WAVELET BASED MULTIRESOLUTION REPRESENTATIONS: IMAGES, CURVES AND SURFACES

by

SHRIJEET S. MUKHERJEE

A THESIS

Presented to the Department of Computer
and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

August 1997

WAVELET BASED MULTIRESOLUTION REPRESENTATIONS:

IMAGES, CURVES AND SURFACES

by

SHRIJEET S. MUKHERJEE

A THESIS

Presented to the Department of Computer
and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

August 1997

" Wavelet Based Multiresolution Representations: Images, Curves and Surfaces ," a thesis prepared by Shrijeet S. Mukherjee in partial fulfillment of the requirements for the Master of Science degree in the Department of Computer and Information Science. This thesis has been approved and accepted by:

Gary W. Meyer

Dr. Gary W. Meyer

8/25/97

Date

Accepted by:

Vice Provost and Dean of the Graduate School

An Abstract of the Thesis of

Shrijeet S. Mukherjee for the degree of Master of Science
in the Department of Computer and Information Science
to be taken August 1997

Title: WAVELET BASED MULTIRESOLUTION REPRESENTATIONS:
IMAGES, CURVES AND SURFACES

Approved: Gary W. Meyer
Dr. Gary W. Meyer

This thesis explores wavelet based multiresolution representation schemes. We survey existing research in this area, describe implementation issues that arise, and, in general, attempt to provide a unified discussion of past work. Since the computer graphics community uses images, curves, and surfaces as its three primary primitives, we have used that as the ordering for our investigations into wavelet based representation schemes.

We begin by constructing a general purpose wavelet transformer and building the related mathematical background. Some common properties and basis functions are then examined. The framework is then extended to two dimensional image data, following which we extend the results to three dimensions. We then develop and instantiate a framework for an implementation of multi-resolution B-splines.

Lastly we present a preprocessing step to convert arbitrary unstructured meshing into a structured scheme. A wavelet basis for the structured spaces is introduced followed by an implementation of a multiresolution surface.

CURRICULUM VITA

NAME OF THE AUTHOR: Shrijeet S. Mukherjee

PLACE OF BIRTH: Calcutta,India.

DATE OF BIRTH: 26th of December, 1972.

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon
University of Poona.

DEGREES AWARDED:

Master of Science in Computer Science, 1997, University of Oregon.
Bachelor of Engineering in Computer Engineering, 1994, University of Poona.

PROFESSIONAL EXPERIENCE:

Network Engineer, University of Oregon, Network Services.

Video and Networking Division, Tektronix Inc, Summer Intern, 1996.

Research Engineering, Aerospace Engineering, Indian Institute of Tech, Bombay.

AWARDS AND HONORS:

Second Rank Award 1992-93.
Second Rank Award 1993-94.

PUBLICATIONS:

S. Mukherjee, M. Apte, A. Isaacs, S. Hadap and G. R. Shevare. Multiblock grid generation for structured grids in volume. In the 5th *International Conference on Numerical Grid Generation for Computational Fluid Simulation.* (Apr 96). 1996.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Gary Meyer for the encouragement, support and education into the art of science that he provided. To put it simply, without him, this document would not exist.

I also take this opportunity to thank the numerous students and faculty members, here at the University of Oregon, without whom I would not have had any chance of completion. In particular, I have to thank Ferenc Rakoczi, Mark Bolin and Amitabh Roy for the spirited discussions and help regarding the rather cryptic subject matter. Chris Chung and Gregston Chu without whose help with editing and typesetting, this document would not have been able to gain the present form. Lastly, I need to thank Chris Schleicher and Sara Goodale for their help in proof reading my ghastly grammar.

I would also like to take this space to thank the wonderful areas of mathematics and computer science that made this investigation, both possible and necessary.

DEDICATION

To my Parents

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

- The Wonderful World of Wavelets 1
- Why Wavelets 3

II. THE MATHEMATICAL BASIS 5

- Mathematical Preliminaries 5
- Discrete Wavelet Transformation 9
- Multiresolution Analysis 13
- Coveted Properties 16
- The Filter Bank Algorithm 17
- Haar Wavelets: A Case Study 20
- Other Famous Wavelets 25

III. 2-D WAVELETS : IMAGES 29

- Wavelets in 2-Dimensions 29
- Standard Versus Non-Standard Decomposition 30
- Applications 33
- The Power of Wavelets 37

IV. 3-D WAVELETS : CURVES 39

- Choosing The Wavelet 39
- Implementation Issues 48
- Curve Sweep Versus Character 49
- Examples 49

V. SUBDIVISION CONNECTIVITY AND REMESHING 52

- Surface Representations 53
- Need for Order 54
- Meshing Basics 55
- Subdivision Connectivity 56

Chapter Page

The Remeshing Algorithm 57
Conclusion 65

VI. SURFACE WAVELETS : ARBITRARY MESHES 67

Vertex Encoding, Traversing and Triangulating 67
Lazy Wavelets: A First Look at Surface Wavelets 74
Disc Wavelets : The Sophisticated Surface Wavelet 80
Surface Compression and Applications 87

VII. CONCLUSIONS AND FUTURE WORK 89

APPENDIX

A. A SIMPLE WAVELET TRANSFORMER 92

B. A SAMPLE SUBDIVISION CONNECTED MESHING 97

BIBLIOGRAPHY 100

LIST OF TABLES

Table Page

1. The Vertices of the Example Meshing 82
2. The Vertices for the Face Under Consideration at Level $j-1$ 84

LIST OF FIGURES

Figure Page

1. A Multiresolution Analysis using Multistage Filters 21

2. The Haar Scaling Functions for $V^0, V^1 and V^2$ 22

3. The Haar Wavelet Functions for $W^0 and W^1$ 23

4. The $Daubechies_4$ Wavelet Function for W^0 26

5. The $Spline_{2\_4}$ Wavelet and its Bi-orthogonal Pair Functions for W^0 . . 26

6. The Non-Standard Image Decomposition 32

7. Two Levels of Decomposition of the Image on the Left, Using the $Daub_2$ Wavelet. 33

8. Two Levels of Decomposition of the Image on the Left, Using the $Daub_2$ Wavelet. 34

9. Two Levels of Decomposition of the Image on the Left, using the $Daub_2$ Wavelet. 35

10. One Level of Decomposition of the Chess Board and Spheres Image, Using the $Daub_2$ Wavelet and half the Number of Co-efficients. . . . 36

11. Two Levels of Decomposition of the Chess Board and Spheres Image, Using the $Daub_2$ Wavelet and 1/4 the Number of Co-efficients. . . . 37

12. The End Interpolating B-spline Basis Functions for V^0 to V^2 44

Figure Page

13. Sweep Extraction: The Curve on the Left is a 256 Point Analytic Circle, with Random Perturbations Imposed on it. The Curve on the Right is the Approximation Using 16 Points. The Basic Circular Sweep is Clearly Visible. 45

14. Sweep Extraction: The Innermost Curve is the same 256 Point Analytic Circle, with Random Perturbations Imposed on it. The Surrounding Curves are the Unnormalized Approximations, using 128, 64, 32 and 16 Points 51

15. An Example of a Structured Surface Mesh 53

16. An Example of an Unstructured Surface Mesh. (Picture Courtesy Aerospace Engg Dept, Indian Institute of Technology, Bombay, India) 54

17. One Level of Subdivision, the Splitting Used is a 4:1 Split on Each Edge. 56

18. The Voronoi Partitioning, and Delauney Triangulation Scheme to Extract the Base Meshing from an Arbitrary Meshing M. (Pictures Shown are Modified from the thesis by Lounsbery). 66

19. The 6-tuple Vertex Encoding for a Point in a M^2 Meshing (the Thick Lines Denote the Base Mesh). 69

20. An Example of an Edge at Level j and $j+1$. Note the Parent-Child Relationship and the Correlation with the Level of Subdivision . . . 71

21. The Polyhedral Base Mesh for the Example Surface. The Face with the Vertices Numbered will Constitute Our Example 74

22. The Top Section of the Example Surface with Shading Turned On . . . 75

23. The Wire Mesh Version of the Top Section of the Example Surface with the Vertices Marked 76

24. The Top of the Polyhedra at Level j Flattened Out. The Bold Lines are the Edges of the Base Mesh. 81

Figure Page

25. The Central Face of the Base Meshing at Level $j-1$ 83

26. The Example Surface at Level $j-1$ 85

27. The Wire Mesh Version of the Surface at Level $j-1$ with the Vertex Locations Clearly Pointed Out 86

CHAPTER I

INTRODUCTION

> The term wavelet was coined by Ricker in 1940 and was used to describe the disturbance that moves outwards after seismic activity.

The requirements for modeling the complexity of real world objects are dictated by the context of the model, and vary from computational efficiency to mathematical accuracy. This leads to an inherent trade-off, as design decisions needed to maximize one of these goals lead to a poor showing in the other.

One approach to this trade-off is to have a multi-resolution modeling framework that can be used to select between maximum accuracy and maximum computational efficiency (i.e., simplest approximation) without having to maintain multiple copies of the object being modeled. That is where wavelets enter the scene.

The Wonderful World of Wavelets

Active wavelet research can be traced back to the 1930's. During that period of time the prevalent analysis tools were based on the Fourier technique invented by Joseph Fourier in 1807. Having dealt with orthogonal systems and Fourier series convergence, mathematicians gradually moved along the path of frequency analysis (or decomposition) to scale analysis. The first mention of what later came to be known as wavelet functions was made in the appendix to the thesis by Alfred Haar in 1909. Haar constructed a compactly supported orthonormal basis that was used to lay the foundations of modern wavelet theory.

Around the same period Paul Levy used the Haar basis functions to study Brownian motion. Between 1960 and 1980 mathematicians Guido Weiss and Ronald Coifman searched for the simplest components of function spaces and the assembly rules for reconstruction of all components of these spaces using these simple components. The field of wavelets got a new lease when Grossman and Morlet in 1982 used previous results in scale analysis to construct a mathematical model of Ricker's seismic disturbances and went on to broadly define wavelets in the context of quantum physics.

The latest thrust came with Stephane Mallat [24] in 1985 and his proposition of the pyramid algorithm. His work in digital signal processing resulted in the discovery of many relationships between quadrature mirror filters, orthonormal bases and pyramid algorithms. Following up on his work, Y. Meyer [25] constructed the first non-trivial wavelet (i.e. non Haar wavelet) which, unlike the Haar, was continuously differentiable. The main shortcoming of the Meyer wavelet was the lack of a compact support. The last piece of the puzzle was filled by Ingrid Daubechies with the construction of a wavelet that had both compact support and an orthonormal basis, making it the most preferred wavelet for the signal/image processing community.

The early nineties was the period that the computer graphics community jumped on the wavelets bandwagon. Investigations were conducted into the usage of wavelets for multiresolution representation and editing of curves and surfaces for a variety of applications. In 1991 Hanrahan et al [17] [28] [32] demonstrated the usage of wavelets to solve the global illumination problem. Adam Finkelstein and David Salesin [11] applied B-spline wavelets to create a multiresolution curve representation system for PostScript documentation. Around the same time Matthais Eck, Tony Derose, Tom Duchamp and Michael Lounsbery [7] [4] combined efforts to create a multi resolution surface representation for an arbitrary triangulated meshing.

Simultaneously, in early 1995, Wim Sweldens [37] formulated the lifting scheme for an efficient implementation of the wavelet transform based on bi-orthogonal wavelets. Later Zorin, Schroeder and Sweldens [39] investigated yet another technique that employed spherical wavelets for editing multiresolution surface representations.

Why Wavelets

The classical orthonormal bases that are used in applied mathematics as analysis tools use Fourier or Fourier like bases. The pervasiveness of Fourier techniques often let us ignore the big shortcoming that results from using non-local bases. For a complete representation of a signal, all the elements of the basis are active all the time. Scientists have long sought a basis that is localized in time and frequency (unlike the Fourier which is localized in frequency alone). Gabor and Slepian [14] proposed some adaptations that could be made to Fourier techniques to provide localization by using a windowed Fourier transform. Wavelets instead allowed a basis function that was orthonormal and localized in both time and frequency. Two of the main advantages of using wavelet based techniques over Fourier based techniques are

- *Spatial/Temporal Coherence*: The main win of the Wavelet Transform over the Fourier Transform is that for non-stationary signals it provides a correspondence between a feature in the transformed signal and the spatial/temporal location of the feature in the original signal.

- *Finite domain definitions*: Since the Wavelet Transform is based on functions defined in a finite domain, the error due to an approximation is predictable and measurable (at least in the least squares sense) which is not the case of the Fourier, as it has an infinite series as the basis function.

In the next five chapters we will study the mathematical basis of the wavelet transform and look at the modifications that are needed to apply the theory to the areas that interest the computer graphics community the most: images, curves and surfaces. The structuring of the topics is based loosely on the sequence followed by Stollnitz et. al. [35] in their book "Wavelets for Computer Graphics".

CHAPTER II

THE MATHEMATICAL BASIS

Here we shall take a first look at the components that go into the building of a wavelet transformer. We examine the mathematical basis on which the wavelet transform has been built, the qualities that make this desirable, and a framework to design and construct a wavelet transformer. In the end we shall see how the steps of the wavelet transform can be implemented for the simple case of the Haar basis wavelet transform. We finish by introducing some popular wavelets and discuss some standard construction techniques.

Mathematical Preliminaries

Before we delve into wavelet theory, let us review some mathematical results that will prove to be useful. At this point we assume a good working knowledge of elementary geometry, Fourier transforms, and time-frequency transformations. We begin by defining the basic terms and then draw analogies to the Fourier representation and Cartesian geometry.

Some Definitions

Bases

Let $u_1, u_2,...,u_n$ be a collection of vectors in some space V.

The vectors are said to be *linearly independent* if $c_1u_1 + c_2u_2 + \cdots + c_nu_n = 0$ is satisfied if and only if $c_1 = c_2 = \cdots = c_n = 0$.

Such a collection of $u_1, u_2, \cdots, u_n$ which are linearly independent is said to be a *basis* for the space V, as all vectors v, v ∈ V, can be constructed as a weighted linear combination of the vectors $u_1, u_2, \cdots, u_n$

$$v = \sum_i c_i u_i. \tag{II.1}$$

In the Fourier case, think of V as the infinite line of possible frequencies and each u_i as the amplitude of frequency i present in the signal being studied. Then the complete signal, f(x), can be expressed as the linear combination [21]

$$f(x) = \sum_i a_i cos(ix) + b_i sin(ix). \tag{II.2}$$

Thus the sinusoid is the basis function in the Fourier function space.

Inner product

The *inner product* is basically a generalization of a more familiar case of the dot product, which is used to measure the projection of one vector on another, in the

IR^3 space.

If the space was the IR^3 then the three co-ordinate axes would be the bases, and the dot product of any pair of axes would be zero, as they have no projection on each other. In other words, the vectors representing the co-ordinate axes are linearly independent and therefore have the mutual dot products equal to zero.

In the general case, the dot product $< .|. >$ on the space V is a map from $V \times V$ to IR, such that the mapping is

1. symmetrical $< v|u > = < u|v >$
2. bilinear $< au + bv|w > = a < u|w > +b < v|w >$

   and
3. positive definite $< u|u > > 0$, for all u, $u \neq 0$.

In multiresolution analysis, the *standard* inner product is commonly used, which is defined to be

$$< f|g >= \int_0^1 f(x)g(x)dx \tag{II.3}$$

The solution to the right hand side of this expression has to satisfy the stated conditions to be an inner product and covers the space of all square integrable functions.

Orthogonality

The above discussion about the *inner product* is to formalize the idea of *orthogonality*. Vectors v and u are said to be *orthogonal* if the inner product on the space where u and v are defined is equal to zero:

$$< u|v >= 0. \tag{II.4}$$

Thus if the vectors $u_1, u_2, \cdots, u_n$ are mutually orthogonal, then they are *linearly independent*. For example the three axes in the cartesian space, that are the basis for all vectors in the space, are orthogonal as they all have mutual dot-products equal to zero.

A point to be noted is that orthogonal vectors in a space implies that the two vectors have no overlapping regions of support. If a function is represented using such vectors as a basis, then there is no redundant overlapping data in the representation.

Normalization

A *norm* is a function that measures the length of vectors in a given vector space. Since norms can be constructed to an arbitrary order, the notation used is to write the norm as L_p, where $p =$ order of the norm.

Thus in the familiar Cartesian space, the method to measure the length of a vector u is

$$\| u \| = < u.u >^{1/2} \tag{II.5}$$

This is the L_2 norm under the space.

Normalization, therefore, is the process of measuring the value of $\| u \|$ within the range $0 \cdots 1$. In other words, the process of measuring the weight of a vector, relative to the space in which it resides.

Orthonormal

A basis composed of vectors that are normalized in the L_2 norm, and are orthogonal to each other, are said to form an *orthonormal* basis. Mathematically it is defined as

$$< [u_i] | [u_j] > = \ \delta_{ij} \tag{II.6}$$

where $[u_i]$ and $[u_j]$ are the matrices composed of vectors $u_i, u_2, \cdots, u_n$ and δ_{ij} is the *Kronecker delta.*[1]

Discrete Wavelet Transformation

The wavelet transformation aims to find a nested set of vector spaces that approximate the original function with increasing coarseness.[2] The detail that is

---

[1] The Kronecker delta is defined as a function, that is one for i=j, and zero everywhere else.

[2] Basically an optimal low pass filter.

excluded in the coarser space is captured by a complement space to the coarser[3] one. Put another way, the wavelet transform splits a sequence into two channels. One of the channels contains the low frequency band of the signal and the other channel contains the high frequency band that was left out in the first channel. Further decompositions repeat this step on the low frequency band till the desired level of decomposition is achieved.

Since the basic idea behind the wavelet transform is to be able to build a 2-channel sub-band encoding, it necessitates the construction of a *smoothing* or *scaling* filter (denoted by ϕ) and a *detail* or *wavelet* filter (denoted by ψ). The transformation itself is lossless and reversible. There is no gain in entropy inherent in the transformation.

Let us see how this compares to the well known Fourier transform. The Discrete Fourier transform takes a sequence that is represented in the spatial (or temporal) domain using amplitudes of discrete distance (or time) as the basis function. The transformation represents it as a sequence of amplitudes on the infinite line of discrete sinusoids. Recall that the sinusoid is the basis vector for the Fourier space [27].

Mathematically the sequence $a_1, a_2, \cdots, a_n$ in the temporal/spatial domain could be written as

$$x[n] = \sum_{t=-\infty}^{\infty} a[t]\delta[n-t]$$

---

[3] This is basically a high pass filter.

with t as the basis vector. The discrete Fourier space representation of this sequence is

$$x[n] = 1/N \sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn}$$

and the components $X[k]$ are defined by the Inverse Fourier Transform

$$X[k] = \sum_{k=0}^{N-1} x[n] e^{-j(2\pi/N)kn}$$

where $2\pi/N$ is the principle frequency and uses the sinusoids $cos(2\pi/N)kn$ and $sin(2\pi/N)kn$ as the basis vectors.

Instead the Discrete Wavelet transform represents the transformed signal using two sequences. The first sequence is the list of amplitudes for a basis that has exactly half the number of basis vectors as the original sequence, and approximates the original function closely. The second sequence also is a list of amplitudes for exactly half the number of basis vectors and captures the detail information that has escaped the first sequence. Therefore the sequence $a_1^j, a_2^j, \cdots, a_n^j$ gets transformed to the two sub-band sequences $a_1^{j-1}, a_2^{j-1}, \cdots, a_{n/2}^{j-1}$ and $b_1^{j-1}, b_2^{j-1}, \cdots, b_{n/2}^{j-1}$ which are the co-efficients for the basis vectors for the scaling and wavelet function spaces respectively.

Here then is where the wavelet transform differs from the Fourier transform, because unlike the Fourier, which has as its basis the sinusoid, the wavelet bases are defined recursively [8] [1] as a set of scaling functions spanning the space V^j

$$\phi^j(x) = \sum_{k=0}^{M-1} c_k \phi(2^j x - k) \tag{II.7}$$

and a set of wavelet or detail functions that are orthogonal to the dilations or scales spanning the space W^j

$$\psi^j(x) = \sum_{k=0}^{M-1} -1^k c_{1-k} \phi(2^j x - k) \tag{II.8}$$

where V^j and W^j together make up the function space of the original data (i.e. V^{j+1}). The function ϕ is the basic function that is used to design all the basis functions and is sometimes referred to as the *mother wavelet.* Thus this combination of the sequences $\phi^j(x)$ and $\psi^j(x)$ and the corresponding co-efficients c_k at level j represent the signal completely.[4]

Note that the definition of the basis functions as shifted (or translated versions) of the basic function ϕ provides the localization in space, and the definition of the basis functions at various levels, called dilations, provide, the localization in frequency.

---

[4] The c_k terms in the wavelet function definitions have negative subscripts. The negative terms are to be taken in a mod sense.

Multiresolution Analysis

There are many ways to approach the study and construction of a wavelet transformation. Since *multiresolution analysis* is the main focus of this work, let us take a look at the world of wavelets from that perspective [24].

Nested Spaces

For the multiresolution approach the starting point is to define the vector spaces in which we are interested. This is the space that will be spanned by all the functions that are defined for this space (in other words this is the domain for the functions), and all values in this space will be defined in terms of the basis vectors for the space.

By definition, multiresolution analysis dictates that for any given space V^j there has to exist nested spaces[5] such that

$$V^0 \subset V^1 \subset V^2 \cdots \subset V^j \subset \cdots \subset V^n. \tag{II.9}$$

These spaces are function spaces for the above mentioned *scaling functions* of Equation II.7. In other words, the scaling functions are the basis functions for these spaces.[6]

---

[5] Even though the scaling and wavelet functions can be defined on the infinite real number line, most definitions of the wavelet and scaling functions are presumed to be on the bounded domain; i.e. they have a finite set of basis functions for V^j and thus also for the spaces W^j.

[6] the superscript j used in the definition of the scaling functions, corresponds to the space for which they are the basis functions

The next step is to define the *wavelet space*, which is the space over which the wavelet functions form the basis functions as per Equation II.8. The space W^j is defined as the space contained in space V^{j+1} and orthogonal to the space V^j

$$V^{j+1} = V^j + W^j. \tag{II.10}$$

The point to note here is that the definitions of both the scaling functions and the wavelet functions are recursive and this will mean higher and higher resolution at each discrete step from V_j to V_{j+1}. This property is also referred to as *refinability.*

Orthogonality & Inner Products Constraints

The most efficient usage of this transform would be to make the spaces V^j and W^j orthogonal to each other. That would imply linear independence, which in turn would imply that the spaces V^j and W^j are not overlapping each other in the portion of V^{j+1} that each of them captures.

The tightest constraint dictates that

$$< \phi^j_k | \phi^j_l > = \delta_{k,l} \tag{II.11}$$

$$< \psi^j_k | \psi^j_l > = \delta_{k,l} \tag{II.12}$$

$$< \phi_k | \psi_l > = 0 \tag{II.13}$$

where $\delta_{k,l}$ is the Kronecker Delta. Intuitively, we can see that the idea is to make the most of each and every co-efficient that is used to represent the function. Equations II.11 and Equation II.12 simply ensure that the scaling functions spanning the space V^j and the wavelet functions spanning the space W^j do not overlap in their respective regions of support. The last Equation II.13, however is more than just a direct consequence of defining the wavelet spaces to be orthogonal to the scaling function spaces. The tightest constraints require that all the scaling functions for all levels are orthogonal to all the wavelet functions for all levels, and not just for the same level.

As we shall see later this condition is relaxed for most implementations of wavelet functions. The typical approach is to redefine the equation to imply that the scaling functions are orthogonal to only the wavelet functions defined on the same level j.

Normalization

The need for normalization of the wavelet co-efficients arises because the transformation induces a scaling on the scaling and wavelet function co-efficients.

Certain applications, as we shall see later, need to be able to select wavelet functions that would be excluded from a wavelet transformed representation of the function. Since the only way to make this comparison is to compare the normalized versions of the co-efficients, easy computation of the L_p norm is a big advantage.

Coveted Properties

We have thus far outlined the guidelines under which the scaling functions ϕ and ψ can be created. Before taking the next step of looking at a real implementation of those rules, let us list the main features we want for the functions [24].

1. Refinability

   This is a basic requirement of the wavelet transform. The scaling functions are defined so that they can be refined. This forms the premise of the multiresolution representation because going to a lower or higher level of detail is merely an exercise in choosing the right level of refinability in the scaling functions to use as the basis for the interpolation. This is also known as having *nested spaces.*

2. Compact localized support

   The scaling functions should have compact support. Compact support implies that the filter functions for the decomposition and reconstruction are sparse and easy to compute. The role of a filter in the wavelet transform is explained in Section (II.5).

3. Symmetry

   Usage of filters that are symmetric around the centers is essential for certain operations so as to not bias the next level in a particular direction.

4. Smoothness

   Since the final function will be represented using the scaling and wavelet functions as the basis, a smooth basis will yield smoother final interpolations.

5. Vanishing moments

   A wavelet $\psi(x)$ is said to have n vanishing moments, if the value of

$$\int \psi(x)x^k dx = \begin{cases} 0 & \text{for } 0 \leq k \leq n-1 \\ non-zero & \text{for } k = n. \end{cases}$$

   This guarantees that polynomials up to degree n are locally supported within the multiresolution spaces, and also that the wavelet function ψ satisfies C^N continuity.

The Filter Bank Algorithm

The filter bank algorithm was proposed by Stephane Mallat in 1985 for his work on image pyramids and forms an excellent platform for multiresolution analysis.

Single Stage Filter

Let s_i^{j+1} be the co-efficients of all the scaling functions ϕ_i^{j+1} defined at the level $j+1$. Then the *analysis filters* A^j and B^j, which decompose the space V^{j+1}, are defined as

$$\begin{tikzcd}
s_1^{j+1} & s_1^j \\
\vdots \arrow[r, "A^j"] \arrow[rdddd, "B^j"] & \vdots \\
s_n^{j+1} & s_{n/2}^j \\
 & -- \\
 & d_1^j \\
 & \vdots \\
 & d_{n/2}^j
\end{tikzcd} \tag{II.14}$$

where s_i^j are the co-efficients for the i scaling functions and d_i^j are the co-efficients for the i wavelet functions[7] defined at level j.

Recall that the recursive (i.e nested) definition of the scaling and wavelet functions makes it possible to transform between the functions at different levels, and filter functions A^j and B^j are the formulation of the decompositions.

Since the above mentioned decomposition step allows us to break the original signal down to two different parts, the inverse transform would have to be able to take the scaling function co-efficients and the wavelet co-efficients, and add them back together to reconstruct the original sequence. The filters P^j and Q^j that implement this step are called the *synthesis filters* and are defined as

---

[7] Henceforth the representation of a signal will be given in terms of the co-efficients s_i^j and d_i^j, as the underlying scaling and wavelet functions are always of a standard form, and do not need to be explicitly specified.

$$\begin{array}{ccc} s_1^j & & s_1^{j+1} \\ \vdots & \longrightarrow^{P^j} & \vdots \\ s_n^j & & s_{n/2}^{j+1} \\ -\,- & & \\ d_1^j & & \\ \vdots & \nearrow^{Q^j} & \\ d_{n/2}^j & & \end{array} \qquad \text{(II.15)}$$

The filter functions A^j, B^j and P^j, Q^j are typically defined as matrices that operate on arrays of s_i^j and d_i^j.

For a transition from a known level $j+1$ to another level j or vice versa, the analysis and synthesis filters have a definite relationship [1] which can be expressed as

$$\begin{pmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{pmatrix} = \begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix}^{-1} \qquad \text{(II.16)}$$

where A,B,P and Q are all denoted as matrices.

Matrix Construction

Since matrices constitute a compact and easy way to compute transformations, it forms the most painless solution. However an immediate problem that crops up is that the matrices can be infinite in the limit because the size of the matrix will grow proportional to the level of refinement.

For finite sequences and a given level j, we can construct the scaling filter (P^j) and wavelet filter (Q^j). Elements of P and Q can be expressed as

$$P_{xy} = 1/2c_{2i-j}; \qquad Q_{xy} = 1/2(-1)^{j+1}c_{j+1-2i} \tag{II.17}$$

where c_i, for $i = 0, 1, \cdots, L-1$ are the co-efficients of the scaling and wavelet matrix equations, Equation II.7 and Equation II.8.

Multistage Filter

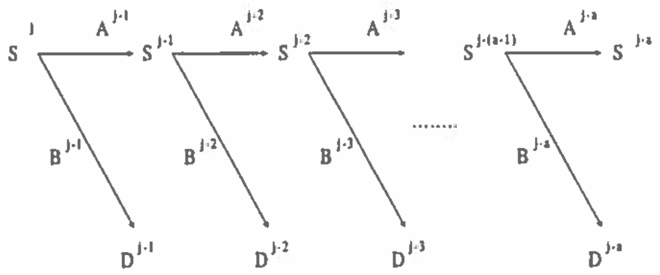
The above formulation would allow the transition from the signal represented in space V^{j+1} to V^j and vice versa. The entire process to step from any arbitrary resolution level j to an arbitrary level $j-a$ would require a cascade of filters each way.

Thus the analysis and synthesis filters would appear as in Figure 1. This combination of Analysis and Synthesis filters is called a *multiresolution analysis.*

Haar Wavelets: A Case Study

Now that we have seen the formulation of a wavelet transformation and discussed a possible framework for its implementation, let us take the simple example of the 1-dimensional Haar function and trace through the theory and construction of

he Analysis Phase Filter
array



The Synthesis Phase
Filter array

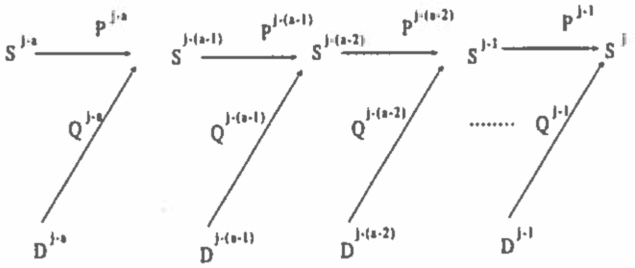


FIGURE 1. A Multiresolution Analysis using Multistage Filters

the Haar basis wavelet transformer.

The main steps that go into designing a multiresolution analysis are :

1. *Selection of the scaling functions.* Consider the function space [0,1] for a 1-dimensional function, and the basis vectors for the space at level j to be defined as 2^j piece-wise linear functions, as shown in Figure 2.

   The equation representing the scaling functions can be written as

$$\phi_i^j(x) := \phi(2^j x - i), i = 0, 1, \cdots, 2^j - 1$$

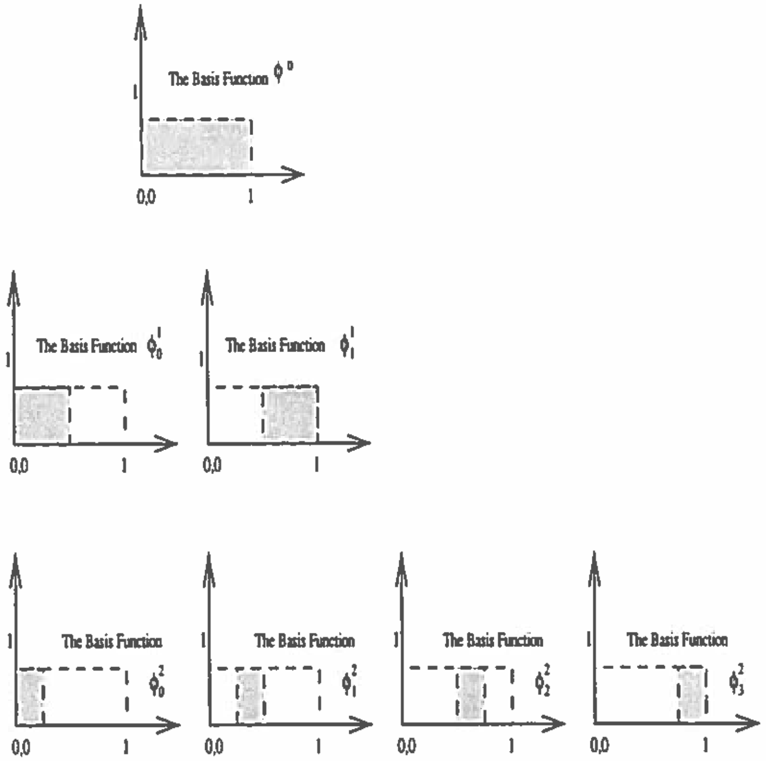

The Basis Function ϕ^0
1
0,0
1
The Basis Function ϕ_0^1
The Basis Function ϕ_1^1
The Basis Function ϕ_0^2
The Basis Function ϕ_1^2
The Basis Function ϕ_2^2
The Basis Function ϕ_3^2

FIGURE 2. The Haar Scaling Functions for $V^0, V^1 and V^2$

where

$$\phi(x) := \begin{cases} 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

which makes it obvious that the space spanned by these scaling functions are refinable. Thus V^j is refinable.

2. *Selection of the inner product* on the spaces $V^0, V^1, \cdots, V^n$. The choice of the inner product is important as this will decide the condition of orthogonality, and thus provide the definition of the wavelet spaces W^j. One important and often overlooked consequence of the inner product is that the inner product also

determines the L_2 norm. The L_2 norm is used to scale the wavelet co-efficients and also to measure error induced by eliminating a particular co-efficient. In this case, we shall select the standard inner product which is defined as per Equation II.3.

3. Solution of the inner product, and *selection of the set of wavelets* that span W^j for each j. Solving for the inner product gets us the solution for ψ which are the sequences as shown in Figure 3.

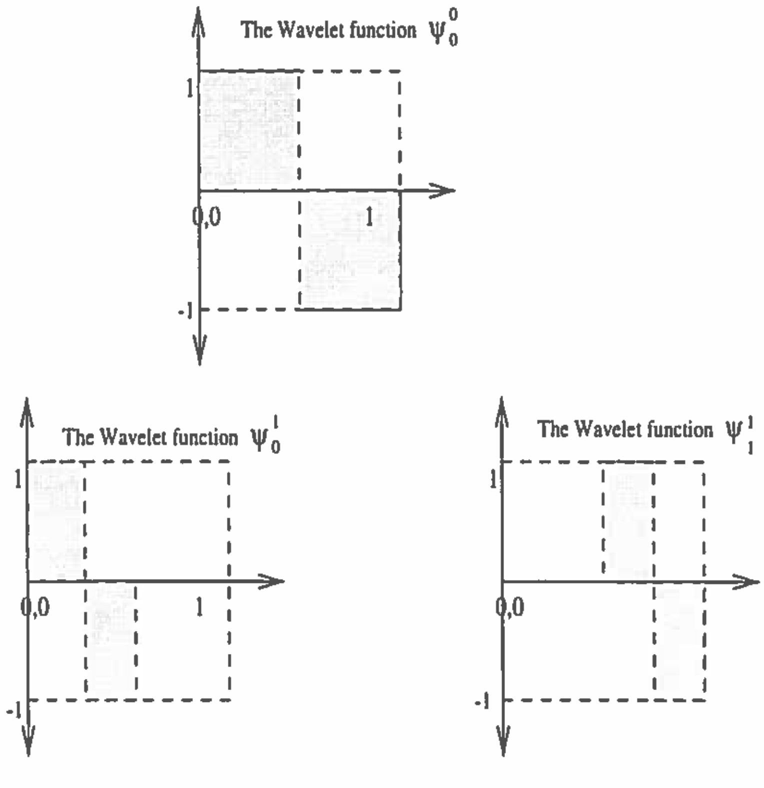


FIGURE 3. The Haar Wavelet Functions for W^0andW^1

$$\psi_i^j(x) := \psi(2^j x - i), i = 0, 1, \cdots, 2^j - 1$$

where

$$\psi(x) := \begin{cases} 1 & \text{for } 0 \leq \text{x} < 1/2 \\ -1 & \text{for } 1/2 \leq \text{x} < 1 \\ 0 & \text{Otherwise} \end{cases}$$

We now see that the members of this sequence are the Haar basis functions, and thus the name Haar Wavelets. This suggests that another way to proceed would be to select the wavelet functions first, and then use the inner product to select the scaling functions.

4. *Design of the filters* that would actually implement the transform for the given set of scaling and wavelet functions.

   Recall that

$$V^{j+1} = V^j + W^j$$

$$V^{j+1}P^{j+1} = V^j$$

$$V^{j+1}Q^{j+1} = W^j$$

   which implies that

$$\phi^{j+1} = \phi^j + \psi^j$$

$$\phi^{j+1}P^{j+1} = \phi^j$$

$$\phi^{j+1}Q^{j+1} = \psi^j.$$

Now we can trivially see that the matrices P^2 and Q^2 needed to convert the scaling functions at level 2 to the scaling functions at level 1 are

$$P^2 := \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \& \qquad Q^2 := \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}$$

From Equation II.16, the rule that relates the analysis filters to the synthesis filters for a given level j, we can derive the A^j and B^j filters.

A result of great importance that we have observed, is that the size of the matrix grows proportional to the data size, which is a great factor in real implementations. A possible solution to that problem has been proposed. This scheme, known as the lifting scheme, has excellent time characteristics for certain wavelets [37].

Other Famous Wavelets

The Haar wavelet is unusual because it is probably the only commonly used wavelet that obeys all the wavelet conditions. As we have seen, for more complex systems the size of the matrices increase as j increases. This is a major hurdle, and non-sparse matrices are not appreciated. Some expediencies are used in certain applications like computer graphics to make close approximations to orthogonality, just to maintain the matrices as sparse.

Consequently many other families of wavelets have been constructed. However almost none of them obey all the laws that were described for construction of wavelets. Instead they make agreeable digressions to gain in area such as interactive speed of implementation, or ease of computation.

The three commonly used methods [36] for generation of wavelet basis functions are :

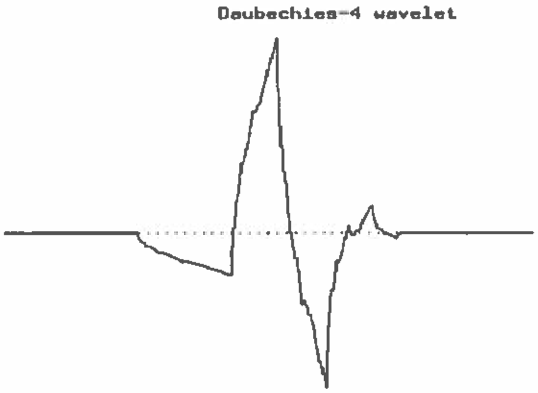


FIGURE 4. The *$Daubechies_4$* Wavelet Function for W^0

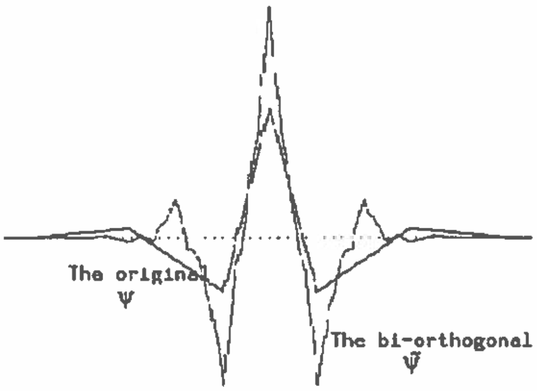


FIGURE 5. The *$Spline_{2\_4}$* Wavelet and its Bi-orthogonal Pair Functions for W^0

1. **Iteration**

   Iteration of the recursive Equations II.7 and II.8 starting with ϕ as a box function and some values of c_k. The splines family can be generated this way. Some common wavelets generated in this manner are :

No.	Wavelet	initial values for c_k
1	box	1, 1
2	hat	1/2, 1, 1/2
3	quadratic	1/4, 3/4, 3/4, 1/4
4	cubic	1/8, 4/8, 6/8, 4/8, 1/8
5	daubechies-4	$\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}$

2. **Frequency definition**

   Synthesis can be done by starting from the Fourier definition of ϕ, imposing constraints on it, and then using orthogonality to get to ψ.

3. **Direct**

   This method will show up again in the usage of wavelets for surface compression and will be discussed in detail there. Stated simply this method says, that if the recursion is easily predictable, you can take the values at the known integer locations and apply the recursion rule to get the values at $i/2^j$.

The plots in Figures 4 and Figure 5 are for some wavelets that have been used in this thesis. The Daubechies wavelet [6] has seen considerable usage in the signal/image processing community because it is easy to synthesize and also has the much coveted properties of compact local support and orthogonality. On the other hand, the B-spline wavelets have had a lot of exposure within the computer graphics community as it yields a smooth multiresolution analysis.

CHAPTER III

2-D WAVELETS : IMAGES

Now that we have a handle on the mathematical foundation of the wavelet transform, the stage is set to look at the first and obvious application for the graphics community, namely, images. In this chapter we will examine the extension of the transform to two dimensions and the various features and implications that make this transform appealing.

Wavelets in 2-Dimensions

The extension of the wavelet transform to n-dimensions in the general case, and to 2-dimensions in the specific case, is similar to the extensions for the Fourier transform. Since the basis functions for the wavelet transform are orthogonal, the rules of separability [20] enable the 2-dimensional data to be computed as a pair of unitary[1] transforms.

For the remainder of the discussion, keep in mind that the basic idea is to take the image and break it down by decomposing it n times to get a base image that is

---

[1] This means that data can be considered in each dimension as 1-dimensional data, and the results can be simply added together.

2^n times smaller than the original, and use the remaining memory space for the level of detail co-efficients at the n intermediate levels.

Standard Versus Non-Standard Decomposition

There are two popular methods used to separate the unitary transformations. The first is to do the transformation for all the rows, then take the result and transform all the columns. This is called the standard decomposition. The alternative method used is to take the transform of the image along all the rows once, then to take the image and transform along the columns once. This alternating pattern is continued until the number of decompositions required is met.

Standard

To obtain the standard decomposition[2] of an image, the functions used as basis functions are the tensor product of all the one-dimensional basis functions at that level. Therefore the basis functions now become $\phi(u) \times \phi(v)$ (the scaling function), $\phi(u) \times \psi(v)$ (a wavelet function), $\psi(u) \times \phi(v)$ (a wavelet function) and $\psi(u) \times \psi(v)$ (a wavelet function)

The advantage of this approach is that a single one-dimensional transform in each direction is all that is needed. The biggest disadvantage that this suffers from is that the resulting basis functions for this method tend to have non-local support.

---

[2] This discussion is included for closure on this topic, for a better discussion refer to [24] and [1]

This makes them inefficient for use in applications needing real time performance as inverting a non-sparse matrix in real time is highly compute intensive.

Non-standard

This is a preferred method for most applications of wavelet image transformation. The basic steps are :

1. Perform one step of horizontal pairwise smoothing, and differencing of the pixels in each row.
2. Take the result and apply the same filters to a transpose of the image.[3]
3. Repeat steps 1 and 2 recursively on the quadrant containing the smoothing basis in both directions, until the desired level of decomposition has been achieved.

The 2-dimensional basis functions can be constructed by first defining the 2-dimensional functions :

$\phi\phi(x, y) = \phi(x)\phi(y)$ (scaling function),

$\phi\psi(x, y) = \phi(x)\psi(y)$ (wavelet function),

$\psi\phi(x, y) = \psi(x)\phi(y)$ (wavelet function) and

$\psi\psi(x, y) = \psi(x)\psi(y)$ (wavelet function).

The main advantages of this approach are:

---

[3] That is equivalent to performing the operations on the pixels in the vertical direction

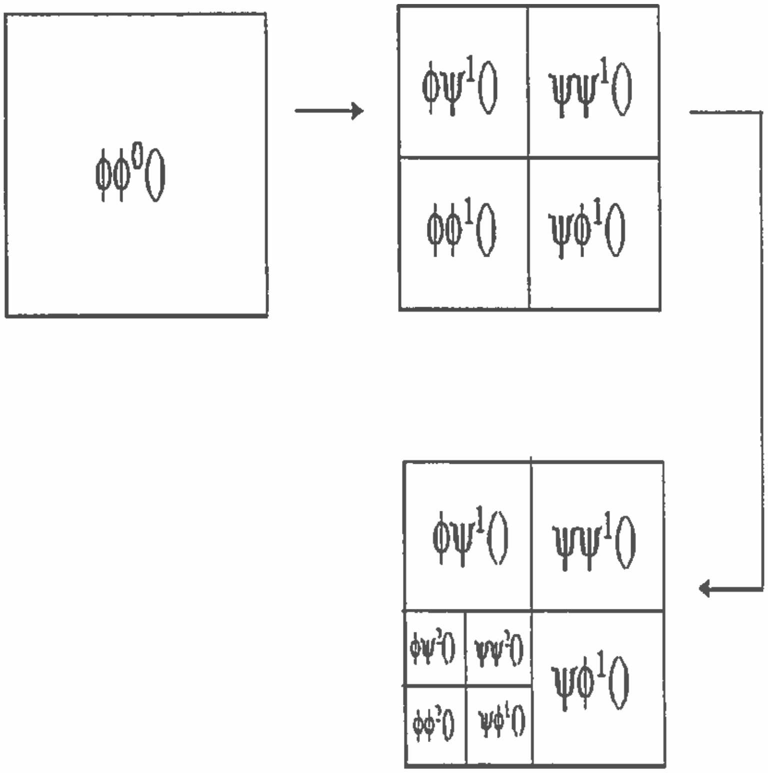


FIGURE 6. The Non-Standard Image Decomposition

1. The basis functions all have square support, whereas in the standard method only some have square support.
2. The intermediate images are ready for presentation after normalization. This makes this method ideal for progressive display.
3. The transformation converges faster, as each recursive step works on a quarter of the data, instead of half as in the case of the standard transform.
4. Another interesting consequence is that all the quadrants (other than the one that will be recursively decomposed) will for certain wavelets (e.g $Daubechies_4$)

act as directional edge detectors as shown[4] in Figure 9.

The main disadvantage of this approach is that the input image needs to be transposed once for each level adding to cost overhead.

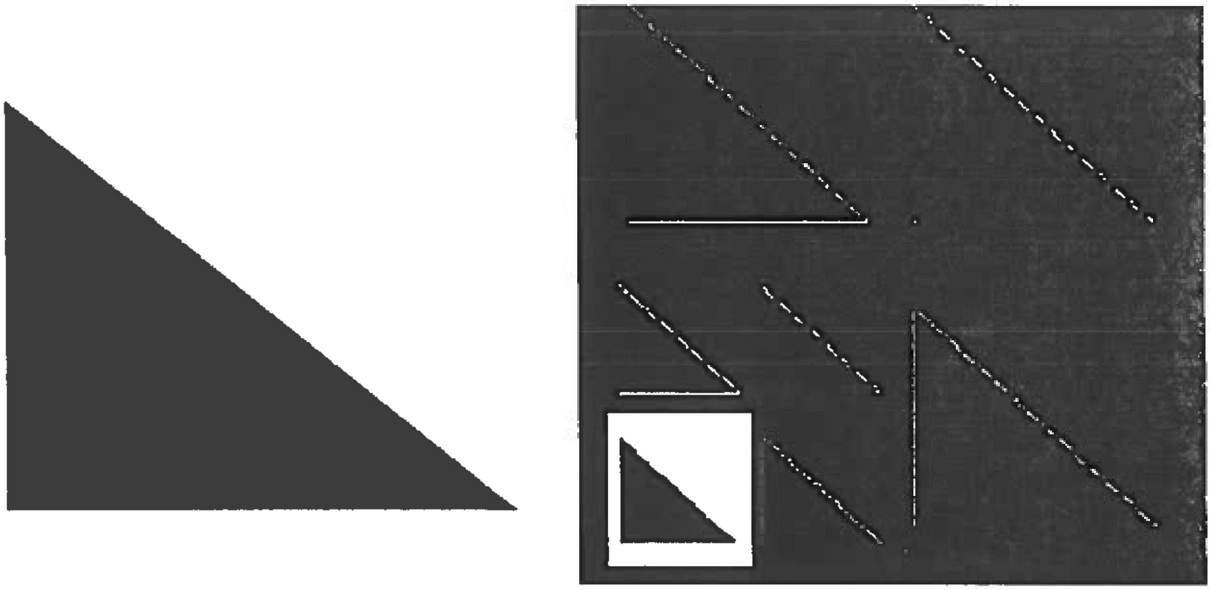


FIGURE 7. Two Levels of Decomposition of the Image on the Left, Using the $Daub_2$ Wavelet.

Applications

The most useful applications are in the fields of compression, multiresolution editing, and restoration.

---

[4] All the transformed images are shown with un-normalized co-efficients, and values clamped between 0 and 255.

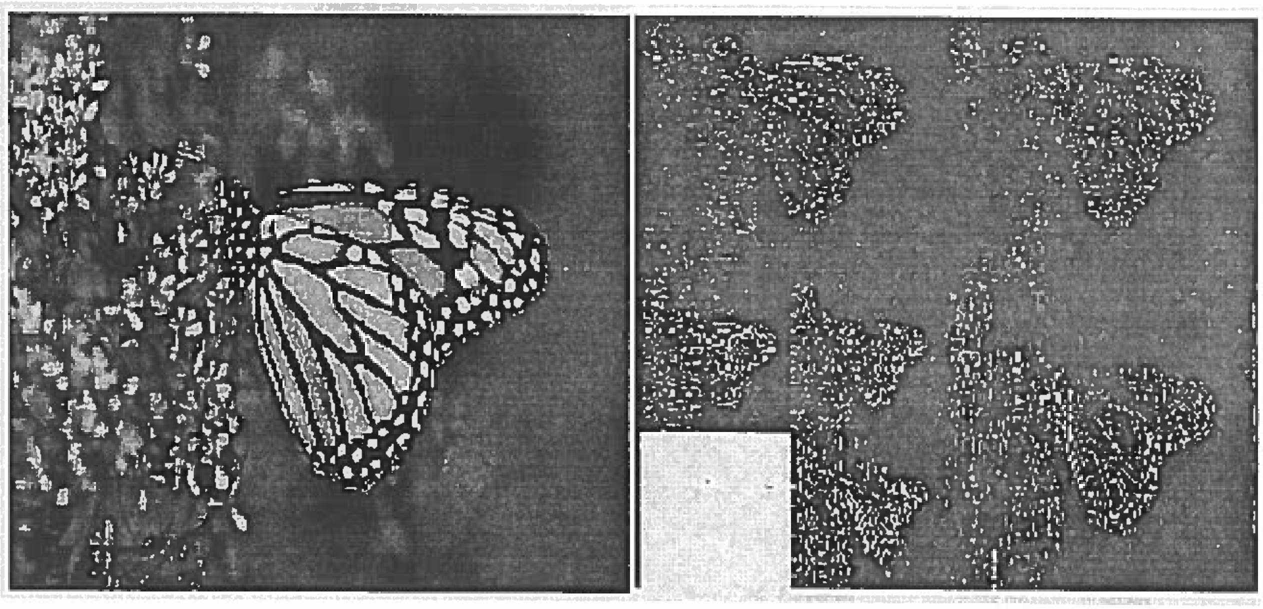


FIGURE 8. Two Levels of Decomposition of the Image on the Left, Using the $Daub_2$ Wavelet.

Wavelet Based Image Compression

One consequence of using fully orthogonal basis functions is that the contribution of a co-efficient to the overall image is easily computed as the normalized value of the co-efficient under the L_p norm for that wavelet space. This implies that the error under the L_p norm induced by leaving out a particular wavelet co-efficient is easy to compute, and can be computed for minimal overhead, as it is simply the sum of the normalized co-efficients.

The normalized wavelet co-efficients can be sorted and used in the reconstruction based on the tolerance that is prescribed for the image.

The commonly used criterion for selecting wavelet co-efficients to add to the base signal are:

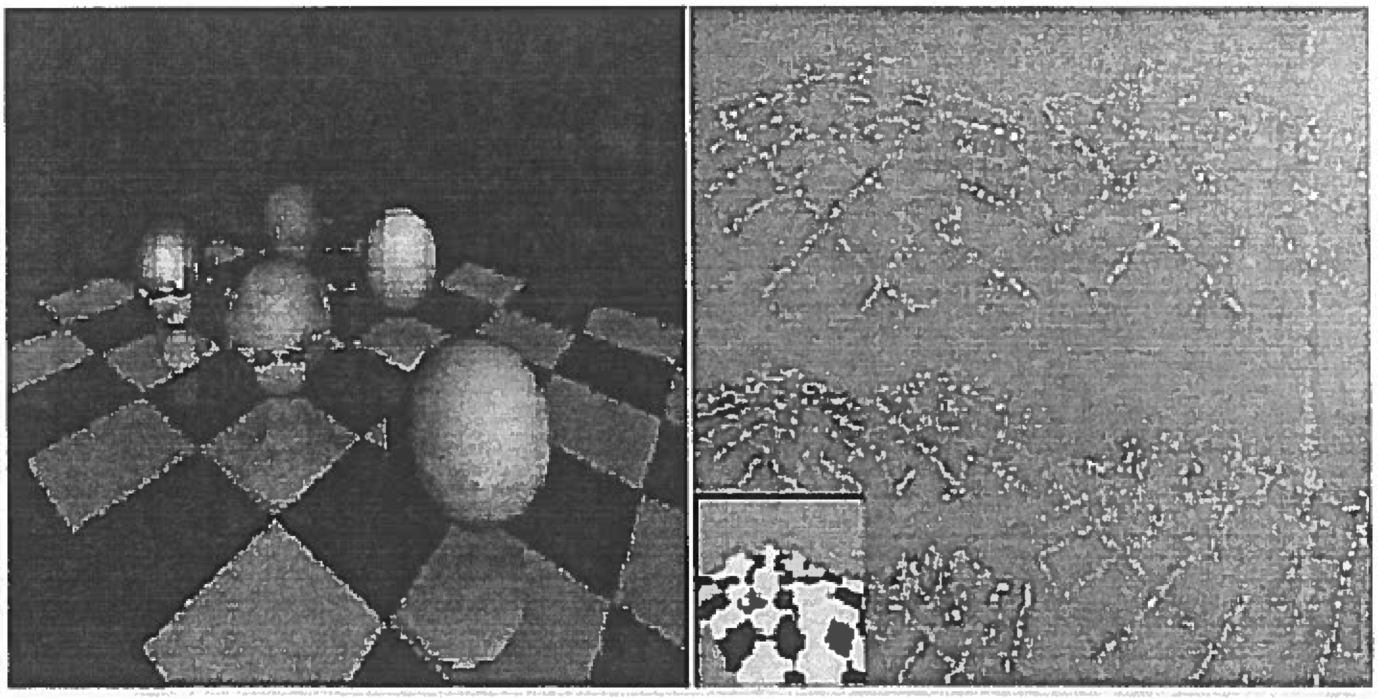


FIGURE 9. Two Levels of Decomposition of the Image on the Left, using the $Daub_2$ Wavelet.

1. L_2 error: Sufficient co-efficients are added to keep the error within a certain bound.
2. Thresholding: Only co-efficients that cross a certain threshold, indicating that they have a certain minimum contribution to the picture, are added back in.
3. Bandwidth: Only sufficient co-efficients are added to fill up the space in bytes that are available to represent a frame.

Wavelet Based Image Editing

Another consequence of orthonormal basis functions is that error introduced at a coarser level, say δx, is carried up by the inverse transform for a fully orthogonal

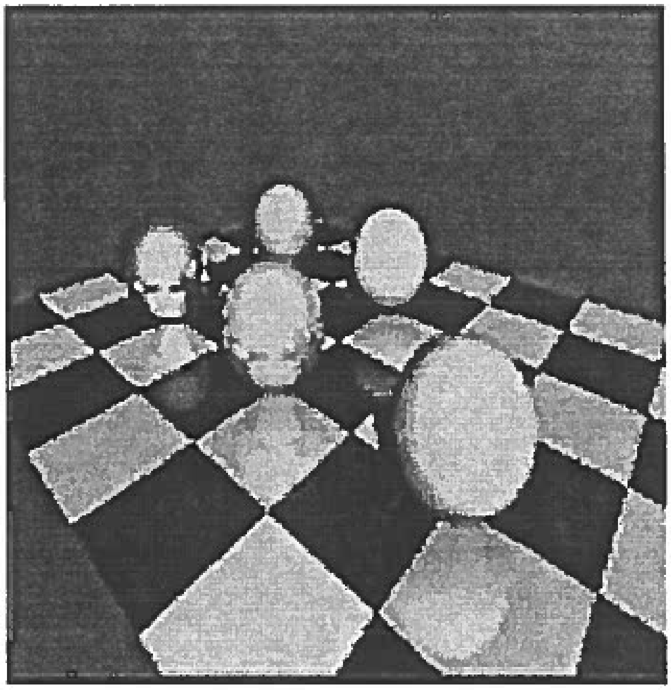


FIGURE 10. One Level of Decomposition of the Chess Board and Spheres Image, Using the *Daub*$_2$ Wavelet and half the Number of Co-efficients.

transform. This results in a predictable spread in the error in the final image.

In a mathematical sense, editing is the same as introducing error into the original signal thereby making this feature very appealing as changes can be made on coarse or fine areas depending on the level of transformation of the edited image.

Combined with the spatial coherence property of the wavelet transform, wavelet transformations thus make an invaluable image editing tool. The image can be taken to a coarser level, and big blocks of image space can be colored to produce a subtle shade or hue that would be impossible to draw directly on to the image.

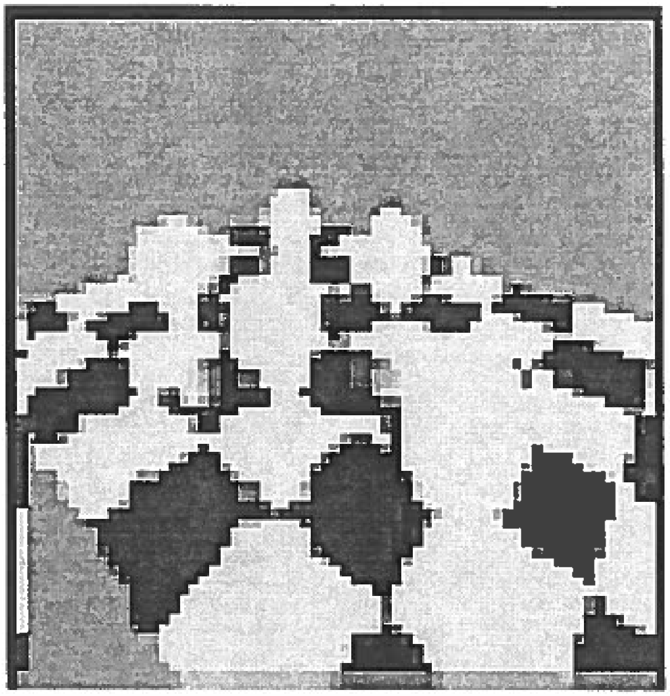


FIGURE 11. Two Levels of Decomposition of the Chess Board and Spheres Image, Using the $Daub_2$ Wavelet and 1/4 the Number of Co-efficients.

The Power of Wavelets

To recap, the reasons for which wavelets have seen a great deal of interest and usage in imaging applications are:

1. Time and frequency localization: Compression for any signal is achieved by removing redundant data. This is accomplished by using correlation to compute parts of the signal that are not needed to represent the signal completely.

   For images these correlations are spatial correlation, spectral correlation and temporal correlation. While frequency based techniques exploit the spectral correlation, wavelets score over them because each co-efficient of the transform represents both the spatial and frequency contribution to the overall signal. This makes it possible to exploit the spatial and temporal correlations also.

2. $O(n)$: There are algorithms for certain wavelet transforms that can be performed in $O(n)$ time, thus gaining in efficiency over the $O(n \, log \, n)$ Fourier algorithms. [30] [2]
3. Flexibility: The wavelet co-efficients can be compared to each other under some L_p norm (which can be computed easily, due to the construction). They can therefore be re-arranged, and added in an arbitrary order.

CHAPTER IV

3-D WAVELETS : CURVES

Now that we have seen the design and application of wavelet based multiresolution analysis in 2-dimensions we shall take the next logical step of extending the results to 3-dimensions. In this chapter we look at the implementation of a class of smooth wavelets (namely the b-spline wavelets) that will be used for the multiresolution analysis of curve data. We will then look at the actual application of the analysis to b-spline curve data, and the use of multiresolution analysis for separation of character and sweep of a curve. Finally we will examine some results of our implementation.

Choosing The Wavelet

The wavelet decomposition of curves and surfaces are computed just like images, by applying the decomposition filters to each of the co-ordinate values separately. This is permitted as the co-ordinates axes are by definition, orthogonal to each other.

Primary importance, however, has to be placed on the construction of the scaling and wavelet functions, as they should be well adapted to the intervals that are being decomposed. The b-spline stands out as an obvious construction for curves and tensor product surfaces as it inherently has the properties of local support and refinement making it the function of choice for the wavelet functions as it is a good match for the intervals, of the control points or control polygons.

For the remainder of this discussion, keep in mind that the construction of wavelets typically begins with laying out the guidelines for the synthesis filter P^j and Q^j, as those are the matrices relating the scaling or wavelet functions at the current level j, to the functions at level $j+1$. The analysis filters are then computed from the synthesis filters using the inverse relationship.

Candidate Wavelets

Let us first understand the properties that a wavelet analysis would have to exhibit to make it a good candidate for approximations to b-spline curves.

Properties

The most important factors that need to be considered for the choice of a wavelet function[1] are:

1. Vanishing moments: Recall from Chapter II, that the vanishing moments of the wavelet function is a measure of the order of the function that it can represent accurately. In the case of multiresolution analysis, we can say that if the analyz-

---

[1] Note that though all the statements are made for the wavelet functions, in reality they imply constraints on the scaling functions. This notation is used as the scaling functions are related to the wavelet functions under the orthogonality rules.

ing wavelet has N vanishing moments, then the polynomial curves (e.g. bezier, the spline family, etc.) of degree up to $N-1$ will be accurately represented by the scaling functions.

2. Symmetry & Smoothness: As already noted, the symmetry and smoothness of the wavelet function is of primary concern because geometric curves and surfaces are inherently smooth. Another related factor is oscillations in the scaling functions. Functions with small oscillations are preferred for obvious reasons.
3. Compact support: Since the region of support of the wavelet function decides the size of the analysis filter matrices, the smaller the region of support the sparser the resultant matrix and the easier the computations.

There is another implication of the last condition for real applications. A non-sparse matrix is also harder to invert making it computationally hard to find the corresponding analysis filters for globally supported wavelet functions (i.e. the non-sparse matrices). The reason that most analysis/synthesis filters, constructed for wavelets satisfying the conditions in Equation (II.11) to Equation (II.13) are non-sparse, is because of Equation (II.13), which states that the orthogonality condition is global. Therefore there has to be an entry corresponding to each possible combination of scaling and wavelet functions, for all possible levels resulting in a non-sparse matrix.

The b-spline basis functions have vanishing moments equal to the order of the b-spline functions used. The basis functions are, for the most part, symmetric[2] and smooth. The local control [12] property of the b-spline basis means that the functions have compact support, thus satisfying all the criterion listed above for a good candidate function.

Geometric multiresolution analysis relies very heavily on being able to complete computations in real time and thus cannot afford to implement a b-spline based wavelet using all the constraints for a wavelet decomposition. Applications typically use wavelets that relax part or all of these conditions. Two types of wavelet functions that are very popular are the Semi-orthogonal and Bi-orthogonal wavelets based on the b-spline basis functions.

Construction of Semi-orthogonal and Bi-orthogonal Wavelets

Semi-orthogonal Wavelets

Mathematically speaking, the case of semi-orthogonal wavelets, also known as pre-wavelets, is one where the condition in Equation (II.13) has been relaxed to

$$< [\Phi^j], [\Psi^j] >= 0 \tag{IV.18}$$

---

[2] The end interpolating b-spline basis has asymmetric basis functions at the end points as shown in Figure 12

namely the scaling and wavelet functions are orthogonal only at the level j and none other. To be noted here is that the orthogonal wavelet is just a special case of the semi-orthogonal one.

Using the filter bank concept and results for Equation (IV.18) we can write

$$\Phi^{j-1} = \Phi^j P^j$$

$$\Psi^{j-1} = \Phi^j Q^j$$

and we can rewrite Equation (IV.18) such that[3]

$$P^{j^T}[< \Phi^j | \Psi^j >] Q^j = 0. \tag{IV.19}$$

Now let

$$M^j = P^{j^T}[< \Phi^j | \Psi^j >]$$

then Equation IV.18 reduces to the homogeneous solution of

$$M^j Q^j = 0. \tag{IV.20}$$

Since this does not yield a unique solution, additional constraints are imposed upon the possible solutions to narrow the possible candidates. The most common criterion

---

[3] where $[\Phi]$ and $[\Psi]$ represent the matrices that are comprised of the individual functions ϕ_i^j and ψ_i^j for all values of i and j.

used are symmetry and local support.

The B-spline Wavelets

A specific example of the family of semi-orthogonal wavelets, namely the b-spline wavelets, has been favored by the graphics community as it defines smooth basis functions.[4] The wavelets used for multiresolution curve representation are defined on the basis functions for the end-interpolating b-spline curves as shown in Figure 12 [11].

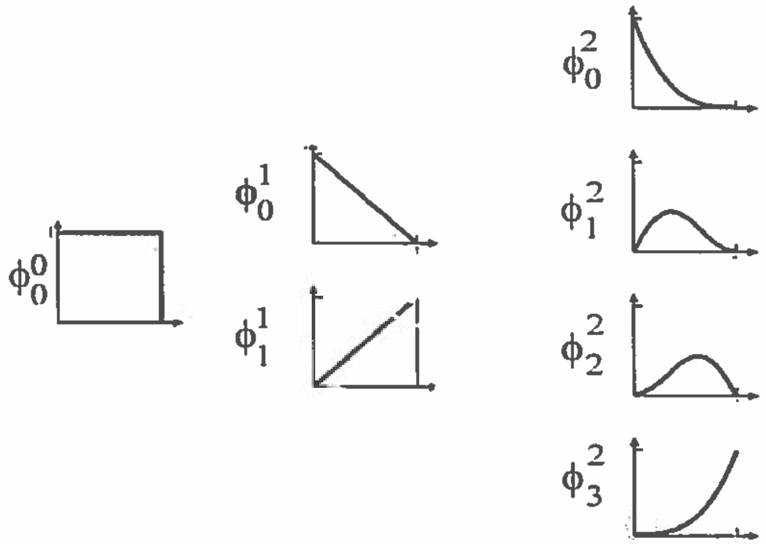


FIGURE 12. The End Interpolating B-spline Basis Functions for V^0 to V^2

The first step in the construction of the wavelets is to understand Equation (IV.20). Let M^j represent the parameters that are chosen and Q^j represent the unknown parameters. M^j consists of the scaling functions and the part of the synthesis filter that relates the scaling functions of one level to the scaling functions of the higher

---

[4] Chui et. al. [5] constructed a whole class of wavelets based on the b-spline basis functions.

resolution level (i.e P^j). For the b-spline case, we pick the scaling functions to be the basis functions for uniform end interpolating b-splines[5], and P^j to be the Cox-de Boor knot insertion algorithm [9].

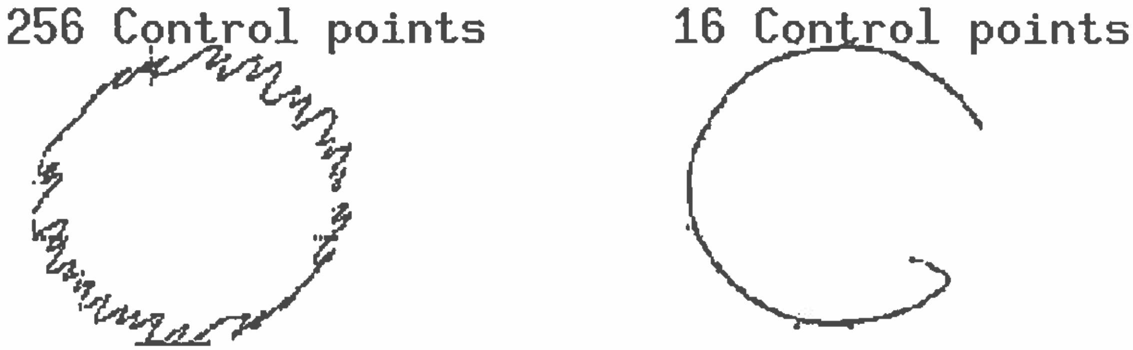


FIGURE 13. Sweep Extraction: The Curve on the Left is a 256 Point Analytic Circle, with Random Perturbations Imposed on it. The Curve on the Right is the Approximation Using 16 Points. The Basic Circular Sweep is Clearly Visible.

The choice of the b-spline basis conforms to the scaling function constraints in that they span nested spaces. In this case the spaces

$$V^0 \subset V^1 \subset V^2 ..... \subset V^j \subset ...... \subset V^n$$

are the representations of the curve between the end points. The only difference between the spaces is the number of knots present.

For a given level V^j the knot sequence $x_0, x_1, \cdots, x_{2^j+2m}$ is defined as

$$(x_0, x_1, \cdots, x_{2^j+2m}) = 1/2^j(0, \cdots, 0, 1, 2, \cdots, 2^j-2, 2^j-1, 2^j, \cdots, 2^j) \qquad \text{(IV.21)}$$

---

[5] the Bernstein polynomials

As the resolution increases, the number of knots present in the representation is increased using the Cox-de Boor knot insertion algorithm.

The matrix Q^j is then selected from the solutions to Equation (IV.20) by placing additional constraints like local support.

For the multiresolution analysis to be complete, the corresponding analysis filters have to be computed. The matrices A^j and B^j are inferred from Equation (II.16).

An interesting feature of this construction is that the structure of the matrices P^j and Q^j are banded, and sparse, since the b-spline basis has local support. This makes it possible to have very fast implementations for the synthesis phase of a multiresolution analysis based on this basis. However the inverses are non-local and have global support, making the analysis phase an undesirable $O(m^2)$ operation[6].

Bi-orthogonal Wavelets

An important motivation for multiresolution analysis is the ability to step through resolutions in interactive time. This necessitates the need for all the filter matrices to be sparse. The obvious solution is to construct all four matrices (namely P^j, Q^j, A^j, B^j) as sparse matrices, and not have to compute one pair as the inverse of the other.

The bi-orthogonal property comes by relaxing

---

[6] A workaround to this has been suggested by Quak and Weyrich [30] which performs the operation in linear time, using the banded and symmetric nature of the synthesis matrices

$$< [\Phi], [\Psi] > \; = \; 0$$

as the orthogonality condition, and choosing duals that satisfy the orthogonality condition. The constraints now become:

$$
\begin{aligned}
< [\Phi], [\Psi'] > \; &= \; 0 \\
< [\Phi'], [\Psi] > \; &= \; 0
\end{aligned}
$$

where Φ' and Ψ' are the respective duals and are defined as some functions that are complementary under the space. Mathematically speaking that is

$$
\begin{aligned}
< [\Phi], [\Phi'] > \; &= \; I \\
< [\Psi], [\Psi'] > \; &= \; I
\end{aligned}
$$

As a consequence of the relaxation, the premise that the scaling functions for the space V^j is the basis for the best approximation of the functions in space V^{j+1} is violated, but the computation of the filters is made much easier. Thus the co-efficients of $[\Phi]$ and $[\Psi']$ are used as the synthesis filters P^j and Q^j and the co-efficients of $[\Phi']$ and $[\Psi]$ are used as the analysis pair A^j and B^j.

Implementation Issues

The algorithm to implement the multiresolution b-spline curves is simple. In the analysis or decomposition phase, the individual co-ordinates of the control points of the original b-spline curve are decomposed by passing them through a b-spline basis wavelet transformer[7]. The intermediate scaling function co-efficients then represent the curve at the intermediate resolutions. Some points that must be considered are:

1. The minimum number of points that the curve can be decomposed to, must be greater than the order of the underlying basis functions.

2. Resolution steps of 2^j can be both visually disturbing, and too gross an approximation for good multiresolution representation. Some methods have been suggested to get a continuous level of detail [11] representation. The method adopted was to ascertain the co-efficients for the curve at the two bounding step points (say j and $j+1$) and then use linear interpolation based on an externally controlled variable μ.

Thus the curve is fractionally defined and is continuously represented as

$$C^{j+\mu} = (1-\mu)\Phi^j(t)c^j + \mu\Phi^{j+1}(t)c^{j+1}$$

---

[7] All the illustrations shown here are constructed using order 3, bi-orthogonal b-spline wavelets.

Curve Sweep Versus Character

Curves are often characterized in terms of sweep and character. Sweep is defined as the underlying shape of the curve and character is the local variations in it.

This is well adapted to the wavelet paradigm. In a multiresolution decomposition, the scaling functions would represent coarser and coarser approximations to the curve and in the limit produce the gross shape of the curve. The co-efficients of the wavelet functions, on the other hand, capture the local higher frequency detail for each level j.

The fall out of this representation is tremendous for curve editing applications. Any b-spline curve can be broken down to a level s^{j-k} with level of detail co-efficients $d^{j-k+1}, d^{j-k+2} \ldots . d^{j}$. The low resolution coefficients s^{j-k} can then be modified and the detail co-efficients added back on to change the sweep, but retain the character of the curve.

Examples

The picture in Figure 13 is the decomposition of the 256 control point cubic b-spline curve, after 4 decompositions. The other picture, Figure 14 is the same curve shown with the intermediate transformation levels and approximated with 128, 64, 32 and 16 b-spline control points respectively.

The original curve is a uniform circle that had its radius values perturbed with a positive random value. The decomposition clearly extracts the basic circular shape, and the level of detail co-efficients would have stored the information about the perturbations.

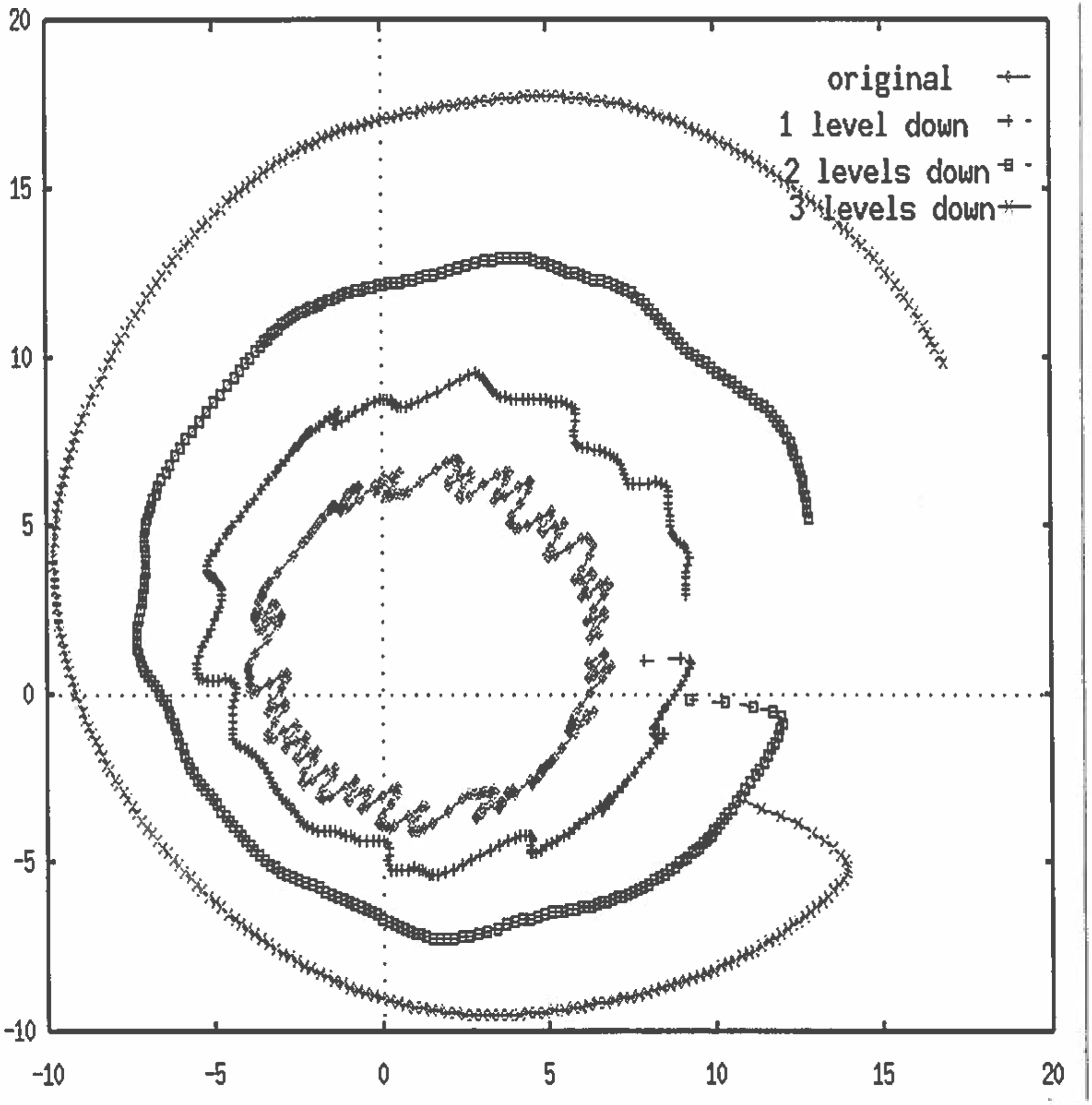


FIGURE 14. Sweep Extraction: The Innermost Curve is the same 256 Point Analytic Circle, with Random Perturbations Imposed on it. The Surrounding Curves are the Unnormalized Approximations, using 128, 64, 32 and 16 Points

CHAPTER V

SUBDIVISION CONNECTIVITY AND REMESHING

Now that we have dispensed with images and curves we arrive at the final aspect of multiresolution representation that we are going to consider in this thesis: application of a wavelet based decomposition to arbitrary surfaces. Recall that wavelets are constructed to have optimal approximations at lower levels of resolution. This prescribes that the construction be adaptive to the spaces that are being represented or the spaces be adapted to enable a wavelet decomposition.[1] An understanding of Voronoi Diagram and Delauney Triangulation is implicitly assumed for the remainder of the text.

The common unstructured polygonal surface representation is not well suited to the wavelet representation. In this chapter we shall look at a pre-process designed by Eck et. al. [7] that takes an unstructured polygonal representation, and converts it to an intermediate form that is more conducive to wavelet decomposition.

The need for the pre-process step may not be immediately apparent but is necessary for the algorithm that will be presented in Chapter VI to work. At this point

---

[1] As we shall see later, in the case of surface wavelets, both the spaces and wavelets have to be modified.

let us just say that this step takes spaces that may not be nested and reconstructs the spaces over which the functions are defined so that they are guaranteed to be nested.

Surface Representations

Surfaces are generally represented as a distribution of vertices that are connected as piece-wise linear polygonal sections. The global connectivity of these polygons defines the topology of the surface. The local connectivity of these polygonal sections are broadly categorized as either structured or unstructured.

A surface polygonization (hereto referred as meshing) is structured if each and every polygon on the surface can be given an index that uniquely identifies the neighboring polygons. An example of such a surface definition would be a piece-wise bilinear surface patch constructed using vertices generated by a de Casteljau type subdivision [33] of a bezier control polygon as shown in Figure 15.

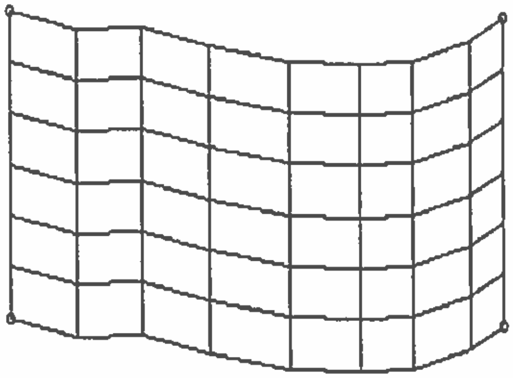


FIGURE 15. An Example of a Structured Surface Mesh

On the other hand, construction of an unstructured surface meshing would begin by picking a subset of vertices and connecting them to form a polygon. This step would be repeated on all the remaining vertices on the surface, until all the vertices have been exhausted. An example of this construction would be the Delauney triangulation of vertices that are distributed along a surface[2] as shown in Figure 16.

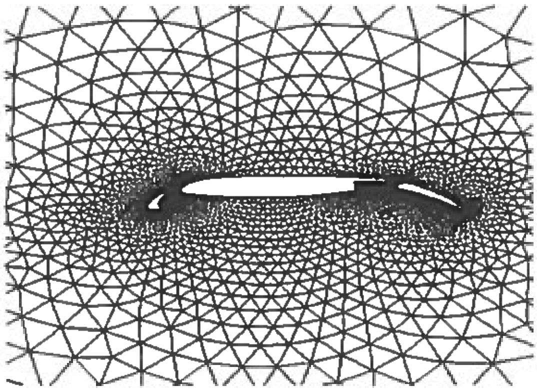


FIGURE 16. An Example of an Unstructured Surface Mesh. (Picture Courtesy Aerospace Engg Dept, Indian Institute of Technology, Bombay, India)

Need for Order

Structured surface meshing lends itself well to a wavelet based representation. The vertices can all be considered to be the control polygon for a b-spline interpolation[3] and the resultant surface classified as a tensor product surface. Then the b-spline based wavelets discussed in Chapter IV can be used in each parametric direction to perform the wavelet decomposition much like the treatment used for decomposition of images.

In the case of unstructured surfaces, however, the spaces are not nested (or

---

[2] Note here that the vertices are not generated with any ordering. In case they are generated in an ordered way, the information is discarded.

[3] For a piece-wise bi-linear surface, this would be an order one b-spline curve.

refinable) by definition. Even if they happen to be nested, there is the practical issue that creating a wavelet decomposition[4] for an unordered mesh in real time is very difficult because the locality or neighborhood is not clearly defined.

Meshing Basics

Let us review some meshing basics and notations that we need to understand before we look at the actual remeshing algorithm. For the remainder of this discussion the notation M, will be used to denote a meshing. The meshing M is defined by a 2-tuple (K, V), where K is called a *simplical complex* or a base complex [34] [19] and V is the set of all vertices on the surface. The simplical complex K in turn is represented by the 3-tuple (vertices, edges, faces) of the polygon based representation of the surface. Thus for a triangulation the 0-simplices are the vertices in the meshing, 1-simplices are the edges and 2-simplices are the polygons.

At this point it is also important to realize the basic difference between topology and geometry. The simplical complex K basically represents the topology (i.e the connectivity information). Topology can be defined in arbitrary dimensions and has no spatial information. Geometry on the other hand is defined in $I R^3$ and has fixed positions in that space. For a simplical complex K made up of m vertices, let the mapping ρ be a linear mapping that translates the vertices specified in K to geometric vertices in $I R^3$. Thus $\rho(|K|)$, where $\rho: \ R^m \rightarrow I R^3$ is the geometric representation

---

[4] Recall that wavelet decomposition is an exercise in local approximation.

of the topology and is called the image of K. If every vertex in K has an unique pre-image then ρ is called an *embedding*. Conversely, if ρ is an embedding, then all points $p \in \rho(K)$, have an unique pre-image b. We can then say that b is the *barycentric co-ordinate* of p. Thus the geometric realization of a surface in IR^3 space from the meshing M is obtained as the image $\rho(K)$.

Subdivision Connectivity

A meshing is considered to be *subdivision connected* when the meshing has been generated by some uniform splitting rule from some base mesh. For a meshing M, the subdivision connected version of M is denoted as M^j, where j is the number of recursive splits that have been applied to the edges in the original meshing.

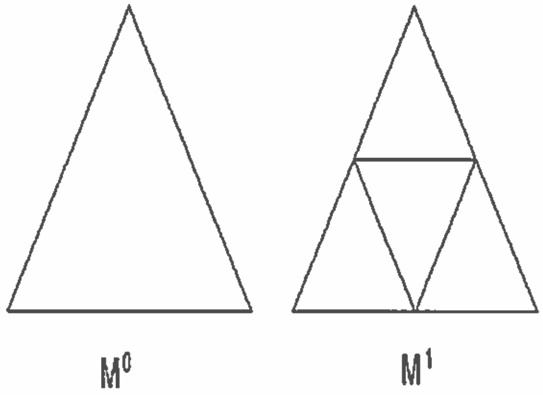


FIGURE 17. One Level of Subdivision, the Splitting Used is a 4:1 Split on Each Edge.

From Figure 17 which shows a subdivision connected version of the original polygon (triangle) for $j = 1$, we can see that if K^0 was the simplical complex associated with M^0 then the simplical complex K^1 associated with M^1 is a nested version of

K^0. Thus we see that subdivision connectivity in a meshing satisfies the requirement of nested spaces for a wavelet decomposition.

The contribution of the pre-process is to take a generic unstructured meshing and create a subdivision connected meshing that approximates the original meshing within an acceptable and specifiable error bound.

The Remeshing Algorithm

The underlying idea here is to take a mesh and generate internal points on that meshing, such that the re-triangulation of the surface will have subdivision connectivity (i.e. perform recursive splits along the edges of the original meshing). Here then is the main challenge that this algorithm overcomes [7]. Generating internal vertices on a plane is easy but generating the internal vertices on a surface in IR^3, such that the new vertices lie on the surface, is extremely hard.

This algorithm achieves that result with following distinct steps:

1. *Partitioning:* First it partitions the entire meshing into small regions. These regions are selected to have the characteristic of small local variations within them. The regions also help in construction of a base mesh that retains the topology of the original meshing.

2. *Parameterization:* The edges within the regions are then projected down to a plane using a harmonic mapping. The inverse of the mapping defines a parameterization.

3. *Re-sampling and Reconstruction:* This step does the actual j recursive splits on all the edges in the planar projection of the regions of the original meshing. The new edges and vertices are then re-projected back onto the surface, by using the inverse from Step(II).

Basic Tools

Harmonic Mapping

Before we look at the actual method let us analyze an important tool that is used several times: harmonic mapping. Let D be a section of the original meshing M, let P be the convex planar projection of D and the mapping $g : Boundary(D) \rightarrow Boundary(P)$ be the mapping for the boundaries. The mapping $h : D \rightarrow P$ is called a harmonic mapping if h agrees with g on the boundary, and minimizes the stretching of internal regions of D [7].

Intuitively, consider the region D of the meshing to be made of rubber sheets corresponding to polygons, sewn together along the edges. The region is going to be laid flat and the boundary stretched so that it agrees with the boundary of the planar region P to form the mapping g. A harmonic mapping would then have mapped the internal polygons and vertices within D to planar locations within the polygon P. Since this sheet would be at rest, the mapping can be thought of as one that minimized the total energy $E_{harmonic}[h]$ of the configuration of rubber sheets.

The harmonic mapping also exhibits the properties of being (i) an embedding

[31] (ii) independent of the meshing of D and (iii) differentiable everywhere. The fact that it is an embedding shall prove useful later, as it implies that the inverse h^{-1} is the parameterization of D over P.

For the actual application, g is approximated by a piece-wise linear mapping that maps the boundary vertices of D onto an n-gon such that the planar vertices lie on a circle, and the sides subtend angles proportional to the arc lengths of the corresponding boundary edges in D. The next part is to compute the mapping h. The solution for h, as noted above, is the solution to the minimization problem for the energy of a system of springs. Therefore

$$E_{harmonic}[h] = 1/2 \sum_{i,j \in edges(D)} k_{i,j} ||h(i) - h(j)||^2 \tag{V.22}$$

where the spring constant is defined as

$$k_{i,j} = (L_{i,k1}^2 + L_{j,k1}^2 - L_{i,j}^2)/Area_{i,j,k1} + (L_{i,k2}^2 + L_{j,k2}^2 - L_{i,j}^2)/Area_{i,j,k2}$$

Using the notation that i, j denotes an edge, $L_{i,j}$ denotes the length of the edge between i and j in D, and $Area_{i,j,k}$ denotes an approximation of area. $k1$ and $k2$ arise as each interior edge corresponds to two polygons.

Partitioning

The reasons for the partitioning is two fold. It locates the regions of small local variation and also defines the base complex K^0. This base meshing serves as the domain for the parameterization in Step(II).

The way the algorithm goes around computing the r regions that constitute the partitions is as follows. The first step is to construct a discrete version of a *Voronoi diagram* on the surface of the mesh M [38] [18] [26]. The diagrams are constructed using a simultaneous version of the advancing front technique [13] where a random, unselected polygon from the original meshing is selected to be the seed for the "advancing fronts". The partition τ_i created with the polygon p_i as the seed point is grown by appending polygons that share two and only two adjacent vertices with a polygon that is currently on the boundary of τ_i. Stated mathematically, the partition is grown by adding polygons as long as the result after the addition is homeomorphic to a disk. If the homeomorphism condition is violated, then the partition growth is terminated and a new partition is grown using an unselected polygon as the seed.

The order in which the polygons are considered for selection is obviously a factor on the outcome. To aid quick computation, the straight line distance between the centers of the polygon is considered to be a measure of the geodesic distance, and is used to order the polygons as they are selected for consideration. The partitions τ_i created this way are the Voronoi tiles for the Voronoi diagram.

Recall that one aim for the partitioning step was to create the base mesh.

Having computed the Voronoi diagram the logical next step is to use the polyhedral dual to the Voronoi diagram (i.e Delauney triangulation), to create the base meshing. The Delauney triangulation will be a dual to the Voronoi diagram, if the diagram by construction:

- is made of partitions that are homeomorphic to disks.
- has no pairs of partitions that share more than one contiguous set of edges.
- has no more than three partitions meeting at a vertex.

If the partitions τ_i are created using these constraints, the Delauney triangulation on the surface of M is the polyhedral dual to the Voronoi tiling.

Triangulation

Before creating the triangulation, each tile τ_i is projected on a planar polygon P_i using the harmonic mapping discussed earlier. Given the harmonic maps a naive approach to creating the Delauney triangulation is simply connecting the centroids[5] of all the P_i's after re-projection into IR^3 using a shortest path algorithm. This can run into severe problems with arbitrary manifold surfaces, as it does not guarantee a triangulation that does not intersect itself. Also finding the shortest path in IR^3 is extremely hard.

---

[5] If the surface has boundaries, then the entire outer domain is to be considered as a one big partition.

The solution is to use the harmonic mapping that was discussed earlier. For each partition τ_i construct the harmonic map h_i that maps it to the planar region P_i. Since the harmonic map is an embedding, the inverse h^{-1} can be used to parameterize τ over P (i.e. be used to generate points on P_i and find the inverse projection into τ_i). Now consider two adjacent partitions τ_i and τ_j with planar projections P_i and P_j that share an edge. Let the shared edge be denoted as $e_{i,j}$ when mapped using the mapping h_i and as $e_{j,i}$ when mapped using the mapping h_j. The line joining the centroids of P_i and P_j is then drawn as two lines. The first joins the h_i^{-1} of the centroid of P_i and the midpoint of $e_{i,j}$ and the second joins the h_i^{-1} of the center of $e_{j,i}$ and the centroid of P_j.

A special case that has to be considered are the partitions that lie on the boundary. These partitions have only one line drawn through them, and that is the one joining the center of the shared edge[6] to the centroid for the planar projection of the boundary partition.

The straight lines joining the vertices generated using the above method is the base mesh for the given meshing. The projections of these lines onto the surface of the original meshing form the surface Delauney triangulation.

---

[6] There is an implicit assumption that there exists a fictitious Voronoi partition outside of M, such that it touches each of the boundaries of M

Parameterization

The Delauney triangulation returns r contiguous regions $T_1, T_2, \ldots\ldots, T_r$ where each region is triangular [7]. The goal of this step is to construct a continuous parameterization $\rho : K^0 \rightarrow M$ (K^0 is the simplical complex for the base mesh) of the initial meshing over the base complex. Harmonic mapping is used again to generate the mapping function h which maps each of those regions T_r to a planar triangular polygon. We can then take the polygon and use an affine mapping[7] to map into the corresponding face F_i of the base triangulation K^0. The final resultant mapping ρ that is produced as a result of the combination of two mappings is an embedding, and therefore the inverse ρ^{-1} defines the parameterization of T_i over F_i. Also by construction ρ agrees on the common boundaries for all T_r thus making ρ^{-1} a globally continuous parameterization.

Re-sampling and Reconstruction

Now that we have a parameterization of each region T_i into planar regions F_i all we need to do is introduce additional vertices on each face F_i in the base mesh and use ρ^{-1} to re-project them back onto M. If these points are generated by recursively splitting each edge in the planar projection polygon P_i, j times, then the resultant meshing M^j after re-projection is called the j subdivision connected version of M.

---

[7] A mapping that is invariant under transformations.

Parametrically uniform sampling: The simplest strategy is to perform the splitting such that the new vertices (hereto referred to as *knots*) are introduced at the midpoints of the existing edges. This is called *parametrically uniform sampling* or *isoparametric sampling.* Though this is easy to implement and has a fast algorithmic implementation, geometric features may be distorted due to geometric under-sampling.

Geometrically uniform sampling: To reduce the sampling artifacts, a formulation for a *geometrically uniform sampling* was proposed by Eck et. al. at the University of Washington. The idea is to be able to generate the triangles on the planar surface, such that when they project back to triangles in IR^3 they are roughly uniform in size. This is an optimization problem that is approximated by the following recursive greedy algorithm.

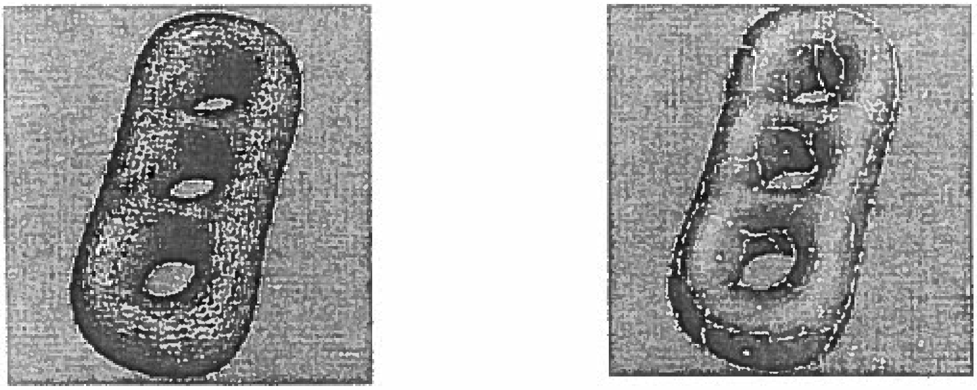
In the parametrically uniform re-sampling process the knot x_i^j at level j is computed as the midpoint of the two neighboring knots $x_{n1(i)}^{j-1}$ and $x_{n2(i)}^{j-1}$ at the level $j-1$. Note that $e_{x_{n1(i)}^{j-1}, x_{n2(i)}^{j-1}}$ defines an edge at level $j-1$. For geometrically uniform sampling instead of doing a uniform subdivision the edges are divided using the parameter λ_i^j such that

$$x_i^j = (1-\lambda_i^j).x_{n1(i)}^{j-1} + \lambda_i^j.x_{n2(i)}^{j-1} \quad where \quad \lambda_i^j \in (0,1). \tag{V.23}$$

The problem then reduces to that of defining the splitting parameter λ_i^j. The ideal, though impractical, solution is to solve a local optimization problem for each triangular region T_i for all the λ_i^j contained in that region. Eck et. al. [7] However, claim to have had reasonable results using a discrete approximation method that they devised.

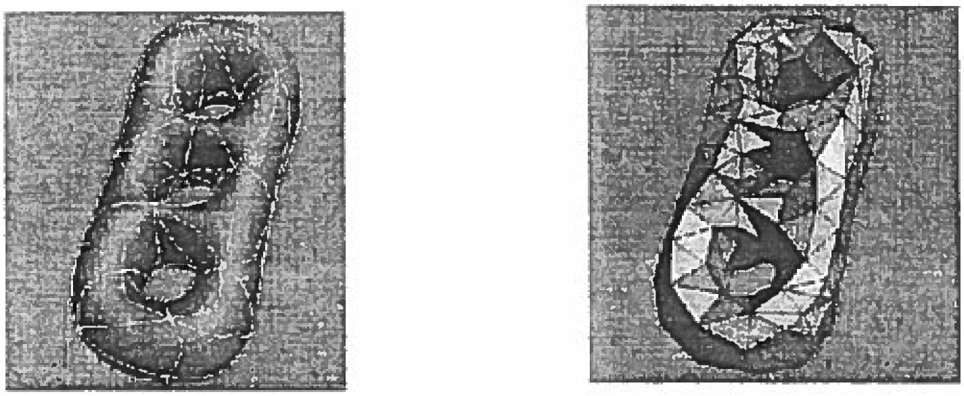
Conclusion

The resultant mesh after the three steps has j subdivision connectivity (i.e j nested spaces). The stage is then set to design a wavelet that will be able to decompose the final triangulation K^j.



The original Meshing M

M with the Voronoi partitions shown. Note the centroids.



The corresponding Delauney Surface Triangulation.

And the Base Mesh

FIGURE 18. The Voronoi Partitioning, and Delauney Triangulation Scheme to Extract the Base Meshing from an Arbitrary Meshing M. (Pictures Shown are Modified from the thesis by Lounsbery).

CHAPTER VI

SURFACE WAVELETS : ARBITRARY MESHES

Now that we have seen the remeshing scheme that creates a subdivision connectivity let us understand why the remeshing was needed. In this chapter we look at a unique vertex encoding scheme that efficiently stores the subdivision connectivity information, and also comes in handy for wavelet decomposition. We then examine the design of a very simple surface wavelet that can be used on a subdivision connected meshing. Lastly, we examine some enhancements that can be made for a better multiresolution decomposition[1].

Vertex Encoding. Traversing and Triangulating

As we have seen in the previous chapter, an arbitrary meshing has to be converted into a subdivision connected meshing to introduce an ordering within the vertices of the triangulation. This ordering is very important to wavelet decomposition as it guarantees the presence of nested spaces. Since the remeshing is done on an unstructured arbitrary mesh, the ordering information has to be stored on each node

---

[1] The notations used for this chapter have been borrowed extensively from Michael Lounsbery's PhD thesis [23].

in such a way that the ordered triangulation can be reconstructed from just the vertex information. In short, we need a vertex encoding scheme that enables recreation of the triangulation and also efficient traversal of the resultant meshing.

Vertex Encoding

All the vertices need to be encoded using the following rules.

The Vertex Encoding Rules

1. All vertices are named using a 6-tuple. The elements of the 6-tuple for a triangulated surface are V_a, V_b, V_c, i, j, k where the values of V_a, V_b and V_c are the vertex indices for the vertices of the triangle on the base mesh that contains the vertex being considered. The values of i, j, k are the the barycentric co-ordinates of the vertex within the triangles that has V_a, V_b and V_c as vertices.
2. The values of $i + j + k = 2^j$ where j = level of subdivision.
3. The i, j, k fields are sorted in increasing order. If any of the indices i, j, k are zero then the corresponding vertex weight, V_a, V_b, and V_c, is ignored.[2]

Figure 19 shows an example face, with the vertex encoding for one the vertices. The example face also shows the vertices generated by two levels of subdivision.

---

[2] In practice, a zero is stored in its place.

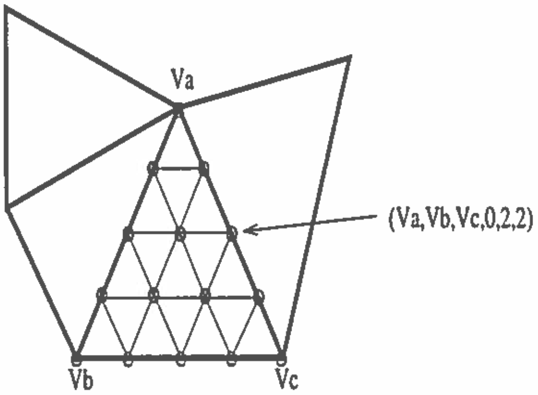


FIGURE 19. The 6-tuple Vertex Encoding for a Point in a M^2 Meshing (the Thick Lines Denote the Base Mesh).

Traversal

Since this encoding scheme is used to construct the ordered triangulation of these vertices, the following primitive functions are defined on it.

Traversal Functions

IsCorner(V) := checks whether V is a corner vertex, i.e. a vertex from the original base mesh. The check is to see if the first two entries of the sorted list of i, j, k are zero .

IsEdge(V) := checks whether V is an edge vertex, i.e. a vertex on an edge in the original base mesh. The check is to see if the first entry of the sorted list of i, j, k is zero .

IsNewVertex(V) := checks whether V is a new vertex, i.e. a vertex that was introduced at this level of subdivision[3] and did not exist at the coarser level. The

---

[3] The usage of the level of subdivision will be clear after the surface wavelet has been designed

check is to see if any of the i, j, k indices are odd.

IsOldVertex(V) := Conversely, the check whether V is a vertex that existed in the coarser version of the triangulation is to see if all the indices are even.

NV = Coarser(V) := returns NV which is the coarser version of V, i.e. if V is a vertex in the triangulation at subdivision level j then NV is the vertex that corresponds to V in the subdivision level $j-1$. The rule to get to NV is to keep the V_a, V_b, V_c as it is, and shift the i, j, k index values to the right by 1. Do note that the vertex encoding at a coarser level exists only for vertices that existed in the finer level. Thus only vertices that return TRUE for $IsOldVertex(V)$ are considered.

NV = Finer(V) := returns NV which is the finer version of V, i.e. if V is a vertex in the triangulation at subdivision level j, then NV is the vertex that corresponds to V in the subdivision level $j+1$. The Rule to get to NV is keep the V_a, V_b, V_c as it is, and shift the i, j, k index values to the left by 1.

NV1,NV2 = Parent(V) := returns the two vertices $NV1$ and $NV2$ which are the parents of the vertex V, i.e. the vertex V that was formed by splitting the edge $NV1, NV2$ that existed in the coarser level. Note that a vertex that has been created due to a subdivision will contain exactly one index out of i, j, k as even and the rest will be odd. The vertex encoding of the parents are constructed by finding the even index and then creating the encodings as

$$TempV1 = V_m, V_n, V_o, m, n+1, o-1 \text{ and}$$

$$TempV2 = V_m, V_n, V_o, m, n-1, o+1$$

where

m = the even index within i, j, k

n = the next index after m

o = the index after n, where "after" is taken in a mod 3 sense.

V_m = the vertex index corresponding to m

V_n = the vertex index corresponding to n

V_o = the vertex index corresponding to o

and then finding

$NV1 = Coarser(TempV1)$ and

$NV2 = Coarser(TempV2)$

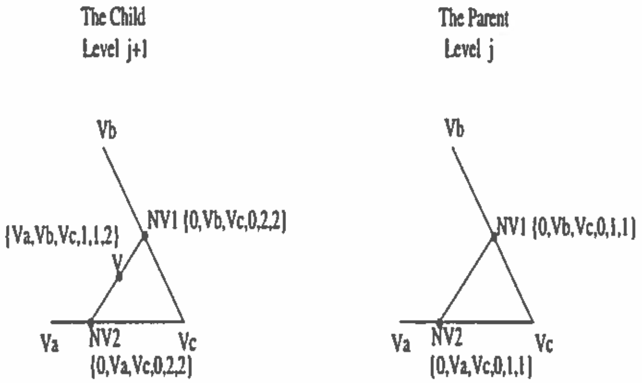


FIGURE 20. An Example of an Edge at Level j and $j+1$. Note the Parent-Child Relationship and the Correlation with the Level of Subdivision

$NV = Child(V1,V2) :=$ where NV is the child vertex of $V1$ and $V2$, or NV is the vertex introduced into the edge $e_{V1,V2}$ that existed in the coarser version of the triangulation. The method to get to NV is to generate $v1$ and $v2$ as two finer level versions of $V1$ and $V2$. Then the 6-tuple of v is rearranged so as to match the base mesh vertices of $v1$ and $v2$. The i, j, k values are computed as the sum and average of $v1$ and $v2$.

$$v1 = Finer(V1) \text{ and}$$

$$v2 = Finer(V2)$$

thus we get

$$NV = (v1 + v2)/2$$

after matching the (V_a, V_b, V_c) values for $v1$ and $v2$.

Figure 20 shows an example of the transition from parent vertices to child vertices and vice versa for a section of a face on the base triangulation.

Triangulation

The first challenge is to recreate the subdivision connected triangulation given the vertices encoded in the manner described and the above mentioned primitive operations. As noted in Appendix B, a typical representation of the subdivision connected meshing has a base mesh representation and then the vertices that lie on the actual surface expressed as the x, y, and z co-ordinates along with the 6-tuple. The triangulation algorithm simply takes each triangle face from the base mesh and recursively fills in the subdivision triangles according to the following algorithm.[4]

```
function Triangulate {
```

---

[4] For efficiency reasons the actual co-ordinate information for the vertices should be stored independent of the 6-tuple name. This is especially useful when generating a vertex tuple using the primitive functions, as the 6-tuple can then be used to lookup the spatial location information.

```
input Vertices V1,V2,V3 ; Depth d;

        if (d == Level of Subdivision)
                makeTriangle(V1,V2,V3);
        else {
             V4 = Child(V1,V3);
             V5 = Child(V1,V2);
             V6 = Child(V2,V3);

        /* we are implicitly assuming that the subdivision
        connected mesh has been generated using  4:1 recursive
        splits */

             Triangulate(V1,V5,V4,d+1);
             Triangulate(V5,V2,V6,d+1);
             Triangulate(V5,V6,V4,d+1);
             Triangulate(V4,V6,V3,d+1);
        }
}
```

Lazy Wavelets: A First Look at Surface Wavelets

For the development of the surface wavelets we shall adopt a slightly different policy than before. We shall consider a specific case of surface decomposition and then show that the decomposition is indeed a wavelet decomposition. The results will then be expanded to a framework for a general purpose multiresolution representation.

The Polyhedral Example

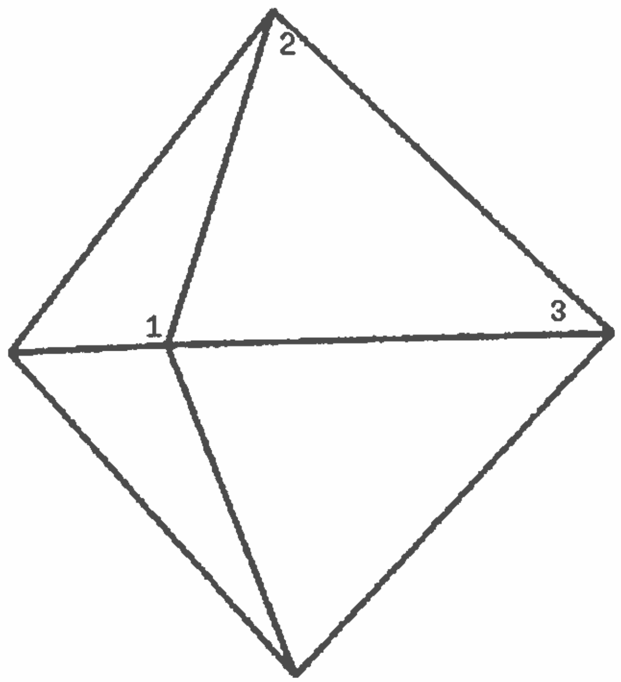


FIGURE 21. The Polyhedral Base Mesh for the Example Surface. The Face with the Vertices Numbered will Constitute Our Example

Let us start by making the assumption that the example that we are considering has a base mesh that is in the shape of a simple polyhedra as shown in Figure 21 and that the surface at level of decomposition j has been created by the subdivision connectivity algorithm of Chapter V. Recall that a multiresolution analysis is completely defined if all the filters that take the decomposition from level j to $j-1$ and back from $j-1$ to j are defined.

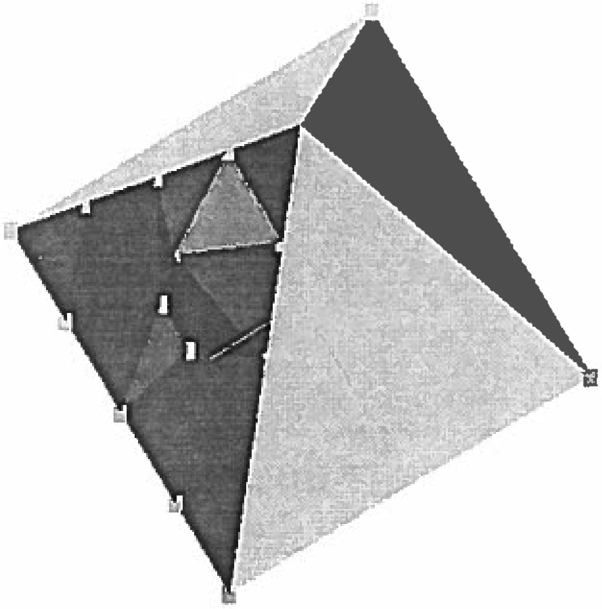


FIGURE 22. The Top Section of the Example Surface with Shading Turned On

Figure 22 and Figure 23 show the top section of the example surface that we are interested in. Figure 24 shows the same section flattened out and with the vertices numbered for a dereferenced lookup of the vertex location in Table 1, which has actual vertex locations corresponding to vertex numbers. In a real application, the vertex number would be replaced by the 6-tuple vertex encoding scheme that was described earlier and a lookup function defined on the 6-tuple would be used to retrieve the vertex locations.

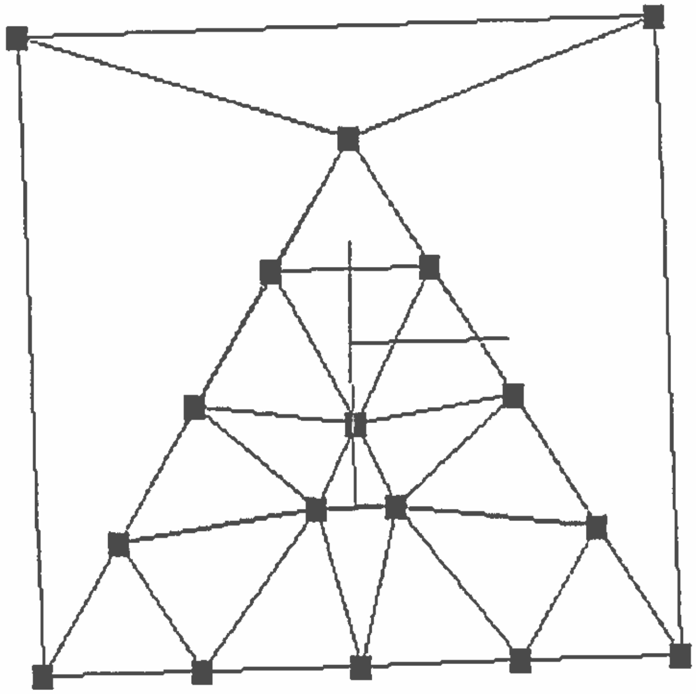


FIGURE 23. The Wire Mesh Version of the Top Section of the Example Surface with the Vertices Marked

The Analysis Step

In Chapter V we saw that triangulation is defined as a simplical complex which uses the vertex locations as the canonical basis vector in R^m, where m is the number of vertices in the simplical complex. Also the pre-process step of creating the subdivision connected meshing from the arbitrary meshing also created the nested spaces for the basis vectors. These nested spaces were generated as the R^m spaces were created by subdivision of the base meshing. From Chapter II we know that the analysis step is basically the construction of two filters, namely A^j and B^j for each level j. Here then consider

$A^j = O^j$ such that $V^j = V^{j+1}O^j$

where

V^j : is a matrix of vertices and represents the connectivity at level j

O^j : is called the older vertex matrix, and contains the same the element "1" where the corresponding vertex existed in V^j, in the level $j-1$ and is zero everywhere else.

```
/* A simple implementation of the Analysis filter A (smoothing filter) */

        For (all vertices in level j) {
                if (IsOldVertex(theCurrentVertex)) {
                        encode theCurrentVertex appropriately
                        include theCurrentVertex in Vj
                }
        }
```

Therefore we see that the subdivision connectivity has defined the A^j filter by construction. The filter B^j also can be derived from the construction of the subdivision connectivity . Consider the filter to be defined as

$B^j = [-N^j \;\; I]$

where

N^j : is the new vertex matrix and is the formulation used for the subdivision

I : is the Identity matrix.

We can see that the two components of the B^j filter are responsible for computing the difference between the actual location of the vertex (the reason for the I component in the matrix) and the location of the vertex as would be computed by the subdivision computation (the function of the N^j part of the matrix). A simple implementation of the B^j filter would be

```
/* A simple implementation of the Analysis filter B (detail filter) */

        /* note this part is common with the A filter */
        For (all vertices in level j) {
                if (IsOldVertex(theCurrentVertex)) {
                        encode theCurrentVertex appropriately
                        include theCurrentVertex in VLISTj
                }
        }
        For (all vertices in VLISTj) {
                select(v1,v2) from VLISTj
                v3 = Child(v1,v2)
                temp = Lookup(v3) /* the lookup function uses the
                encoding to find the actual co-ordinates of the vertex */
                Lookup(v3) = temp -( Lookup(v1) + Lookup(v2))/2
                encode v3 appropriately
        }
```

The Reconstruction Step

To complete the wavelet decomposition, we need to have the reconstruction filters that correspond to the analysis filters that we have just constructed. We shall now switch back to familiar matrix notation for the smoothing and detail filters and compute the reconstruction filters as per Equation (II.16). For simplicity let us assume that the vertices are presented as a table with increasing index number, ordered according to the level of subdivision they were introduced in (as shown in the Table 1). Then the analysis filters that operate on the points can be broken into two sections each.

$$A^j = [\ \textit{old vertices at level } j \mid \textit{new vertices at level } j\ ]$$

$$B^j = [\ \textit{old vertices at level } j \mid \textit{new vertices at level } j\ ]$$

Re-writing the filter we get

$$A^j = [\ I \ \ O\ ] \quad \text{and} \quad B^j = [\ -N^j \ \ I\ ]$$

where

N^j : is the new vertex matrix and is the formulation used for the subdivision

I : is the Identity matrix

O : is the Zero or Null matrix.

Using the inverse matrix rule, we get the values of P^j and Q^j to be defined as

$$P^j = \begin{bmatrix} I \\ N^j \end{bmatrix} \ and \ Q^j = \begin{bmatrix} O \\ I \end{bmatrix}$$

For the example surface, we see that one level of decomposition using the analysis filters result in the surface shown in Figure 26 and Figure 27. The part of the simplical complex corresponding to the face that contained the bump is shown in Figure 25 and the actual vertex information after one pass is shown in Table 2.

Disc Wavelets : The Sophisticated Surface Wavelet

Now that we have seen the surface decomposition and reconstruction algorithms the next step is to verify that it satisfies all the constraints of the wavelet transform as discussed in Chapter II. The lazy wavelet satisfies the requirement of nested spaces, but violates the orthogonality property. The scaling functions (namely the simplical complexes) are not orthogonal to the detail functions and results in a less than good

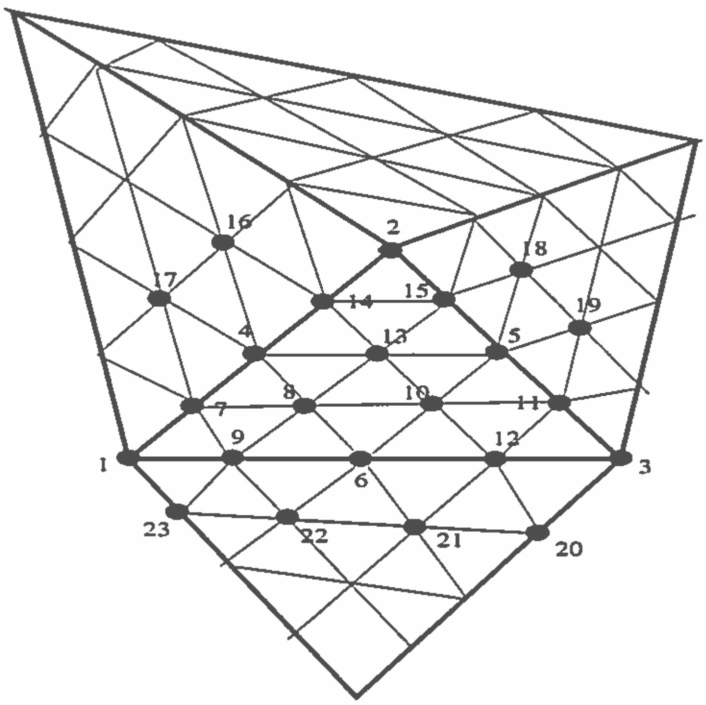


FIGURE 24. The Top of the Polyhedra at Level j Flattened Out. The Bold Lines are the Edges of the Base Mesh.

approximation for local surface characteristics at coarser levels. This is obvious from the example surface, as the small localized mound at level j completely vanishes in the representation at level $j-1$.

The ideal solution to this problem is to construct wavelet functions that are fully orthogonal to the scaling functions. In real time graphics applications this becomes an immediate problem, as the filters will have global support and the resultant matrices will become dense. To enable the usage of sparse matrix inversion routines [15] for generating the reconstruction filters, the orthogonality condition is relaxed to bi-orthogonal filters. Recall from Chapter IV that the bi-orthogonal filters are constructed so that the scaling functions and the wavelet functions are mutually orthog-

The Scaling function co-efficients C_i^j

Vertex #	x	y	z
1	10.0	0.0	-10.0
2	0.0	10.0	0.0
3	10.0	0.0	10.0
4	5.0	5.0	-5.0
5	5.0	5.0	5.0
6	10.0	0.0	0.0
7	7.5	2.5	-7.5
8	7.0	3.0	-1.25
9	10.0	0.0	-5.0
10	7.0	3.0	1.25
11	7.5	2.5	7.5
12	0.0	0.0	5.0
13	7.0	7.0	0.0
14	2.5	7.5	-2.5
15	2.5	7.5	2.5
16	5.0	5.0	-5.0
17	-3.3	3.3	6.6
18	5.0	5.0	5.0
19	3.3	3.3	6.6
20	7.5	-2.5	7.5
21	6.6	-3.3	3.3
22	6.6	-3.3	-3.3
23	7.5	-2.5	-7.5
24	10.0	0.0	-10.0

TABLE 1. The Vertices of the Example Meshing

onal to the duals and not the functions themselves. Thus using the bi-orthogonality constraint

$$< \phi_i^j, \psi_i^{j'} >= 0$$

we can get a local mask that can be used as the matrix component N^j instead of a simple averaging function.

One solution suggested by Lounsbery et. al. is to rewrite the wavelet functions

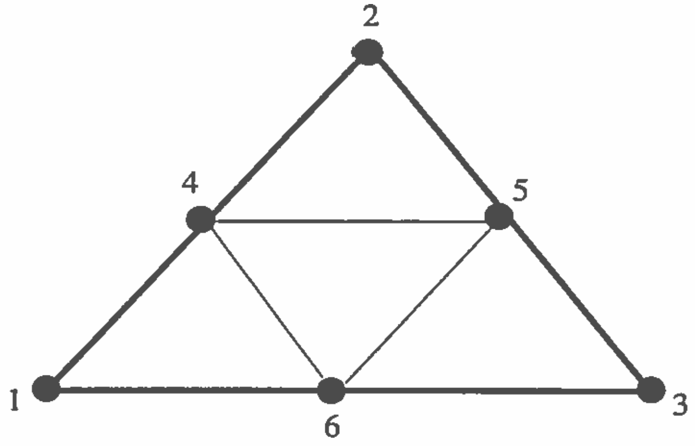


FIGURE 25. The Central Face of the Base Meshing at Level $j-1$

as

$$\psi_i^{j-1'} = \phi^j_{m,i}(x) - \sum_k Surf^j_{k,i}.\phi_k^{j-1}(x) \tag{VI.24}$$

where the notation $\phi_{m,i}$ is used to denote the i-th scaling function co-efficient at that level and ϕ_k denotes a subset $k \subset m$, of all the scaling function co-efficients defined at that level, in the neighborhood of i. The values of $Surf^j_{k,i}$ forms a mask centered at i [16] and the elements of the mask are obtained by solving the system of equations

$$< \psi_i^{j-1} | \phi_l^{j-1} >= 0 \tag{VI.25}$$

for all l such that the support of ψ_i^{j-1} overlaps with ϕ_l^{j-1}.

Vertex #	x	y	z
The Scaling function co-efficients C_i^{j-1}			
1	10.0	0.0	-10.0
2	0.0	10.0	0.0
3	10.0	0.0	10.0
4	5.0	5.0	-5.0
5	5.0	5.0	5.0
6	10.0	0.0	0.0
The Wavelet function co-efficients D_i^{j-1}			
7	0.0	0.0	0.0
8	-0.5	0.5	1.25
9	0.0	0.0	0.0
10	-0.5	0.5	1.25
11	0.0	0.0	0.0
12	0.0	0.0	0.0
13	2.0	2.0	0.0
14	0.0	0.0	0.0
15	0.0	0.0	0.0
16	0.0	0.0	0.0

TABLE 2. The Vertices for the Face Under Consideration at Level $j-1$

The disc like shape of the mask is used to characterize the filter that is being used. Derose et. al. define the term k-disc to mean the mask that consists of all the vertices that can be reached at a distance of k edges from the vertex under consideration. As the size of the disc grows, the region of support of the wavelet functions grow and they become more orthogonal to the scaling functions, and the approximation by the coarser scaling functions becomes better. The flip side is that in implementation it means having to deal with denser matrix inversions which can be extremely hard for large matrices.

In terms of an implementation, the change that would have to be made to the lazy wavelet implementation would be to take the scaling function co-efficients at

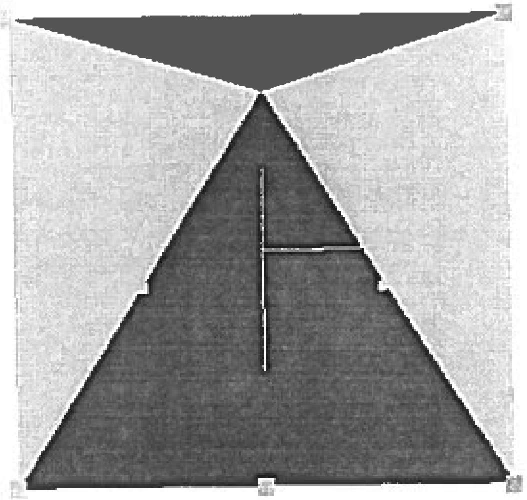


FIGURE 26. The Example Surface at Level $j-1$

each stage and modify it to be

$S_i^j = S_i^j + \sum_k Surf_{k,i}^j . D_{k,i}^j$ in the Analysis Step

and

$S_i^j = S_i^j - \sum_k Surf_{k,i}^j . D_{k,i}^j$ in the Reconstruction Step

where

$Surf_{k,i}$: is the k-disc centered at i

S_i^j : are the co-efficients of ϕ_i^j and

$D_{k,i}^j$: are the co-efficients of ψ_i^j in the k-neighborhood of i.

The case of the lazy wavelets can be thought of as a special case of the k-disc wavelets with $k = 1$ and $Surf_1 = 1$.

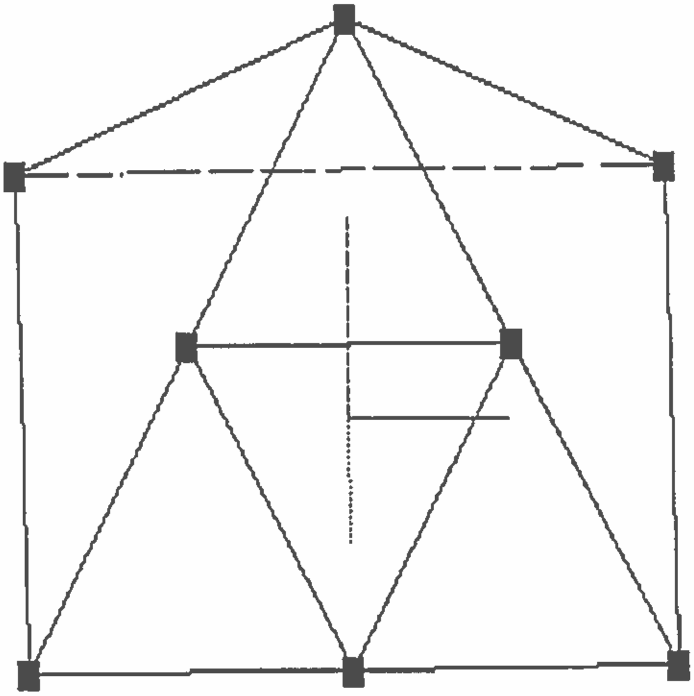


FIGURE 27. The Wire Mesh Version of the Surface at Level $j-1$ with the Vertex Locations Clearly Pointed Out

The framework for multiresolution surface representation thus provides a solution to the issue of being able to represent an arbitrary triangulated surface using an application defined resolution for surface features. The shape of the surface can be evolved from some basic, easy to render shape, to the required complexity. In our example we have explicitly assumed that the base meshing was a polyhedra, and that is the simplest representation for our surface. This result can be extended to other basic shapes with similar results, as long as the triangulated, highest definition surface is a subdivision connected version of the base mesh.

We also see now the purpose of the pre-processor described in Chapter V was to create a subdivision connected meshing and at the same extract a base mesh that was topologically the same as the original surface.

Surface Compression and Applications

Geometric compression, as can be achieved by this technique, has many instant applications. One of the most important applications has been in modeling for radiosity based rendering engines. The ability to use the gross shape or the fine shape from a singular representation is an enormous win for the form factor calculations. Another conceivable application is a vision model based raytracer that uses surface models at the required resolution to improve upon efficiency. A last and as yet emerging application, is the use of multiresolution modeling for internet based applications. The modeling system can be hooked to the network performance, and performance bottlenecks can be avoided by using a progressive display, where the resolution improves as data arrives. Conversely for interactive applications, the resolution for the entire scene can be hooked to the bandwidth available.

A problem that crops up with these applications, is the determination of a metric that can used to decide the resolution level that is required. One solution suggested by the authors of this thesis, is to use the simplest level first. Then we pick a triangle that is roughly parallel to the screen and project it. Following that compute the centroid of the projection, and measure the average length of the vectors from the centroid to the three vertices. The desired resolution is the one that makes the average length to be approximately one pixel long. Due to the subdivision connectivity property this can be easily computed.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

The greatest lure to the scientific and engineering community of using a multiresolution paradigm stems from the ability to step, from microscopic to macroscopic resolutions, within the data being examined. There are two distinct aspects of graphics that benefit most from multiresolution representation.

The first successful scenario, is for display of objects of large complexity which do not need to be drawn in frame time. The object can be displayed in the coarsest form possible, and can then be progressively refined. The main advantages of such an approach is that the user gets constant visual feedback and also the refinement can be allowed to proceed for just the length of time available for that frame. This ensures optimal utilization of the data bandwidth available for the application.

The second way to gain from a multiresolution representation is to use some metric to decide, before rendering complex scene definitions, the required complexity for the objects comprising the scene. The objects comprising the scene can then be constructed to the needed degree of resolution. The decision process used for selecting the desired resolution is the key factor, and can be based on a variety of weighting factors ranging from the sensitivity of the Human Visual System to a simple distance metric. This methodology would ensure an optimal utilization of the computing resources that are available.

Another emerging area for the use of multiresolution representation is in the areas of data visualization and simulation studies. The ability to have multiple resolution representations that are local in the temporal domain and in the spatial domain, allows for simulations that generate copious data to be trimmed, thus allowing a finer grained study along the areas needing higher resolution, and a coarser representation along the regions of lesser interest. Finkelstein et. al. [10] created an application based on this idea, where they allowed real time motion of large vector flows, and varied the complexity of the data being visualized depending upon the importance of the region in the overall scheme.

Yet another application of separable wavelet transformation was suggested by [22] who designed a novel technique to significantly speed up volume visualization applications. The basic idea was to decompose the entire volume using a separable 3-dimensional wavelet transform, and then use the reconstruction as the model for the intensity integral along a ray. Thus the problem of drawing the properly oriented voxels was reduced to one of reconstruction of the wavelet decomposition.

As we have seen wavelets provide a fast and elegant way to solve most of the computer graphics complexity problems, and can be easily implemented in hardware too. However, there are some problems that need to be ironed out. The main focus of most wavelet based modeling research is directed towards creation of a more robust framework than the one we saw for creating multiresolution representations for surfaces. If a general purpose framework existed, then the graphics community should be seeing a tremendous increase in rendering performance, as scenes can then be optimally rendered. In fact more complex modeling applications can then go one step ahead and model the lighting scenario using a multiresolution paradigm akin to the one suggested by Bolin and Meyer [3].

In conclusion, the multiresolution representation made possible with a wavelet based decomposition is poised to make as big an impression on the scientific community as the Fourier transform has done.

APPENDIX A

A SIMPLE WAVELET TRANSFORMER

A Simple wavelet transformer: This routine will take as input

1. The wavelet function co-efficients for fully orthogonal wavelet (e.g. Haar, Daubechies etc) and
2. The input sequence

The generated output will contain a sequence of equal length composed of the half length scaling function co-efficients and the half length wavelet function co-efficients. The matrix is constructed as per Numerical Recipes in C [29].

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>

// this program is to implement the general format of the wavelet
// transform for the single dimension, using the general purpose
// wavelet transform matrix ..

// the aim is to write a function that will read in a data file, and
// load up the values into the transform matrix, it will then read the
// input sequence and generate the transformed sequence.
// program written 1/22/1997
// Author Shrijeet Mukherjee
// MS thesis, university of oregon.



// Matrix multiplier, this is a generic routine now, but may need to
// be sped up for future use, as this is the most compute intensive
// job here ..


void MatMultiplier(double **KernelMatrix,
double DataBlock[], double ResultMatrix[], int size ) {

  int row,col,var;

  row = 1;
  for( col = 1 ; col <= size ; col++ ) {
    ResultMatrix[col] = 0.0;
    for( var = 1 ; var <= size ; var++ )
      ResultMatrix[col]  =  DataBlock[var] * KernelMatrix[var][col]
+ ResultMatrix[col];
  }
}

// The Main Program ..

main() {

  int NumCoeffs, MatSize;
  double **DWTMatrix, *IDWTMatrix, *Coeffs,
    *DataBlock, *OutBlock, *ResultMatrix;
  int k,l,StartCol,TmpInt;
  float TmpFloat;

  // get number of Co-efficients ..
  scanf("%d",&NumCoeffs);

  if ((Coeffs = (double *) calloc(NumCoeffs, (sizeof(double)))) == NULL) {
    printf("\n\a Memory allocation for the Coefficients failed ..\n");
    exit(0);
  }

  // load them into the array ..
  for(k=0;k<NumCoeffs;k++) {
    if ((scanf("%f",&TmpFloat)) == EOF) {
      printf("\n\a Mismatch in number of Coefficients, Operation failed ..\n");
      exit(0);
    }
    Coeffs[k] = (double) TmpFloat;
  }
  // get number of elements in the block ..
  scanf("%d",&MatSize);

  // fail if block size is not a power of 2 (for the time being I have
  // it set as a multiple of 2 .. will have to change it later ..

  if ((MatSize % 2) != 0) {
    printf("\n\a Error in input, the input data block size has to
be a power of 2 ..\n");
    exit(0);
  }
  else {
    // allocate pointers for each row ..
    if((DWTMatrix = (double **) malloc
((size_t)((MatSize) * sizeof(double *))))== NULL) {
      printf("\n\a Memory allocation for the Coefficients failed ..\n");
      exit(0);
    }
    // fill up pointer to row 0 ..(and also the whole space, as memory
    // is actually linearly allocated ..
    if((DWTMatrix[0] = (double *) malloc
((size_t)(MatSize*MatSize*sizeof(double))))==NULL) {
      printf("\n\a Memory allocation for the Coefficients failed ..\n");
      exit(0);
    }
    // fill up pointers to the rest as offset ..
    for (k=1;k<=MatSize;k++)
      DWTMatrix[k] = DWTMatrix[k-1] + MatSize;

    // and now set the matrix to zero ..
    for (k=0;k<MatSize;k++)
      for (l=0;l<MatSize;l++)
        DWTMatrix[k][l] = 0.0;
  }

  // now we have the co-efficients, let's make the DWT matrix ..
```

```
printf("\n ... please wait building the DWT matrix ..\n");

// note that every even row starts at even offset from the left edge..
// this is used to calculate the offset column for each row..
StartCol  = -2;

for(k=0;k<MatSize;k++) {

  // the matrix needs to have the elements setup so that odd rows
  // have the co-effs just laid out ..
  if ((k%2) != 1) {

    // set the offset for the next two rows ..
    StartCol = StartCol + 2;

    for(l=0;l<NumCoeffs;l++) {
      DWTMatrix[k][(StartCol + l) % MatSize ] = Coeffs[l];
    }
  }
  // and the even rows have it laid out in reverse, with alternate
  // elements negated.
  else {
    TmpInt = -1 ;
    for(l=0;l<NumCoeffs;l++) {
      DWTMatrix[k][(StartCol + l) % MatSize ] =
                      Coeffs[NumCoeffs - l - 1] * -1 * TmpInt;
      TmpInt = TmpInt * -1;
    }
  }
}

// allocate space for the input block of data ..
if ((DataBlock = (double *) calloc(MatSize, (sizeof(double)))) == NULL) {
  printf("\n\a Memory allocation for the Coefficients failed ..\n");
  exit(0);
}


// load them into the array ..
for(k=0;k<MatSize;k++) {
  if ((scanf("%f",&TmpFloat)) == EOF) {
    printf("\n\a Mismatch in number of Data Elements, Operation failed ..\n");
    exit(0);
  }
```

```
    DataBlock[k] = (double) TmpFloat;
  }

  // allocate space for the temp output block of data ..
  if ((ResultMatrix = (double *) calloc
(MatSize, (sizeof(double)))) == NULL) {
    printf("\n\a Memory allocation for the Temp Space failed ..\n");
    exit(0);
  }


  // allocate space for the output block of data ..
  if ((OutBlock = (double *) calloc(MatSize, (sizeof(double)))) == NULL) {
    printf("\n\a Memory allocation for the output Block failed ..\n");
    exit(0);
  }

  MatMultiplier(DWTMatrix, DataBlock, ResultMatrix, MatSize);

  // swap the LPF, and HPF parts , as the output has them alternating ..
  // note: I am reusing StartCol, but it is not a Column here ..

  StartCol = MatSize/2; // this is the index into the OutBlock, where
                        // the HPF sequence will begin

  for (k=0;k<StartCol;k++) {
    OutBlock[k] = ResultMatrix[2*k];
    OutBlock[k+StartCol] = ResultMatrix[2*k + 1];
  }

  for (k=0;k<MatSize;k++) {
    printf("\n %7.2f",OutBlock[k]);
  }

}
```

APPENDIX B

A SAMPLE SUBDIVISION CONNECTED MESHING

The data file will have the following components:- (the lines with the ''*'' the actual data, the other lines are explanation)

* The base mesh
* The subdivision depth (integer)
* The dimension of the mesh data (integer) [ex: height over sphere = 1]
* The relative weight of each field in the mesh data, used when creating a weight of a wavelet coefficient. Weights of 0 may be used, but only at the end if the weight sequence. The effect is that the corresponding data are copied to the output exactly as they were read in, and their value contributes nothing to the weight of the corresponding wavelet.

For example: the following is header input for a base octahedron mesh subdivided 9 times, and carrying 3 pieces of data (e.g, height, latitude, longitude on a sphere). Only the first datum has a non-zero weight, of 1.0.

```
Vertex 1  0 0 0

Vertex 2  0 0 0

Vertex 3  0 0 0

Vertex 4  0 0 0

Vertex 5  0 0 0

Vertex 6  0 0 0

Face 1  1 2 3

Face 2  1 3 4

Face 3  1 4 5

Face 4  1 5 2

Face 5  3 2 6

Face 6  4 3 6

Face 7  5 4 6

Face 8  2 5 6

endbase

9

3

1 0 0
```

* The subdivision input: a series of lines in the format:

vi vj vk i j k x { y z ... }

where

vi,vj,vk are integers giving the vertex index of the face

i,j,k give the multi-index of the point within a subdivided face

x { y z ... } are floats giving the data at the point

NB: if any of i,j,k are 0, the corresponding vertex index

vi,vj,vk is ignored and can be any integer

For example:

2 5 8 1 1 6 0.4 0.67

sets point 1,1,6 of face 2,5,8 to have info 0.4, 0.67

2 5 8 0 2 6 0.3 0.7

sets point 0,2,6 of face x,5,8 (on an edge) to have info 0.3, 0.7

If a point is repeated (such as an edge point shared by 2 faces), the last value is the one that will be stored. It is not necessary to repeat these points, even though they may appear to have different indices.

BIBLIOGRAPHY

[1] Editor Alan Fournier. Siggraph 95 course notes: Wavelets and their applications in computer graphics, 1995.

[2] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. In *Communications on Pure and Applied Mathematics*, pages 44:141–183, 1991.

[3] Mark R. Bolin and Gary W. Meyer. A frequency based ray tracer. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 409–418. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[4] Andrew Certain, Jovan Popović, Tony DeRose, Tom Duchamp, David Salesin, and Werner Stuetzle. Interactive multiresolution surface viewing. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 91–98. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[5] C. K. Chui. *An Introduction to Wavelets.* Academic Press, Inc., Boston, 1992.

[6] Ingrid Daubechies. *Ten lectures on wavelets / Ingrid Daubechies.* Philadelphia, Pa. : Society for Industrial and Applied Mathematics, 1992, 1992. Contains lectures delivered at the CBMS conference organized June 1990 by the Mathematics Dept. at the University of Mass, Includes bibliographical references (p. 341-351) and indexes.

[7] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[8] Tim Edwards. Discrete wavelet transforms: Theory and implementation. Technical report, Stanford University, June 1991. tim@sinh.stanford.edu.

[9] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Boston, 1992.

[10] Adam Finkelstein, Charles E. Jacob, and David H. Salesin. Multiresolution video. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 281–290. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[11] Adam Finkelstein and David H. Salesin. Multiresolution curves. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 261–268. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[12] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, Reading, Massachusetts, 1990. ISBN-0-201-12110-7.

[13] L. Formaggia. An unstructured mesh generation algorithm for three-dimensional aeronautical configurations. In A. S. Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Simulation and Related Fields*, volume Proceedings of 3rd International Conference, page 249, June 1991.

[14] D. Gabor. Theory of communication. In *Journal of the IEEE*, pages 429–457. IEEE, 1946.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Second Edition.* Johns Hopkins Press, Baltimore, Maryland, 1989. ISBN-0-8018-3772-3.

[16] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Second Edition*, pages 519–520. Johns Hopkins Press, Baltimore, Maryland, 1989. ISBN-0-8018-3772-3.

[17] Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 221–230, 1993.

[18] D. G. Holmes and D. D. Snyder. The generation of unstructured triangular meshes using delauney triangulation. In *Proceedings on the Second Conference on Grid Generation in Computational Fluid Dynamics*, Swansea, 1988. Pineridge Press.

[19] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, August 1993.

[20] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[21] Erwin Kreyzig. *Advanced Engineering Mathematics, Fifth Edition.* John Wiley and Sons, Inc., New York City, New York, 1983. ISBN-81-224-0016-7.

[22] L. Lippert and M. H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. Technical report, Institute for Information Systems, Swiss Federal Institute of Technology, Zrich, January 1995. TR-228.

[23] John Michael Lounsbery. *Multiresolution analysis for surfaces of arbitrary topological type.* PhD thesis, University of Washington, Seattle, WA, 1994. Currently in Alias Research.

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

[25] Y. Meyer. Ondelettes sur líntervalle. In *Rev . Mat. Iberoamericana*, pages 7:115–133, 1992.

[26] D. Mount. On finding shortest paths on convex polyhedra. Technical Report 1495, Department of Computer Science, University of Maryland, 1985.

[27] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall of India, New Delhi 110 001, 1992. ISBN-0-87692-720-7.

[28] Sumanta N. Pattanaik and Kadi Bouatouch. Haar wavelet: A solution to global illumination with general surface properties. In *Fifth Eurographics Workshop on Rendering*, pages 273–286, Darmstadt, Germany, June 1994.

[29] William H. Press, Saul A. Teukolsky, William T. Vellering, and Brian P. Flannery. *Numerical Recipes in C, The art of scientific computing*, pages 592–595. Cambridge University Press, Cambridge, United Kingdom, 1992. ISBN-81-85618-16-X.

[30] E. Quak and N. Weyrich. Decomposition and reconstruction algorithms for spline wavelets on the bounded interval. Technical report, Center for Approximation Theory, Texas A&M University, april 1993. CAT Report 294.

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

[32] Peter Schroeder, Steven Gortler, Michael Cohen, and Pat Hanrahan. Wavelet projections for radiosity. In Michael F. Cohen, Claude Puech, and Francois Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 105–114. Eurographics, June 1993. held in Paris, France, 14–16 June 1993.

[33] Hans-Peter Seidel. Algorithms for B-patches. In *Proceedings of Graphics Interface '91*, pages 8–15, June 1991.

[34] Edwin Henry Spanier. *Algebraic topology / Edwin H. Spanier.* New York : Springer-Verlag, 1st corr. springer ed edition, 1966.

[35] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufman Pulishers, Inc, San Fransisco, California, 1996. ISBN-1-55860-375-1.

[36] G. Strang. Wavelets and dilation equations: A brief introduction. In *SIAM Review*, pages vol 31, 4 (Dec 89), 614–627. SIAM, 1989.

[37] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[38] D. F. Watson. Computing the n-dimensional delauney tessalation with application to voronoi polytopes. In *The Computer Journal*, pages 24(2):167–171, 1981.

[39] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics Proceedings (SIGGRAPH 96)*, 1997.