1992.

- [Moreton & Séquin 92] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992.
- [Moreton 92] H. P. Moreton. *Minimum Curvature Variation Curves, Networks, and Surfaces for Fair Free-Form Shape Design*. PhD dissertation, University of California at Berkeley, Berkeley, California, December 1992.
- [Muraki 93] S. Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications*, 13(4):50–56, July 1993.
- [Newhard & Nicol 84] Newhard and M. Nicol. Selective scene management in flight simulator visual systems. In *Proceedings of the Sixth Interservice/Industry Training Equipment Conference*, Oct 22–24 1984.
- [Pentland 92] A. P. Pentland. Fast solutions to physical equilibrium and interpolation problems. *Visual Computer*, 8(5-6):303–314, June 1992.
- [Press et al. 89] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in Pascal*. Cambridge University Press, Cambridge, 1989.
- [Quak & Weyrich 93] E. Quak and N. Weyrich. Decomposition and reconstruction algorithms for spline wavelets on a bounded interval. Technical Report 294, Department of Mathematics, Texas A&M University, April 1993.
- [Ramshaw 87] L. Ramshaw. Blossoming: A connect-the-dots approach to splines. Technical report, Digital Systems Research Center, Palo Alto, Ca, 1987.
- [Reif 92] U. Reif. A unified approach to subdivision algorithms. Technical Report A-92-16, Universität Stuttgart, Stuttgart, Germany, 1992.
- [Reif 93] U. Reif. *Neue Aspekte in der Theorie der Freiformflächen beliebiger Topologie*. PhD dissertation, Mathematisches Institut, Universität Stuttgart, Stuttgart, Germany, 1993.

- [Rioul & Vetterli 91] O. Rioul and M. Vetterli. Wavelets and signal processing. *IEEE Signal Processing Magazine*, 8(4):14–38, October 1991.
- [Rossignac & Borrel 93] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer-Verlag, 1993.
- [Schröder et al. 93] P. Schröder, S. J. Gortler, M. F. Cohen, and P. Hanrahan. Wavelet projections for radiosity. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 105–114, Paris, France, 1993.
- [Schroeder et al. 92] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.
- [Sillion et al. 91] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics*, 25(4):187–196, July 1991.
- [Smits et al. 94] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics Annual Conference Series*, pages 435–442, July 1994.
- [Stollnitz et al. 94] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. Wavelets for computer graphics: A primer. In preparation, 1994.
- [Terzopoulos & Metaxas 91] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.
- [Turk 91] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics*, 25(4):289–298, 1991.
- [Turk 92] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, 1992.

- [Vemuri & Radisavljevic 94] B. C. Vemuri and A. Radisavljevic. Multiresolution stochastic hybrid shape models with fractal priors. *ACM Transactions on Graphics*, 13(2):177–207, April 1994.
- [Weiler 85] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, January 1985.
- [Yserentant 86] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, August 1986.

Vita

John Michael Lounsbery was born September 24th, 1966 in Sasebo, Japan. In 1984 he received a B.S. degree from the University of Maryland at College Park, majoring in Computer Science. In 1989 he received a M.S. degree in Computer Science at the University of Washington, and completed his Ph.D. in Computer Science and Engineering at the University of Washington in 1994. Besides computer graphics and modeling, his interests include classical archaeology, Latin literature, and renaissance and baroque art and architecture.

List of publications:

- “A Tutorial Introduction to Blossoming,” in H. Hagen and D. Roller, editors, *Geometric Modeling: Methods and Applications*, Springer-Verlag, 1990, (with T. DeRose and R. Goldman).
- “A Testbed for the Comparison of Parametric Surface Methods,” in the proceedings of *SPIE/SPSE Symposium on Electronic Imaging Science and Technology*, Santa Clara, CA, February, 1990 (with C. Loop, S. Mann, D. Meyers, J. Painter, T. DeRose and K. Sloan).
- “A Survey of Triangular Scattered Data Fitting,” in H. Hagen, editor, *Curve and Surface Design*, pages 145–172, SIAM, 1992 (with S. Mann, C. Loop, D. Meyers, J. Painter, T. DeRose and K. Sloan).
- “Parametric Surface Interpolation,” in *IEEE Computer Graphics & Applications*, 12(5):45–52, September 1992, (with S. Mann, T. DeRose).

