WAVELET METHODS FOR COMPUTER GRAPHICS

Steven J. Gortler

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE

January 1995

To all great music makers

Abstract

This thesis discusses how a wavelet basis can be used in the context of two computer graphics applications, realistic rendering and geometric modeling, to produce more efficient and flexible algorithms.

The goal of realistic rendering is to simulate the interreflection of light in some geometric environment to produce realistic images. Radiosity is a commonly used solution method for this problem. Recently Hanrahan et al. have introduced a hierarchical method that can solve radiosity problems in $O(n)$ time instead of $O(n^2)$. This thesis explores how the hierarchical radiosity algorithm can be formally understood from the context of wavelet theory. When the radiosity problem is expressed with respect to a wavelet basis, the resulting linear system is sparse, with only $O(n)$ significant terms. By casting the hierarchical method in this framework, a variety of wavelet basis functions can be used, resulting in more efficient radiosity methods.

This thesis also discusses how wavelets can be used in the context of geometric modeling. Geometric modeling is the study of how geometric shapes can be represented and manipulated by a designer. This thesis explores the use of wavelets to represent parametric curves and surfaces within the context of geometric modeling interfaces.

One intuitive modeling interface commonly used in geometric modeling allows the user to directly manipulate curves and surfaces. This manipulation defines some set of constraints that the curve or surface must satisfy (such as interpolation and tangent constraints). Direct manipulation, however, usually leads to an underconstrained problem since there are, in general, many possible surfaces meeting some set of constraints. Therefore an optimization problem must be solved.

This thesis discusses how geometric modeling optimization problems can be solved more efficiently by using a wavelet basis. Because the wavelet basis is hierarchical, iterative optimization methods converge rapidly. And because the wavelet coefficients indicate the degree of detail in the solution, they can be used to determine the number of basis functions needed to express the variational minimum, thus avoiding unnecessary computation. An implementation is discussed and experimental results are reported.

Acknowledgments

None of the work discussed in this thesis would have been possible without the most-excellent guidance of my advisor, Professor Michael Cohen. He has given me insight, guidance, and inspiration every step of the way.

I also would like to thank the readers of this thesis, Professors Pat Hanrahan and Elisha Sacks for their important input to this thesis. The radiosity research described in this thesis was done in collaboration with Professor Hanrahan, and his comments at the fourth floor library meetings helped us focus on the important research issues. Professor Sacks' guidance during my early years at Princeton are sincerely appreciated.

The radiosity research was done in close collaboration with Peter Schröder. It was a delightful experience working with him. James Shaw implemented the user interface for the variational modeling tool. His effort was a great help. I also enjoyed yapping with him and his wife Colleen.

Chuck Rose and Dan Boneh rounded out the yapping crew, and made the department almost a fun place to be. Dan Boneh was also very patient with all my math questions. I would like to thank David Ohsie who always had good advice.

My various office mates throughout the years have enriched my Princeton experience, but certainly none more than Ayellet Tal. It looks like our study meetings for general exams have finally paid off.

I would like to thank my parents for their loving support.

The research reported in this thesis was made possible in part through support provided by the NSF (Grant No. CCR92-96146) and DARPA (Grant No. DABT63-92-C-0053.P00002).

Contents

List of Figures

Chapter 1

Introduction

One use of computer graphics is to create and display images. These computer generated images have many uses, for example to allow people to view and interact with virtual objects and environments that do not actually exist. The exploration of these virtual environments may be useful for people who are designing objects and environments such as mechanical engineers and architects. Computer generated images may also be used for artistic expression or entertainment purposes such as for computer art, video games and special effects in motion pictures. This thesis will focus on two important subproblems of the process of viewing and interacting with virtual objects and environments: *geometric modeling* and *realistic rendering*.

In geometric modeling, a person uses a computer to create the geometric description of objects and environments. A geometric modeling tool must allow the user to easily actualize his abstract notion of some object, and so it must provide him with the ability to interact with the virtual object in an intuitive fashion. It is also important for the modeling tool to respond to the users actions at interactive speeds.

In realistic rendering the computer is used to generate images of virtual objects and environments. The images are called realistic in that they appear similar to how the object would appear in the real world, as governed by the physics of light (or at least to a good approximation of physics). To achieve this result one needs a method of simulating the interreflection of light. If the simulation is accurate, then the resulting image will have realistic looking lighting properties.

In both of these domains, geometric modeling and realistic rendering, there is a need to represent functions. The representation must be flexible, expressive, and lead to efficient algorithms. In geometric modeling the objects which the user designs can be described as a combination of curves and surfaces. which in turn need to be represented in some functional form. In realistic rendering, the lighting simulation results in a light intensity distribution over all of the surfaces of the geometric environment. This light distribution is expressed as an *illumination* function over the domain of the environment's surfaces. This illumination function also requires a representation.

In each of these domains there are a variety of function representations that have been used. In geometric modeling both *implicit* and *explicit* function representations have been used. One example of an implicit representation defines a surface as being the zero set of some 3D function $F(x, y, z)$. This thesis will focus on explicit representations, where the function is represented directly. In particular, in a *parametric representation*, a surface is represented by three functions $(X(s,t), Y(s,t), Z(s,t))$. These three functions define a geometric point (X, Y, Z) for each parametric point (s, t).

In realistic image synthesis, there have also been a variety of representations used for the illumination function. One method used for realistic rendering is *ray tracing* [77]. In ray tracing a brightness value is computed for each pixel that makes up the image. In this method, the illumination function is represented *pointwise*, i.e., the function is only defined at a discrete set of points (which correspond to the surface points "seen" through each pixel). This thesis will focus on the *radiosity* method for realistic rendering [38]. Unlike ray tracing, which generates a single image with a single view of the environment, the radiosity method is view independent; it produces a representation of the illumination over all of the surface points in the environment. One advantage to such a method is that once computed, a radiosity solution can quickly generate realistic images from any point of view in the environment. This can be used for interactive walk-throughs of architectural models. Thus, radiosity requires the representation of a complete function, not just a set of discrete points.

Representing an arbitrary function is a difficult problem. At each point, an arbitrary function can take on any value, and thus there are an infinite variety of functions.

A useful representation requires some form of discretization, where a function's representation is reduced to a finite set of scalar values. This finite set of scalars can then be stored in a computer, and manipulated with algorithms. One approach for such a discretization is *point sampling* where the function is represented at a discrete set of points. In the limit, as one represents more and more points, point sampling can converge to any well-behaved function. Unfortunately the cost of approaching this mathematical limit is prohibitive. By using fewer points, the cost can be kept down, but the representational value of these few points becomes limited.

What is required is a way to generate a good approximation of an entire function, with a finite set of discrete values. For the representation to be useful, it must allow for the function to be evaluated at arbitrary points rapidly. For example, suppose one wishes to approximate a univariate function. One possible representation is to linearly interpolate between point samples of the function. The discrete points define a set of linear polynomials, and these polynomial can be quickly evaluated at any arbitrary point. A drawback of piecewise linear interpolation is the derivative discontinuities at the point samples; these discontinuities may not be desirable for some applications. A different representation method which generates a smooth function is polynomial interpolation (using Lagrange's method [27]). Although these polynomials are smooth, Lagrange interpolation is not a well behaved method; when one interpolates a large set of points, the resulting polynomial tends to oscillate wildly [27].

Both linear interpolation and polynomial interpolation can be expressed by linearly combining a set of functional "building blocks" called *basis functions*. This method allows one to approximately represent a function using a set of discrete scalar values called "coefficients". There are a wide variety of basis functions that can be used, and each gives rise to a representation with particular advantages and disadvantages (such as the tradeoff between piecewise linear interpolation, and Lagrange interpolation).

As this thesis will discuss, there are many different kinds of basis functions that can, and have been used in computer graphics. Many of the basis functions commonly used in computer graphics can be categorized as "piecewise", that is, each basis function represents some local piece of the function. In contrast to the piecewise methods, this thesis will discuss a fairly new family of basis functions called *wavelets* that can be used to represent some of the functions that arise in computer graphics. A wavelet basis represents the function with a multi-level hierarchy. In a wavelet basis, some of the basis functions represent the the broad/coarse aspects of the function, while other basis functions describe local detail.

As one might expect, the type of representation (basis) used has a strong impact on the computational requirements for computing and operating with these functions. In particular, the wavelet basis has been found to have the following powerful attributes:

- If one has represented some function in an appropriate "piecewise" basis, the wavelet representation can be computed in $O(n)$ time, where n is the number of functions in the basis [54].

- The hierarchical nature of a wavelet basis allows many "smooth" functions and *operators* to be compressed, and represented with a *sparse* set of coefficients. It has been shown in [8] that a large class of smooth operators can be represented in a wavelet basis by a sparse matrix with only $O(n)$ significant entries. This is a dramatic improvement over piecewise bases, where these operators are expressed by a dense matrix with $O(n^2)$ significant entries. The operator defining light transport, used in realistic rendering, is such a smooth operator. Thus, this feature of wavelets can be exploited to obtain an efficient method for solving realistic image synthesis problems.

- The hierarchical nature of the wavelet basis allows a large class of differential equations and optimization problems to be expressed with a well conditioned matrix. When a piecewise basis is used, the condition number κ of the resulting matrix grows as $O(n^2)$ [11]. The number of iterations required by iterative methods for convergence is a function of κ. As discussed in [25, 81], for a large class of differential equations, when the problem is expressed in a wavelet basis, the condition number of the resulting matrix remains constant, independent of n. This feature of wavelets can be used to obtain efficient solutions to certain optimization problems that arise in geometric modeling.

1.1 Realistic Image Synthesis

In realistic image synthesis the goal is to produce a realistic image of an environment that is described to the computer. This realism is useful in architectural simulation, as well as special effects production. The description of the environment includes the geometric description of the objects (e.g., walls, furniture, plants), as well as a physical description of the reflective properties of the objects (e.g., the left wall reflects blue light, the mirror is shiny) and finally the positions and properties of light emitting objects (e.g., a fluorescent light, a desk lamp, or the sun shining through a window). In a realistic image, it is important to predict how the lighting would appear if this described environment were real: which areas would be bright, which areas would be less bright, which areas would be in shadow, and which areas would have colors that resemble nearby reflecting objects due to interreflection. The computed lighting may be represented by an illumination function. For every point (and perhaps every direction) in the environment, the illumination function represents the brightness (in each wavelength of color in the visible spectrum).

Computing the correct illumination function has been one of the primary foci of computer graphics research. There are two families of methods commonly used, the ray tracing method [77], and the radiosity method [38]. The ray tracing method simulates the light in the environment by tracing the paths of many photons. In contrast to this, the radiosity method breaks up the environment into little pieces or elements, and computes how the illumination at one element effects the illumination of the others. The result of these interactions is computed by solving a linear system of equations. At the heart of this linear system is a matrix which represents the interactions of these elements. If there are n elements used to discretize the environment, the matrix has n^2 entries (corresponding to all ordered pairs of elements). In a realistic sized problem, n can be large (on the order of millions), and so the cost of computing the matrix entries and solving the linear system can be prohibitive.

This thesis will describe how the use of a wavelet basis to represent the illumination function gives rise to a *sparse* linear system, where only $O(n)$ of the matrix terms are non-zero. Intuitively, this occurs because in many regions of the environment, the energy interaction can be well represented with a very coarse description. With the hierarchical wavelet representation, this coarse description can be accounted for, while ignoring the unnecessary detail. The sparseness of the system allows the solution to be computed very efficiently.

1.2 Geometric Modeling

In computer aided geometric modeling the goal is for a user to create geometric descriptions of objects. These objects may include curves or surfaces (the canonical example being a automotive engineer designing a car fender). These curves or surfaces are functions that must be represented. This thesis will discuss a variety of modeling paradigms and explain how a wavelet basis can be used to obtain efficient and useful modeling tools.

In the simplest geometric modeling paradigm the user sculpts a curve or surface by dragging *control points* (e.g., Bézier or B-spline [27]). Each control point directs how the different basis functions are to be blended, and so this method is sensitive to the type of basis functions being used. In particular, when a piecewise basis, such as a B-spline or Bézier basis is used, the control points have a local and narrow influence on the curve or surface. While this may be desirable in some instances, at other times a user may want to be affecting the curve or surface at a broader resolution, changing the overall sweep of the curve or surface. This thesis will discuss how a wavelet basis can be used to address this problem. The hierarchical nature of this basis allows the user to manipulate the curve or surface at any desired resolution.

A more intuitive modeling paradigm allows the user to directly manipulate curves and surfaces. The user is allowed to click on the curve or surface with a mouse, and drag that point, and use other widgets to manipulate the tangent at some point. This manipulation defines some set of constraints that the curve or surface must satisfy (such as interpolation and tangent constraints). Direct manipulation, however, usually leads to an underconstrained problem since there are, in general, many possible surfaces meeting some set of constraints. Finding the “best” solution requires solving a variational problem.

One of the most popular methods for solving this variational modeling problem is the finite element method[72]. In this method the surface is described by a large collection of basis functions. Each of these functions, or elements, describes the behavior of the surface in some very small region. As with the radiosity method, expressing the problem with respect to some basis gives rise to a (linear) system of equations. This system is then solved using an descent method, that proceeds from some initial guess by iteratively changing the quantity of each of these elements until the best combination is found. Unlike radiosity, the matrices arising from optimal surface problems are usually sparse, with only $O(n)$ terms of the matrix being non-zero. Unfortunately the resulting systems of equations are poorly conditioned and the iterative descent methods converge slowly when a finite element basis such as B-splines is used [72, 71]. This occurs because the solution method is always focusing on a very narrow region of the solution. This thesis discusses how geometric variational modeling problems can be solved more efficiently by using a wavelet basis. Because the wavelet basis is hierarchical, the iterative methods are able to alter the solution at a variety of resolutions. This results in a well conditioned systems of equations, and the iterative methods converge rapidly.

There are other important advantages to using a wavelet basis for geometric modeling. When using a finite element method, it is important to choose the right number of basis functions needed to represent the optimal solution. If too few are chosen, then there may not be enough degrees of freedom to meet the constraints, or to meet them with enough "optimality". If too many basis functions are chosen, then the program will do unnecessary work. This decision can be difficult to make a priori, and so an adaptive method is needed, that can extend or reduce the basis as the iterations proceed. As this thesis will discuss, this adaptivity can be achieved effectively with a wavelet basis.

The organization of this thesis is as follows. Chapter 2 presents an introduction to wavelets. Chapter 3 reviews the radiosity formulation. Chapter 4 discusses how wavelets can be used in the context of radiosity to obtain an efficient solution method. Chapter 5 reviews some basic concepts from geometric modeling. Chapter 6 discusses how wavelets can be used to perform multiresolution modeling. Chapter 7 discusses how wavelets can be used to quickly solve variational modeling problems. The results of the thesis are summarized in chapter 8.

Chapter 2

Wavelets

This chapter will introduce the general concepts used to construct a wavelet basis. Some of the details are technical, and the reader may want to first read this chapter lightly, and later come back to this chapter as a reference. More details about wavelets may be found in [59, 16, 54, 26].

2.1 Basis Functions

Computer graphics applications, in particular geometric modeling and realistic rendering, often require a method for representing or approximating some function $F(t)$. One possible representation is through a linear combination of a set of *basis functions* using scalar coefficients as weights.

$$F(t) = \sum_i f_i \phi_i(t) \tag{1}$$

In this thesis, by convention functions will be denoted by uppercase letters while the corresponding scalar coefficients will be denoted by lowercase letters. The parameter t will sometimes be left off to avoid clutter. In this case, F is a function with parameter t, and the f_i are scalar coefficients.

The use of basis functions is a very general concept, and includes many well known methods for representing functions. For example, a function can be represented at a discrete set of points (without loss of generality let the discrete set of points be have

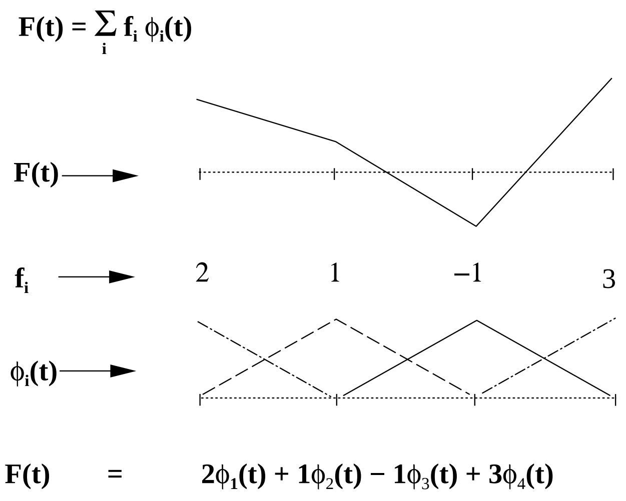


Figure 1: The hat basis functions are linearly combined to construct piecewise linear functions.

the parameter values t_i for i between 0 and n). In this case, the basis functions are *Delta functions*

$$\delta(t - t_i) \tag{2}$$

A function can be represented by linearly interpolating between pairs of discrete points, in this case the basis functions are *hat functions* (Figure 1). A function can be represented by polynomial interplation using *Lagrange* polynomial basis functions.

$$\frac{\prod_{j=0}^{n}(t - t_j)}{\prod_{\substack{j \neq i \\ j=0}}^{n}(t_i - t_j)} \tag{3}$$

A function can be expanded using its taylor series about the origin using *monomial* basis functions

$$t^i \tag{4}$$

A function can be expressed as a combination of different frequencies using the complex *Fourier* basis.

$$e^{-\sqrt{-1}it} \tag{5}$$

Although the use of a linear combination of basis functions is a very general notion, it is mathematically very simple, and therefore useful in domains, such as rendering and modeling, where one needs to solve for the desired solution. More complicated representations, such as the rational B-spline representation that allows the division of one basis function with another, do not lend themselves as easily to such solution methods.

In order to represent an arbitrary function, one needs a basis made up of an infinite number of basis functions. For example if one wants to represent all functions [1] on the interval $[0 \ldots 1]$, one can do this by using a complete Fourier basis (The complex basis functions $e^{-\sqrt{-1}it}$ may be thought of as sines and cosines of all frequencies i). In this case, Equation (1) becomes an infinite sum. Because this representation uses an infinite number of basis functions, the space of functions spanned by this basis is said to have *infinite dimension*.

In order to use the representation in an algorithm, one cannot use an infinite dimensional basis. Instead one must use a basis made of a finite collection of basis functions. This limits the space of representable functions to a *finite dimensional* space. For example, one can obtain a finite dimensional Fourier basis by dropping all basis functions above a certain frequency. The finite dimensional function space is less expressive, but the representation is more manageable. Because the finite dimensional function space is less expressive, arbitrary functions cannot be exactly represented in it, they must be approximated instead. For example, if one wishes to represent some function that has a high-frequency component, using some finite dimensional Fourier basis, one must approximate it by dropping the high-frequency information. This thesis will be primarily discussing finite dimensional bases.

The Fourier basis functions have *global* support. The basis functions, which are sines and cosines, are non-zero over the entire parameter domain. When basis functions have global support, the corresponding coefficients have global influence. This can be a disadvantage. It is difficult to modify local detail using a basis with global support. For example, in geometric modeling, one may be interested in altering some local region of the curve or surface. This is not possible with a global basis, without altering all of the coefficients.

To obtain a more flexible and local representation, "piecewise" bases are often used. In such a basis, each basis function represents a local region of the function,

[1]functions with finite L^2 norm

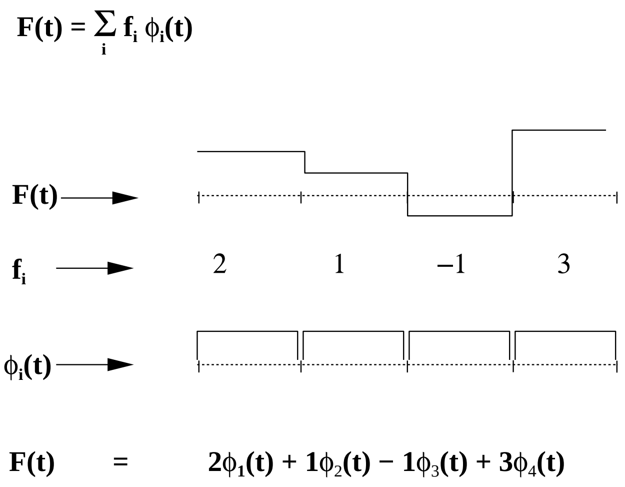


Figure 2: The box basis functions are linearly combined to construct piecewise constant functions.

and so the corresponding coefficients have local influence. Because these basis function are only non-zero over a finite region, they are said to have *compact support.*

Perhaps the simplest imaginable piecewise basis is the "box" basis (Figure 2). With the box basis, the space of functions that can be represented is all functions that are piecewise constant between adjacent integers. If one wants to represent the space of all functions that are C^1 (have continuous derivative) and are piecewise linear between adjacent integers, one can use the "hat" basis (see Figure 1). One can continue this sequence, representing functions which are piecewise higher order polynomials, using B-spline basis functions N_i^d of arbitrary degree d [6]. The B-spline basis functions can be defined recursively using B-spline basis functions of lower degree.

$$N_i^d(t) = \frac{t - t_{i-1}}{t_{i+d-1} - t_{i-1}} N_i^{d-1}(t) + \frac{t_{i+d} - t}{t_{i+d} - t_i} N_{i+1}^{d-1}(t) \qquad (6)$$

The sequence of parameter values t_i is called the *knot sequence.* $N_i^0(t)$, the zero order B-spline, is a box function that is zero everywhere except between the knots t_i and t_{i+1}, where it is one. If the knot values are evenly spaced, the resulting basis functions are called *uniform* B-spline basis functions. This thesis will make extensive use of the uniform cubic B-spline basis ($d = 3$). The uniformly spaced knots will be at the integers $t_i = i$. The resulting basis functions, which will simply be denoted as $\phi_i(t)$,

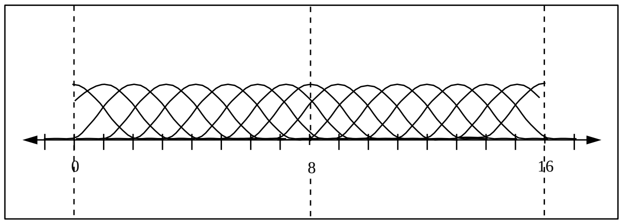


Figure 3: The uniform cubic B-spline basis.

are cubic polynomials between adjacent integers. Each of the basis functions is a translated copy of $\phi_0(t)$, and can be expressed as $\phi_i(t) = \phi_0(t-i)$. The ith uniform cubic B-spline basis function is non-zero between the parameter values i and $i+4$ (see Figure 3). It is made up of four polynomial segments. If one translates each of these segments to the interval between zero and one, they can be expressed as

$$\begin{aligned} \frac{1}{6}(t^3) \\ \frac{1}{6}(-3t^3+3t^2+3t+1) \\ \frac{1}{6}(3t^3-6t^2+4) \\ \frac{1}{6}((1-t)^3) \end{aligned} \tag{7}$$

In geometric modeling, non-uniform rational B-splines or NURBS are often used. These functions are defined as the division of two sets of basis functions, using a set of weighting coefficients w_j.

$$F(t) = \frac{\sum_i f_i N_i^d(t)}{\sum_j w_j N_j^d(t)} \tag{8}$$

This thesis will not be discussing NURBS. For more detail about B-splines see [6, 27].

2.2 Haar basis

Wavelets offer a family of bases that combine the frequency decomposition of the Fourier basis, with the piecewise locality of bases like the box, or hat basis. This combination of features is very useful. Let us start by looking at the simplest and oldest wavelet basis, the Haar basis [42].

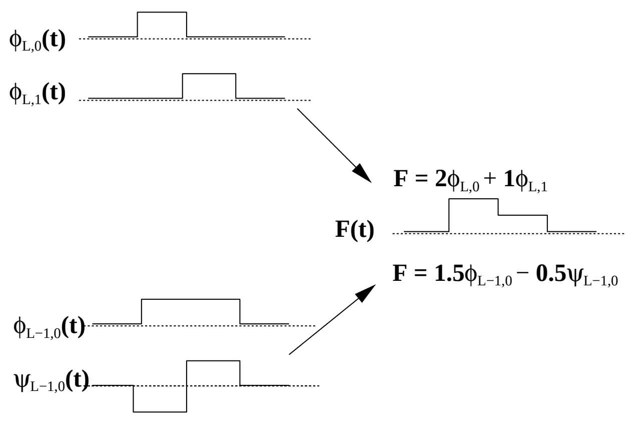

$\phi_{L,0}(t)$
$\phi_{L,1}(t)$
$F = 2\phi_{L,0} + 1\phi_{L,1}$
$F(t)$
$F = 1.5\phi_{L-1,0} - 0.5\psi_{L-1,0}$
$\phi_{L-1,0}(t)$
$\psi_{L-1,0}(t)$

Figure 4: Two adjacent box functions can be replaced by one wider box function and one Haar function to represent the same function F.

To begin the description of the Haar construction, consider two adjacent box functions that will be denoted as $\phi_{L,0}$, and $\phi_{L,1}$, (the meaning of L, the *level*, will become apparent later on in this chapter). It is easy to see that these two basis functions can be replaced by one box function that is twice as wide $\phi_{L-1,0}$, and one step function $\psi_{L-1,0}$ called the Haar function (Figure 4). The new basis functions (at level $L-1$) can be expressed as linear combinations of the old ones (at level L)

$$\begin{aligned} \phi_{L-1,0} &= \phi_{L,0} + \phi_{L,1} \\ \psi_{L-1,0} &= \phi_{L,0} - \phi_{L,1} \end{aligned} \tag{9}$$

and the old (level L) as linear combinations of the new ones (level $L-1$)

$$\begin{aligned} \phi_{L,0} &= \frac{1}{2}\phi_{L-1,0} - \frac{1}{2}\psi_{L-1,0} \\ \phi_{L,1} &= \frac{1}{2}\phi_{L-1,0} + \frac{1}{2}\psi_{L-1,0} \end{aligned} \tag{10}$$

Using these new basis functions, the wider box function represents the pairwise average of the function over the span of the two thinner box functions, and the Haar function represents the pairwise difference of the two halfs from the average. Computationally, obtaining the scalar coefficients for these new basis functions is as easy as taking pairwise averages and pairwise differences of the coefficients of the original box basis.

$$\begin{aligned} f_{\phi_{L-1,0}} &= \frac{1}{2} f_{\phi_{L,0}} + \frac{1}{2} f_{\phi_{L,1}} \\ f_{\psi_{L-1,0}} &= \frac{1}{2} f_{\phi_{L,0}} - \frac{1}{2} f_{\phi_{L,1}} \end{aligned} \tag{11}$$

(In this notation, the symbolic form of the basis function is used as the index of the corresponding scalar coefficient). The inverse coefficient transformation is simply

$$\begin{aligned} f_{\phi_{L,0}} &= f_{\phi_{L-1,0}} - f_{\psi_{L-1,0}} \\ f_{\phi_{L,1}} &= f_{\phi_{L-1,0}} + f_{\psi_{L-1,0}} \end{aligned} \tag{12}$$

This simple replacement of two basis functions, can be done pairwise over the entire range of a box basis with more than two basis functions. Consider a box basis, made up of eight adjacent box functions (the shaded box in Figure 5a). By linearly combining these basis functions with scalar coefficients, all piecewise constant functions over the interval $[0 \ldots 8]$ can be expressed. Call this space of functions V_L. Pairwise averaging and differencing can be used to replace the eight box functions with four wider box functions and four Haar functions (Figure 5a). This new basis is an alternative basis for the same function space V_L, and will be called the *two-part* basis.

Notice though that in this new basis, half of the functions are themselves box function (only twice as wide). This allows for a reapplication of the pairwise averaging/differencing transformation to replace those four box functions with two wider box functions, and two wider Haar functions (Figure 5b). The transformation may again be applied one final time (Figure 5c). The final result is a new basis of eight functions spanning the same function space, called the Haar basis, Figure 5d.

In this Haar basis representation, all functions in V_L are expressed hierarchically. Going from top to bottom, one of the basis functions represents the overall average of the function, and one represents the difference between the left and right half. On the next level the basis functions represent the differences in the quarters, and finally the basis functions on the lowest level represent the differences on the eighths. This

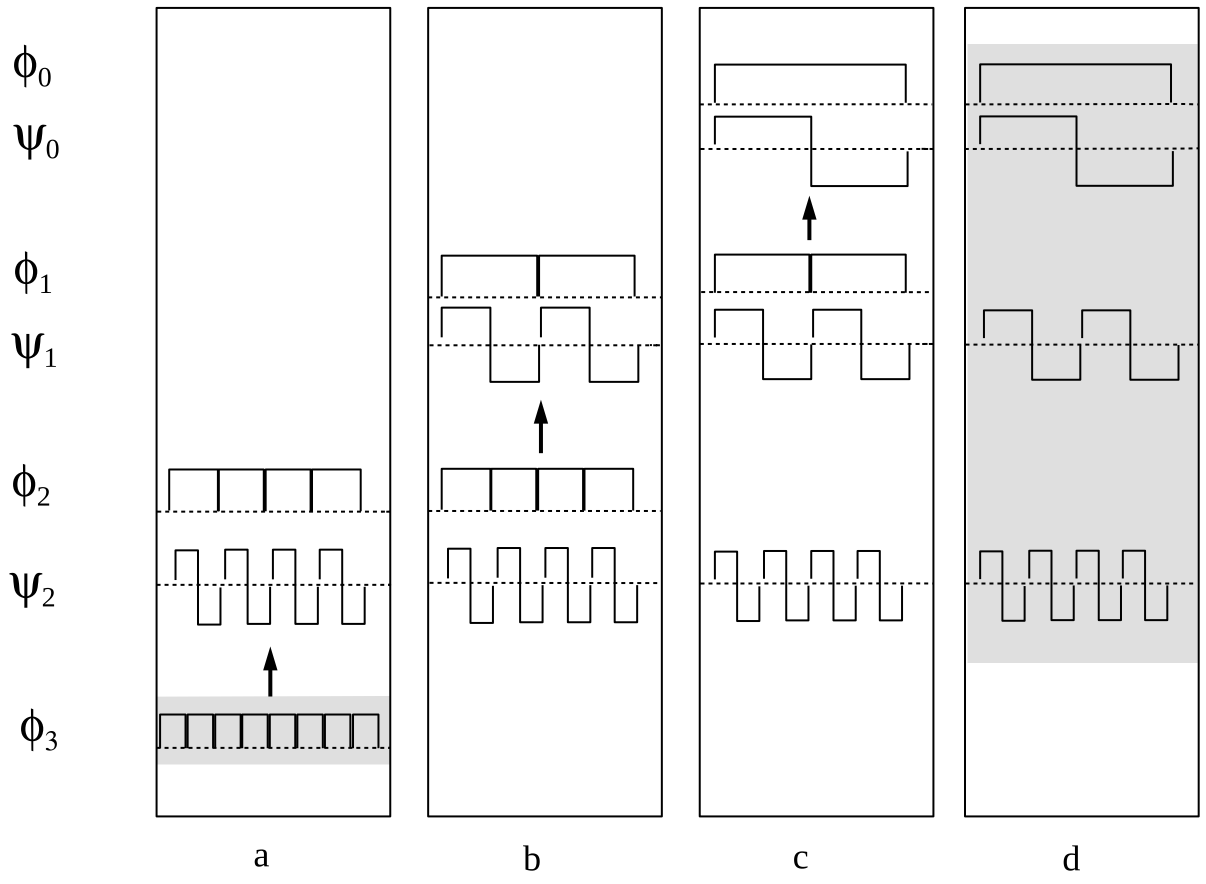


Figure 5: From left to right: a pyramid transformation replaces the box basis with the Haar basis.

construction is very similar to an image pyramid that one might use for texture mapping [78]. In such a pyramid the image is represented at different levels of resolution by successive averaging steps. The Haar pyramid only stores the overall average and all the *differences* between successive levels of the pyramid.

- This new basis combines the locality of the box basis with the resolution/scale decomposition of the Fourier basis. This *multiresolution* property is useful in geometric modeling and allows either the user, or some optimization process to manipulate the solution at a variety of resolutions.

- This hierarchical basis also allows for very economical representations of many functions. If a function has little or no detail in some region (i.e., it is nearly constant over some region), then the Haar coefficient for that region will be near or close to zero. If that basis function is removed from the basis, little is lost in the representation. This *compression* property will be important for both of the applications discussed in this thesis, radiosity and geometric modeling.

- Another feature of this basis is that the transformation of coefficients between the box basis and the Haar basis is very efficient. This level by level transformation described above is called a *pyramid* transformation. Each level of the pyramid transformation can be done in linear time. As each level performs a transformation of half the size, the running time of the complete pyramid is thus governed by the geometric series $(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1) = 2n = O(n)$, and runs in linear time.

The Haar basis is just the simplest example of a wavelet basis. There are an infinite variety of wavelets, where the *scaling function*, ϕ, is some function besides a box function and the wavelet function, ψ, is some function besides a step function. But the concepts are similar; a pyramid-like transformation is applied to the finest resolution ϕ functions to create a hierarchical basis, that has the multiresolution and compression properties.

There are a variety of other properties that classify different wavelet families.

- One property is *smoothness* which can be measured in many ways including the number of continuous derivatives. It is desirable to have smooth basis function if one wants to represent smooth functions.

- Another desirable property some wavelets have is *symmetry*. With a symmetric wavelet, fewer artifacts tend to appear in the representation.

- The *support* of a wavelet refers to the size of the interval over which the wavelet function is non-zero. It is desirable to have wavelets with small, finite support, so that the expense of expressing some local detail is confined to some small region.

- There are two other desirable properties *orthogonality* and *vanishing moments* which will be described in due course in the text.

2.3 B-spline Wavelets

To introduce the concepts and terminology of wavelets in general, this section will describe the construction of wavelet bases based on cubic B-splines. These wavelets are C^2 and symmetric, with finite support.

2.3.1 Two-Part Basis

As with the Haar basis, a first step of the cubic B-spline wavelet basis construction is the creation of a new basis for the same space of curves defined by the B-splines, but consisting of two distinct types of basis functions.

To understand the two-part basis, begin with the uniform cubic B-spline basis made up of translated copies of the B-spline basis function $\phi_0(t)$ (Figure 3). This basis is made up of translated copies of a single hump shaped function $\phi_0(t)$ that will be denoted without the index simply as $\phi(t)$. ϕ is made up of four cubic segments defined in Equation (7).

The set of basis functions can then be defined as

$$\phi_{L,j}(t) = \phi(t - j) \tag{13}$$

(the index j represents the translation of a specific basis from the canonical B-spline left justified at zero, and L is the *level* or resolution of the basis). The space (or family) of curves spanned by all linear combinations of these basis functions is denoted V_L (e.g., V_L contains all functions that are piecewise cubic, with simple knots at the integers).

Wavelets offer an L level hierarchical basis for the space V_L, but let us begin by building a two-part basis at level $L - 1$ of the hierarchy. The two-part basis begins with the basis functions

$$\phi_{L-1,j}(t) = \phi(2^{-1}t - j) \tag{14}$$

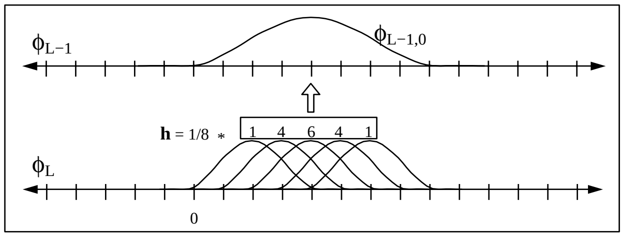


Figure 6: Five B-splines $\phi_{L,j}$ may be combined using the weights h to construct the double width B-spline $\phi_{L-1,0}$

These basis functions are twice as wide as the original B-spline basis functions, and hence the space they span contains piecewise cubic functions with knots at all *even* integers. This space will be referred to as V_{L-1}. According to the well known B-spline knot insertion algorithm [18, 30, 16] there is the following relationship

$$\phi_{L-1,j} \quad = \quad \sum_k h_{k-2j}\ \phi_{L,k} \tag{15}$$

where the sequence h is given in Appendix A.1. This process is illustrated in Figure 6.

Clearly V_{L-1} is a proper subset of V_L and thus, it is not as rich as the space V_L. Therefore, if one begins with V_{L-1}, and wants to find additional basis functions to again span V_L, more basis functions besides the $\phi_{l-1,j}$ are needed. In the wavelet methodology, this is accomplished by introducing into the basis translated copies of a special wavelet shape ψ. Just as with the B-splines, the relationship between the wavelet basis functions and the model wavelet shape is notated $\psi_{L-1,j}(t) = \psi(2^{-1}t-j)$. These new basis functions are also defined as linear combinations of the B-spline basis functions at level L:

$$\psi_{L-1,j} \quad = \quad \sum_k g_{k-2j}\ \phi_{L,k} \tag{16}$$

where the sequence g is given in Appendix A.1 and illustrated in Figure 7. There is some degree of freedom in choosing the sequence g, as long the new basis functions "fill in" the missing space between V_L and V_{L-1} [2]. Advantages to the particular choice

[2]There are some other technical requirements in order to ensure the numerical stability of the basis. These requirements insure a dual basis with certain *bi-orthogonal* properties, and hence are called bi-orthogonal wavelets. See [17].

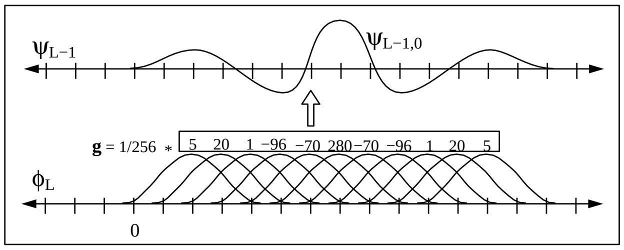


Figure 7: Eleven B-splines $\phi_{L,j}$ may be combined using the weights g to construct the wavelet function $\psi_{L-1,0}$

of g given in Figure 7 include compactness and symmetry. With this construction, we now have two alternate bases for V_L, the B-spline basis

$$\{\phi_{L,j}\} \tag{17}$$

and the *two-part* basis

$$\{\phi_{L-1,j}, \psi_{L-1,j}\} \tag{18}$$

Just as the two-part basis functions can be expressed as a combination of the B-splines basis functions (Equations (15) and (16)), so too, the B-spline basis functions can be expressed as combinations of the two-part basis functions. This is given by

$$\phi_{L,j} = \sum_k \tilde{h}_{j-2k} \; \phi_{L-1,k} + \sum_k \tilde{g}_{j-2k} \; \psi_{L-1,k} \tag{19}$$

where the sequences $\tilde{h}$ and $\tilde{g}$ are given in Appendix A.1. This process is illustrated in Figure 8. In the literature, there are many wavelet constructions, each with its own particular functions ϕ and ψ, and sequences h, g, $\tilde{h}$, and $\tilde{g}$ [3], The sequences and the particular wavelet construction described in this section are derived in [17]. This wavelet construction has been chosen because the associated ϕ is a cubic B-spline, the function ψ is symmetric with compact support, and most importantly, all of the four sequences h, g, $\tilde{h}$, and $\tilde{g}$ are finite.

[3]In this particular construction h is related to $\tilde{g}$ and g is related to $\tilde{h}$ by reversing the sign of every other element. This is not in general true for all wavelet constructions.

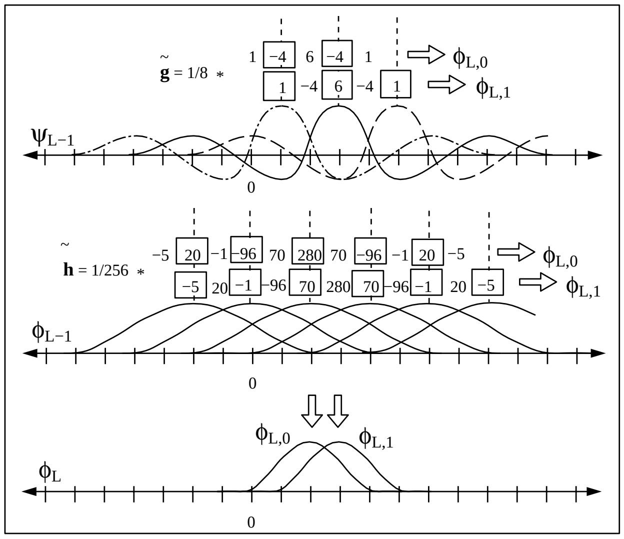


Figure 8: Two wavelets $\psi_{L-1,j}$ together with five double width B-splines $\phi_{L-1,j}$ may be combined using the weights $\tilde{g}$ and $\tilde{h}$ to construct the B-spline $\phi_{L,0}$. Three wavelets $\psi_{L-1,j}$ together with six double width B-splines $\phi_{L-1,j}$ may be combined using the weights $\tilde{g}$ and $\tilde{h}$ to construct the B-spline $\phi_{L,1}$.

The sequences have the following relationships

$$\sum_a \tilde{h}_{a-2j} h_{a-2k} = \delta_{j,k} \tag{20}$$

$$\sum_a \tilde{g}_{a-2j} h_{a-2k} = 0 \tag{21}$$

$$\sum_a \tilde{g}_{a-2j} g_{a-2k} = \delta_{j,k} \tag{22}$$

$$\sum_a \tilde{h}_{a-2j} g_{a-2k} = 0 \tag{23}$$

Where δ_{ij} is the Kroneker delta.

Proof: The first two relationships can be established using the following reasoning

$$\begin{aligned}
\phi_{L-1,j} &= \sum_a h_{a-2j} \phi_{L,a} \\
&= \sum_a h_{a-2j} \sum_k \tilde{h}_{a-2k} \phi_{L-1,k} + \tilde{g}_{a-2k} \psi_{L-1,k} \\
&= \sum_k \phi_{L-1,k} \sum_a \tilde{h}_{a-2k} h_{a-2j} + \sum_k \psi_{L-1,k} \sum_a \tilde{g}_{a-2k} h_{a-2j}
\end{aligned}$$

$$0 = \sum_k \phi_{L-1,k}(\sum_a \tilde{h}_{a-2k}h_{a-2j} - \delta_{j,k}) + \sum_k \psi_{L-1,k}(\sum_a \tilde{g}_{a-2k}h_{a-2j}) \tag{24}$$

And the result then follows from the linear independence of a basis. □

Just as the space of the functions spanned by the $\phi_{L-1,j}$ can be denoted as V_{L-1}, the space of functions spanned by $\psi_{L-1,j}$ can be denoted as W_{L-1}. Any function in V_L can be uniquely expressed as the sum of a function in V_{L-1} and a function in W_{L-1}. Symbolically this is expressed as the following direct sum

$$V_L = V_{L-1} \dot{+} W_{L-1} \tag{25}$$

Suppose some function $F(t)$ in V_L has been expressed as a linear combination of the B-spline basis function

$$F(t) = \sum_j f_{\phi_{L,j}} \ \phi_{L,j} \tag{26}$$

where the f are scalar coefficients. The coefficients needed to express the function in the two-part basis can be computed as

$$\begin{aligned} f_{\phi_{L-1,j}} &= \sum_k \tilde{h}_{k-2j} \ f_{\phi_{L,k}} \\ f_{\psi_{L-1,j}} &= \sum_k \tilde{g}_{k-2j} \ f_{\phi_{L,k}} \end{aligned} \tag{27}$$

(This process is similar to the one illustrated in Figures 6-7 except that h and g and are interchanged with $\tilde{h}$ and $\tilde{g}$).

Proof:

$$\begin{aligned} F(t) &= \sum_j f_{\phi_{L,j}} \ \phi_{L,j} \\ &= \sum_j f_{\phi_{L,j}} * (\sum_k \tilde{h}_{j-2k} \ \phi_{L-1,k} + \sum_k \tilde{g}_{j-2k} \ \psi_{L-1,k}) \\ &= \sum_j \sum_k f_{\phi_{L,j}} \tilde{h}_{j-2k} \ \phi_{L-1,k} + \sum_j \sum_k f_{\phi_{L,j}} \tilde{g}_{j-2k} \ \psi_{L-1,k} \\ &= \sum_k \sum_j f_{\phi_{L,j}} \tilde{h}_{j-2k} \ \phi_{L-1,k} + \sum_k \sum_j f_{\phi_{L,j}} \tilde{g}_{j-2k} \ \psi_{L-1,k} \\ &= \sum_j (\sum_k f_{\phi_{L,k}} \tilde{h}_{k-2j}) \ \phi_{L-1,j} + \sum_j (\sum_k f_{\phi_{L,k}} \tilde{g}_{k-2j}) \ \psi_{L-1,j} \\ &= \sum_j f_{\phi_{L-1,j}} \ \phi_{L-1,j} + \sum_j f_{\psi_{L-1,j}} \ \psi_{L-1,j} \end{aligned} \tag{28}$$

□

Intuitively speaking, the $f_{\phi_{L-1,j}}$ encode the smooth (low frequency) information about the function F , and the $f_{\psi_{L-1,j}}$ encode the fine detail (high frequency) information.

Alternatively if $F(t)$ has been represented with respect to the two-part basis, the representation with respect to the B-spline basis can be found with

$$f_{\phi_{L,j}} = \sum_k h_{j-2k}\ f_{\phi_{L-1,k}} + \sum_k g_{j-2k}\ f_{\psi_{L-1,k}} \tag{29}$$

Orthogonality

The previous section described one of the possible wavelet construction based on cubic B-splines. For certain technical reasons, that construction is called *bi-orthogonal.* There are other possible wavelet constructions based on cubic B-splines. In [16] a different sequence g is chosen (given here in Appendix A.2) so that the resulting basis functions $\psi_{L-1,j}$ are orthogonal to the B-spline basis functions $\phi_{L-1,j}$ with respect to the L^2 inner product, defined for general functions F and G as

$$\langle F(t), G(t)\rangle = \int_{-\infty}^{+\infty} F(t)G(t)\ dt \tag{30}$$

This type of construction is called *semi-orthogonal.*

In the semi-orthogonal construction, all of the functions in V_{L-1} are orthogonal to the functions in W_{L-1}. This is denoted using the orthogonal sum notation

$$V_L = V_{L-1} \oplus W_{L-1} \tag{31}$$

This is a more restrictive relationship then the direct sum relationship ($\dot{+}$).

In a semi-orthogonal construction, the smooth component of Equation (28) $S(t) = \sum_j f_{\phi_{L-1,j}}\ \phi_{L-1,j}(t)$, is the orthogonal projection of F into V_{L-1}. Thus, it is the best approximation of F in the space V_{L-1} where error is measured by

$$\mid E(t) \mid = \langle E, E\rangle^{1/2} \tag{32}$$

This property is not true for the bi-orthogonal construction.

The price paid for the semi-orthogonality in the cubic B-spline setting is the fact that the corresponding sequences $\tilde{h}$ and $\tilde{g}$ are infinite in length. (Although they do decay rapidly from their centers).

There also exist fully *orthogonal* constructions where *all* of the basis functions are orthogonal to each other (not just the basis functions on different levels as in the semi-orthogonal constructions). Modulo scaling, this implies that $h = \tilde{h}$ and $g = \tilde{g}$. The price paid for full orthogonality in the cubic B-spline setting is that the sequence g must be infinite in length, and the function ψ must have infinite support[16].

2.3.2 Bases on the Interval

In a classical wavelet construction, the index j goes from $-\infty \ldots \infty$, and V_L includes functions of unbounded support. In the contexts described in this thesis, only functions over some fixed finite interval need to be expressed, and it is important to only deal with a finite number of basis functions.

The easiest way to do this is to extend the function F, defined on an interval, to all of the real line and use a regular wavelet construction. There are a number of ways to extend F, for example, one can define F to be zero outside of the interval, or one can extend the function by making periodic copies of F. Both of these methods are unacceptable since they will require the function to either be zero at the ends, periodic (a closed B-spline curve), or have discontinuities.

Bi-orthogonal Mirrored Interval Construction

Another solution is to use a mirror-reflection of the interval [26]. This technique, borrowed from signal processing, is simple to understand and requires the least special cases in the code. Because the technical details are non-trivial, this thesis includes specific pseudo-code in Appendix A.1.

Suppose some function $F(t)$ is only defined for t in $[0 \ldots 2^L]$. This function can be extended to the entire real line by placing a mirror at 0 and a mirror at 2^L, and reflecting the function back and forth. (This is the same as reflecting the functions around 0, and then making periodic copies of the singly reflected function). This extended function is then represented as a function on the real line in V_L (piecewise cubic with C^2 continuity at the integers). The representation of this extended function

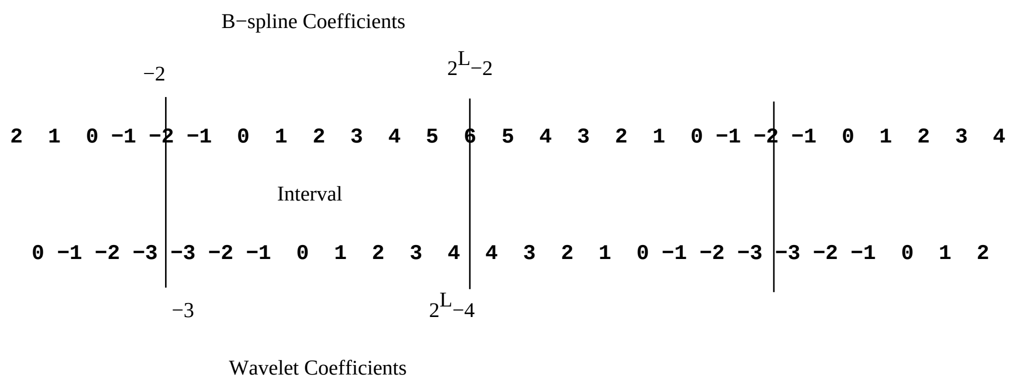


Figure 9: When a function on an interval is extended to the entire real line using mirror reflection, the corresponding B-spline and wavelet coefficients have mirrored patterns as well

will be made up of $2^L + 1$ unique B-spline coefficients $f_{\phi_{L,j}}$ with j in $\{-2 \ldots 2^L - 2\}$. The rest of the infinite number of coefficients will be defined by the mirrors. This pattern is shown in Figure 9. If the unique B-spline coefficients are stored in an array, $b[-2 \ldots 2^L - 2]$ any of the extended B-spline coefficients can be obtained with the indexing procedure `bs_index` given in Appendix A.1. The only limitation of this method is that in order for the extended function to be in V_L and have coefficients with the mirror pattern, $\frac{dF}{dt}$ must be zero at the boundaries.

After obtaining coefficients of the mirror reflected functions in the two-part basis with Equation (27), the result will have $2^{L-1} + 1$ unique coefficients $f_{\phi_{L-1,j}}$ and 2^{L-1} unique coefficients $f_{\psi_{L-1,j}}$. The reflection pattern of the ψ coefficients is slightly different than that of the ϕ coefficients since each wavelet basis function is centered half way between the centers of two B-splines. The pattern of wavelet coefficients is shown in Figure 9.

The unique ψ coefficients may be stored in an array $w[-3 \ldots 2^L - 4]$, and then any of the extended wavelet coefficients can be obtained with the indexing procedure `wave_index` given in Appendix A.1.

Given the above indexing functions, the interval version of Equation (27) can be implemented with the procedure `coef_xform_up`. The inverse transformation described by Equation (29) may be implemented by the procedure `coef_xform_down` (see Appendix A.1).

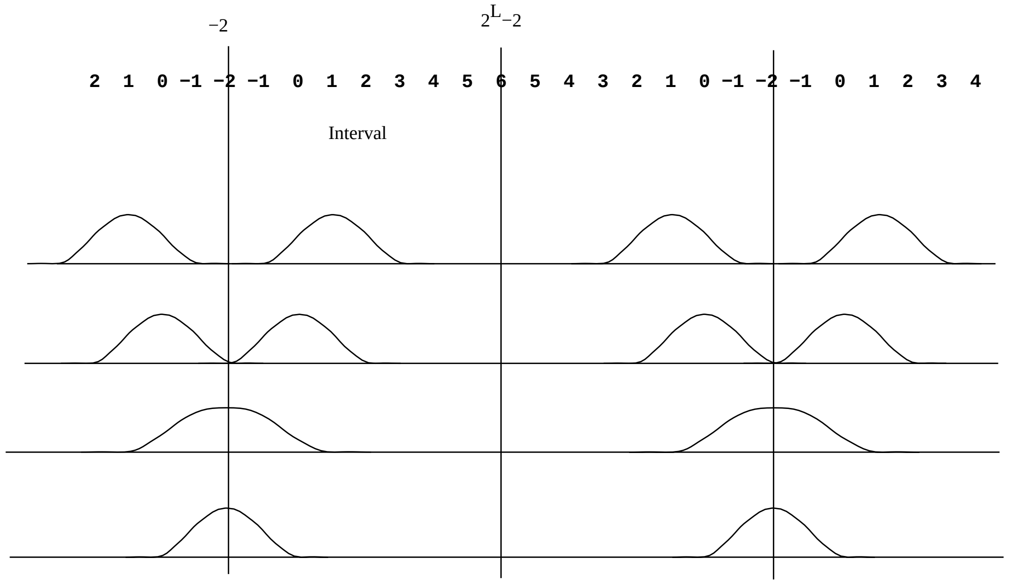


Figure 10: Each line shows a “single” basis function made up of mirror reflected copies of the B-spline basis shape. When the copies overlap (bottom line) they are summed together.

The interval wavelet transformation can be applied to the basis functions as well as the coefficients. Equations (15) and (16) are implemented in the procedure `basis_xform_up`. And finally Equation (19) may be implemented with the procedure `basis_xform_down`.

There are two different ways to view this interval construction. It can be viewed as a way to extend functions from the interval to the real line. Alternatively, it can be viewed as a way of generating basis functions on the interval. Begin with the space of all functions on the real line defined by B-spline coefficients with the mirror reflecting pattern. Because of this assumed pattern, there are only 2^L+1 true degrees of freedom to this family of functions on the real line. Corresponding to these finite degrees of freedom are 2^L+1 basis functions. Each of these “mirrored” basis functions can be constructed by adding together all of the basis functions on the real line that are affected by a single coefficient, or mirror reflected copies of it. For example, one of these new basis functions will be the set of infinite reflected copies of a single cubic B-spline basis function. Near the interval boundaries these copies will overlap, and the new basis functions will be defined by adding up this overlap (Figure 10).

If these new basis functions are restricted to the interval $[0 \ldots 2^L]$, (just discarding everything outside of the interval), then this can be considered a new set of basis functions that are contained in the interval, and these basis functions and coefficients can be manipulated with the procedures described above.

Semi-orthogonality Interval Construction

If this mirror reflection construction is applied to the semi-orthogonal cubic B-spline wavelet basis functions, the resulting wavelet basis functions near the boundaries will not be orthogonal to the mirror reflected B-spline functions with respect to the interval inner product

$$\langle F(t), G(t)\rangle = \int_0^{2^L} F(t)G(t)\ dt \tag{33}$$

Thus, if one is interested in maintaining the semi-orthogonality on the interval, mirror reflection is unacceptable and special boundary basis functions must be constructed. At this point, since the mirror reflection must be abandoned and special boundary functions must be defined, it makes sense to extend the space of functions V_L in the interval to include all C^2 functions defined over the interval $[0 \ldots 2^L]$ that are piecewise cubic between adjacent integers (simple knots at the inner integers and quadruple knots at the boundaries). In order to span this space, $2^L + 3$ basis functions are required. (Recall that when mirror reflection is employed, the function is constrained to have zero first derivative at the boundaries, and thus, there are only $2^L + 1$ basis functions).

A basis for this non-mirrored interval space is made up of *inner* basis functions, which are just those translational basis functions $\phi_{L,j}$ from Section 2.3.1 whose support lies completely within the interval, as well as three special *boundary* B-spline basis functions at each end of the interval. For the boundary basis functions, one may either choose to include the translational basis functions $\phi_{L,j}$ themselves from Section 2.3.1 whose support intersects the boundaries by just truncating those basis functions at the boundary, or else one may use the special boundary basis functions that arise from placing quadruple knots at the boundaries [6]. This complete set of basis functions will be denoted $\phi_{L,j}$ with j in $\{-3 \ldots 2^L - 1\}$, where it is understood that the first and last three basis functions are the special boundary B-spline basis functions.

The two-part basis for this space begins with the wider B-spline functions $\phi_{L-1,j}$ with j in $\{-3 \ldots 2^{L-1}-1\}$ where again the first and last three basis functions are scaled versions of the special boundary B-splines functions. The two-part basis is completed with the semi-orthogonal wavelet functions $\psi_{L-1,j}$ with j in $\{-3 \ldots 2^{L-1} - 4\}$. Here too, the *inner* wavelet basis functions are just those translational functions $\psi_{L-1,j}$ from Section 2.3.1 that do not intersect the boundaries, while the first three and the last three interval wavelet basis functions must be specially designed to fit in the interval and still be orthogonal to the $\phi_{L-1,j}$. A full description of this construction is given in [15, 61].

This interval construction, which on the real line corresponds to Equations (15) and (16), is described by the linear time procedure `basis_xform_up` that is given in Appendix A.2. As this procedure is equivalent to multiplication with a banded matrix, the inverse procedure `basis_xform_down` which describes the interval version of Equation (19) can be implemented by solving the banded linear system. This too can be done in linear time.

The interval version of Equation (29) can be implemented with the procedure `coef_xform_down`. And the inverse transformation, which is the interval equivalent of Equation (27) may be implemented by the procedure `coef_xform_up`. See Appendix A.2.

2.3.3 Complete B-spline Wavelet Basis

The reasoning that was used to construct the two-part basis can now be applied L times to construct a multilevel *wavelet basis*. Thus far, a two-part basis $\{\phi_{L-1,j}, \psi_{L-1,j}\}$ has been discussed as an alternative for the B-spline basis $\{\phi_{L,j}\}$.

Note that (roughly) half of the basis functions in the two-part basis are themselves B-spline basis functions (only twice as wide). To continue the wavelet construction, keep the basis functions $\psi_{L-1,j}$ and re-apply the reasoning of section 2.3.1 to replace the $\phi_{L-1,j}$ with $\{\phi_{L-2,j}, \psi_{L-2,j}\}$. This results in the new basis $\{\phi_{L-2,j}, \psi_{L-2,j}, \psi_{L-1,j}\}$

where $\phi_{L-2,j}(t) = \phi(2^{-2}t - j)$, and $\psi_{L-2,j}(t) = \psi(2^{-2}t - j)$.

Each time this reasoning is applied, the number of B-spline functions in the hierarchical basis is cut in half, and the new basis functions become twice as wide. In the bi-orthogonal mirror reflection interval construction, L applications can be performed and the resulting wavelet basis

$$\{\phi_{0,0}, \phi_{0,1}, \psi_{i,j}\} \tag{34}$$

can be obtained obtained, with i in $\{0 \dots L-1\}$ and j in $\{-3 \dots 2^i - 4\}$, where

$$\begin{aligned} \phi_{i,j}(t) &= \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= \psi(2^{(i-L)}t - j) \end{aligned} \tag{35}$$

This basis is made up of two wide B-splines, and translates (index j) and scales (index i) of the wavelet shape (Figure 11).

The wavelet basis is an alternative basis for V_L, but unlike the B-spline basis, it is an L level hierarchical basis. At level 0 there are two broad B-splines, and one broad wavelet. These three basis functions give the coarse description of the function. At each subsequent level going from level 0 to $L-1$, the basis includes twice as many wavelets, and these wavelets are twice as narrow as the ones on the previous level. Each level successively adds more detail to the function. If the function is sufficiently smooth in some region, then very few significant [4] wavelet coefficients will be required in that region.

The wavelet basis decomposes the space V_L into the following direct sum

$$V_L = V_0 \dot{+}_{i=0}^{L-1} W_i \tag{36}$$

The coefficients representing some function in the B-spline basis can be transformed to the wavelet basis using the procedure `coef_pyrm_up`, that makes L calls to `coef_xform_up` each time with an input vector of 1/2 the length. (Note since this transforms an n-vector to an n-vector, it can be implemented with proper indexing using linear storage).

[4] In this case, significant can be defined to be having an absolute value greater than some epsilon without incurring significant error in the representation.

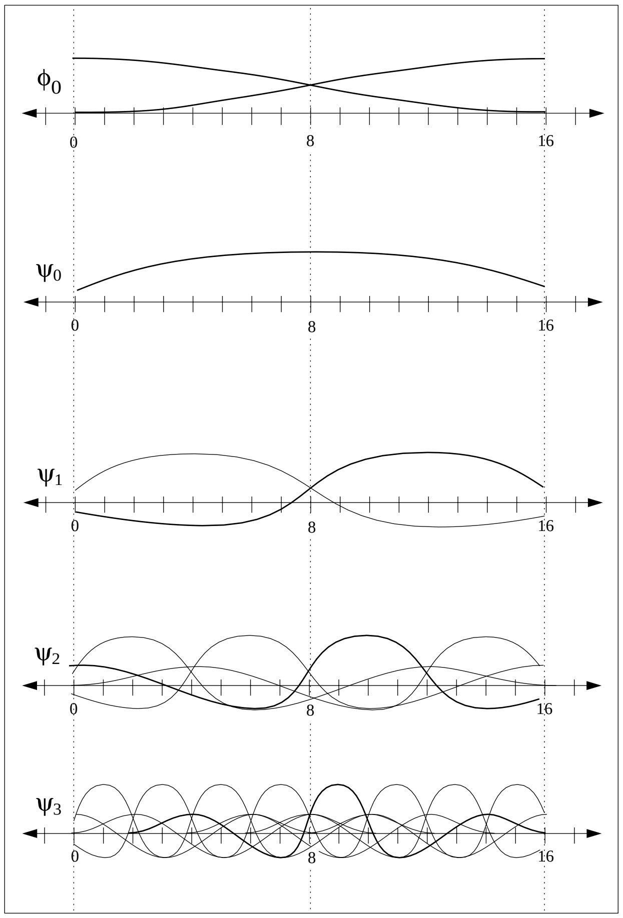


Figure 11: The wavelet basis is a hierarchical basis with L levels (in this figure $L = 4$). On the coarsest level 0, there are two wide B-splines $\phi_{0,0}$ and $\phi_{0,1}$, and one wide wavelet function $\psi_{0,0}$. On each subsequent level there are twice as many wavelet functions, that are half as wide.

```
coef_pyrm_up( $bs_{\text{in}}$[], $bs_{\text{out}}$[], $w_{\text{out}}$[][], $L$ )
    $bs_{\text{temp}}[L][] = bs_{\text{in}}[]$ ;
    for( $i = L$; $i \geq 1$; $i --$ )
        coef_xform_up( $bs_{\text{temp}}[i][]$, $bs_{\text{temp}}[i-1][]$, $w_{\text{out}}[i-1][]$, $i$ ) ;
    $bs_{\text{out}}[] = bs_{\text{temp}}[0][]$ ;
```

The inverse transformation is

```
coef_pyrm_down( $bs_{\text{in}}$[], $w_{\text{in}}$[][] , $bs_{\text{out}}$[], $L$ )
    $bs_{\text{temp}}[0][] = bs_{\text{in}}[]$ ;
    for( $i = 1$; $i \le L$; $i{+}{+}$ )
        coef_xform_down( $bs_{\text{temp}}[i-1][]$, $w_{\text{in}}[i-1][]$, $bs_{\text{temp}}[i][]$, $i$ ) ;
    $bs_{\text{out}}[] = bs_{\text{temp}}[L][]$ ;
```

Finally, the basis transformations are

```
basis_pyrm_up( $bs_{\text{in}}$[], $bs_{\text{out}}$[], $w_{\text{out}}$[][], $L$ )
    $bs_{\text{temp}}[L][] = bs_{\text{in}}[]$ ;
    for( $i = L$; $i \ge 1$; $i{-}{-}$ )
        basis_xform_up( $bs_{\text{temp}}[i][]$, $bs_{\text{temp}}[i-1][]$, $w_{\text{out}}[i-1][]$, $i$ ) ;
    $bs_{\text{out}}[] = bs_{\text{temp}}[0][]$ ;
```

and

```
basis_pyrm_down( $bs_{\text{in}}$[], $w_{\text{in}}$[][] , $bs_{\text{out}}$[], $L$ )
    $bs_{\text{temp}}[0][] = bs_{\text{in}}[]$ ;
    for( $i = 1$; $i \le L$; $i{+}{+}$ )
        basis_xform_down( $bs_{\text{temp}}[i-1][]$, $w_{\text{in}}[i-1][]$, $bs_{\text{temp}}[i][]$, $i$ ) ;
    $bs_{\text{out}}[] = bs_{\text{temp}}[L][]$ ;
```

The running time of these pyramid procedures is governed by the geometric series $(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1) = O(n)$, and hence they run in linear time. Each one of these four procedures transforms one n-vector, to another, and thus can be represented as a matrix. If $\mathbf{W}$ is the matrix of the linear transformation performed by the procedure `coef_pyrm_up`, then $\mathbf{W}^{-1}$ is the matrix of `coef_pyrm_down`, $\mathbf{W}^{-\mathbf{T}}$ is the matrix [5] of `basis_pyrm_up` and $\mathbf{W}^{\mathbf{T}}$ is the matrix of `basis_pyrm_down`.

[5]The inverse transpose of a matrix is notated here by $-T$.

Note that in the semi-orthogonal interval construction, the transformation can be applied only $L-3$ times to insure that the left boundary basis functions do not intersect the right boundary and vice versa. Otherwise, new functions would have to be devised to maintain the orthogonality constraints. The pyramid procedures for the semi-orthogonal case must be appropriately changed to only perform $L-3$ `xforms`.

Note that in this description, wavelets have been used to create an alternative basis for space V_L. But one need not stop with the wavelet basis functions at level $L-1$. Instead one can include ψ functions at greater levels, including more and more descriptive power. In fact, in the limit if one includes an infinite number of wavelet functions on finer and finer levels, the basis has the expressive ability to describe all functions [6] in L^2 (all functions with finite norm).

$$L^2 = V_0 \dot{+}_{i=0}^{\infty} W_i \tag{37}$$

Scaling

One final issue is the scaling ratio between the basis functions. Traditionally [54, 58] the wavelet functions are defined with the following scaling:

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(i-L)/2}\ \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{(i-L)/2}\ \psi(2^{(i-L)}t - j) \end{aligned} \tag{38}$$

This means that at each level up, the basis functions become twice as wide, and are scaled $\frac{1}{\sqrt{2}}$ times as tall. This is done so that the the quantity $\mid \psi_{i,j} \mid$ is independent of i. In the pyramid code, this is achieved by multiplying all of the h and g entries by $\sqrt{2}$, and all of the $\tilde{h}$ and $\tilde{g}$ by $\frac{1}{\sqrt{2}}$.

2.4 Families of Wavelets

The description of the B-spline wavelet bases has introduced many of the concepts and terminology relevant to wavelets in general. At this point the reader should be

[6]Formally, finite combinations of these functions form a dense subset of L^2

able to take the Haar construction, and express the transformations using all of the notation (h and g) developed here.

Roughly speaking, all wavelet constructions are built using two functions ϕ and ψ, and four sequences h, g, $\tilde{h}$ and $\tilde{g}$. Translations and scales of these functions are defined by Equation (38). The relationships between these functions on adjacent levels can be expressed using Equations (15, 16, 19). And these relationships can be cascaded into pyramid algorithms.

There are many other wavelet families that are discussed in the literature. In the signal processing literature, many convolution sequences (h, g, $\tilde{h}$, and $\tilde{g}$) have been developed in the context of band-pass filters used for signal compression. Corresponding to these sequences are associated ϕ and ψ functions.

Perhaps the most famous wavelets are the ones developed by Daubechies [26]. These wavelets are both fully orthogonal and have compact support. (This is impossible when the ϕ is a cubic B-spline [16]). The Daubechies wavelets are not symmetric. They also have no closed form, or piecewise expression, but they can be evaluated numerically at all dyadic rational points ($j/2^i$).

2.5 Duals and Projections

Given some function F, the question arises, how can one obtain its coefficients with respect to some basis? And if the function does not lie in the chosen finite dimensional space, then how can one obtain an approximation of F that does. These questions can be answered using duals and projections.

Given a space V_L, and a basis for that space $\phi_{L,j}$, we define a set of dual basis functions $\tilde{\phi}_{L,j}$ to be some set of functions [7] that has the property

$$\langle \phi_{L,j}, \tilde{\phi}_{L,k} \rangle = \delta_{jk} \tag{39}$$

where δ_{jk} is the Kronecker Delta. The space spanned by the dual basis functions is denoted $\tilde{V}_L$.

[7] When looking for a dual basis for some finite dimensional space, and allowing the dual functions to be arbitrary functions in L^2, there are an infinite variety of duals. Different duals will give rise to different projections.

Given some general function F in L^2 we can use the dual basis functions to *project* F into V_L.

$$P_L F \equiv \sum_j \langle F, \tilde{\phi}_{L,j} \rangle \phi_{L,j} \tag{40}$$

This operation is a projection in that if F is already in V_L, then $F = P_L F$.
Proof:

$$\begin{aligned} P_L F &= \sum_j \langle F, \tilde{\phi}_{L,j} \rangle \phi_{L,j} \\ &= \sum_j \langle \sum_k f_{\phi_{L,k}} \phi_{L,k}, \tilde{\phi}_{L,j} \rangle \phi_{L,j} \\ &= \sum_j \sum_k f_{\phi_{L,k}} \langle \phi_{L,k}, \tilde{\phi}_{L,j} \rangle \phi_{L,j} \\ &= \sum_j f_{\phi_{L,j}} \phi_{L,j} \\ &= F \end{aligned} \tag{41}$$

□

In the case that F is already in V_L, then the projection operator Equation (39) is just a decomposition operation, i.e., it is one way to obtain the coefficients of F.

Given some wavelet construction, where the basis $\phi_{L,j}$ is replaced by the two part basis $\{\phi_{L-1,j}, \psi_{L-1,j}\}$ using the sequences h and g, and given some dual basis $\tilde{\phi}_{L,j}$, there is a dual two-part basis defined by

$$\begin{aligned} \tilde{\phi}_{L-1,j} &\equiv \sum_k \tilde{h}_{k-2j}\ \tilde{\phi}_{L,k} \\ \tilde{\psi}_{L-1,j} &\equiv \sum_k \tilde{g}_{k-2j}\ \tilde{\phi}_{L,k} \end{aligned} \tag{42}$$

(Recall these are the $\tilde{h}$ and $\tilde{g}$ sequences from Equation (19).) This new basis is dual to the two-part basis:

$$\begin{aligned} \langle \tilde{\phi}_{L-1,j}, \phi_{L-1,k} \rangle &= \delta_{jk} \\ \langle \tilde{\psi}_{L-1,j}, \psi_{L-1,k} \rangle &= \delta_{jk} \\ \langle \tilde{\phi}_{L-1,j}, \psi_{L-1,k} \rangle &= 0 \\ \langle \tilde{\psi}_{L-1,j}, \phi_{L-1,k} \rangle &= 0 \end{aligned} \tag{43}$$

Proof: These orthogonality conditions are easily verified. For example

$$\begin{aligned}\langle\tilde{\phi}_{L-1,j},\phi_{L-1,k}\rangle &= \langle\sum_a \tilde{h}_{a-2j}\tilde{\phi}_{L,a},\sum_b h_{b-2k}\phi_{L,b}\rangle\\ &= \sum_a\sum_b \tilde{h}_{a-2j}h_{b-2k}\langle\tilde{\phi}_{L,a},\phi_{L,b}\rangle\\ &= \sum_a \tilde{h}_{a-2j}h_{a-2k}\\ &= \delta_{jk} \qquad (44)\end{aligned}$$

(the last implication uses Equation (20)). □

The spaces spanned by the dual two-part basis functions are denoted $\tilde{V}_{L-1}$ and $\tilde{W}_{L-1}$

Given some dual basis $\tilde{\phi}_{L,j}$ that defines a projection operator P_L from L^2 into V_L, the corresponding dual two-part basis $\{\tilde{\phi}_{L-1,j},\tilde{\psi}_{L-1,j}\}$ defines two new projections into V_{L-1} and W_{L-1}

$$\begin{aligned}P_{L-1}F &\equiv \sum_j\langle F,\tilde{\phi}_{L-1,j}\rangle\phi_{L-1,j}\\ Q_{L-1}F &\equiv \sum_j\langle F,\tilde{\psi}_{L-1,j}\rangle\psi_{L-1,j} \qquad (45)\end{aligned}$$

It can be shown that

$$P_L = P_{L-1}+Q_{L-1} \qquad (46)$$

Analogous to Equation (19), the original dual basis functions can also be expressed as a linear combination of the dual two-part functions.

$$\tilde{\phi}_{L,j} = \sum_k h_{j-2k}\ \tilde{\phi}_{L-1,k} + \sum_k g_{j-2k}\ \tilde{\psi}_{L-1,k} \qquad (47)$$

In light of Equations (42) and (47), it is clear that one can operate with the dual basis functions, just like the primal basis functions as long as one toggles the tildes (˜). There are two bases for $\tilde{V}_L$, $\tilde{\phi}_{L,j}$ and $\{\tilde{\phi}_{L-1,j},\tilde{\psi}_{L-1,j}\}$. Dual coefficients can be transformed using

$$\begin{aligned}f_{\tilde{\phi}_{L-1,j}} &= \sum_k h_{k-2j}\ f_{\tilde{\phi}_{L,k}}\\ f_{\tilde{\psi}_{L-1,j}} &= \sum_k g_{k-2j}\ f_{\tilde{\phi}_{L,k}} \qquad (48)\end{aligned}$$

and

$$f_{\tilde{\phi}_{L,j}} = \sum_k \tilde{h}_{j-2k}\; f_{\tilde{\phi}_{L-1,k}} + \sum_k \tilde{g}_{j-2k}\; f_{\tilde{\psi}_{L-1,k}} \tag{49}$$

This may be summarized as

$$\begin{aligned} \texttt{dual\_basis\_xform\_up} &\equiv \texttt{coef\_xform\_up} \\ \texttt{dual\_basis\_xform\_down} &\equiv \texttt{coef\_xform\_down} \\ \texttt{dual\_coef\_xform\_up} &\equiv \texttt{basis\_xform\_up} \\ \texttt{dual\_coef\_xform\_down} &\equiv \texttt{basis\_xform\_down} \end{aligned} \tag{50}$$

There are the dual projections into $\tilde{V}_{L-1}$ and $\tilde{W}_{L-1}$

$$\begin{aligned} \tilde{P}_{L-1}F &\equiv \sum_j \langle F, \phi_{L-1,j}\rangle \tilde{\phi}_{L-1,j} \\ \tilde{Q}_{L-1}F &\equiv \sum_j \langle F, \psi_{L-1,j}\rangle \tilde{\psi}_{L-1,j} \end{aligned} \tag{51}$$

and

$$\tilde{P}_L = \tilde{P}_{L-1} + \tilde{Q}_{L-1} \tag{52}$$

Just as one can reapply the two-part transformation L times to the basis $\phi_{L,j}$ to obtain the wavelet basis, one can also reapply the dual two-part transformation L times to the basis $\tilde{\phi}_{L,j}$ to construct the dual functions

$$\{\tilde{\phi}_{i,j}, \tilde{\psi}_{i,j}\} \tag{53}$$

which make up the dual wavelet basis

$$\{\tilde{\phi}_{0,0}, \tilde{\phi}_{0,1}, \tilde{\psi}_{i,j}\} \tag{54}$$

with i in $\{0 \ldots L-1\}$. And once again

$$\begin{array}{lcll} \texttt{dual\_basis\_pyrm\_up} & \equiv & \texttt{coef\_pyrm\_up} & = \mathbf{W} \\ \texttt{dual\_basis\_pyrm\_down} & \equiv & \texttt{coef\_pyrm\_down} & = \mathbf{W^{-1}} \\ \texttt{dual\_coef\_pyrm\_up} & \equiv & \texttt{basis\_pyrm\_up} & = \mathbf{W^{-T}} \\ \texttt{dual\_coef\_pyrm\_down} & \equiv & \texttt{basis\_pyrm\_down} & = \mathbf{W^{T}} \end{array} \tag{55}$$

The dual functions at the various levels i define the projections P_i and Q_i into the function spaces V_i and W_i, and

$$P_j = P_0 + \sum_{0 \leq i < j} Q_i \tag{56}$$

This gives two independent ways of computing a wavelet coefficient $f_{\psi_{i,j}}$ for some function F: either obtain all of the $f_{\phi_{L,j}}$ coefficients and apply the pyramid procedures, or else directly compute the inner product

$$\langle F, \tilde{\psi}_{i,j} \rangle \tag{57}$$

This gives us another way to look at Equation (27).

$$\begin{aligned} f_{\phi_{L-1,j}} &= \langle F, \tilde{\phi}_{L-1,j} \rangle \\ &= \langle F, \sum_k \tilde{h}_{k-2j} \tilde{\phi}_{L,k} \rangle \\ &= \sum_k \tilde{h}_{k-2j} \langle F, \tilde{\phi}_{L,k} \rangle \\ &= \sum_k \tilde{h}_{k-2j} f_{\phi_{L,k}} \end{aligned} \tag{58}$$

There is one last subtlety that should be mentioned. In the wavelet construction the ϕ and h are chosen so that the two-part functions $\phi_{L-1,j}$ are just wider copies of the original basis functions $\phi_{L,j}$. This is called a "two-scale" property. In the above discussion of duals, we were free to pick for the dual basis $\tilde{\phi}_{L,j}$ any set of functions that satisfied the duality constraints (Equation (39)). Thus, there is no guarantee that that the dual basis functions $\tilde{\phi}_{L-1,j}$ on level $L-1$, defined using $\tilde{h}$ in Equation (42), are just scaled versions of the original duals. However, it can be shown that under some weak conditions, there does exist a dual shape $\tilde{\phi}$ that does have the two-scale property [17]. When wavelet functions $\psi_{i,j}$ are used to form a basis for L^2 on the real line (i.e., not just V_L) by including "infinite" levels $\{i = 0 \ldots \infty\}$, then there is one *natural* dual basis, and this dual basis satisfies the two-scale property.

In addition if the wavelet basis is semi-orthogonal (or fully orthogonal), then the natural dual basis functions span the same space as the primal functions

$$\begin{aligned} V_i &= \tilde{V}_i \\ W_i &= \tilde{W}_i \end{aligned} \tag{59}$$

Orthogonal Projections

The result of projecting a function into some space V_L is an approximation of the original function. This approximate function differs from the original function by an error function $E \equiv F - P_L F$. For each space V_L, there is a special projection, called the orthogonal projection that has the property that the error, E, is orthogonal to all of V_L i.e.,

$$\forall j \ \langle F - P_L F, \phi_{L,j} \rangle = 0 \tag{60}$$

The orthogonal projection is often used for approximation because $P_L F$ is the *best* approximation to F in V_L using the L^2 norm.

$$P_L F = G \ \ s.t. \ \min_{G \in V_L} \mid F - G \mid \tag{61}$$

If one only considers dual basis functions that are members of V_L (i.e., $V_L = \tilde{V}_L$), then there is a unique dual basis. The projection that arises from taking inner products with this unique dual is the orthogonal projection of F into V_L.
Proof: Because $V_L = \tilde{V}_L$, the orthogonality condition may be stated as

$$\forall j \ \langle F - P_L F, \tilde{\phi}_{L,j} \rangle = 0 \tag{62}$$

The orthogonality of this projection may be verified as:

$$\begin{aligned} \forall j \ \langle F - P_L F, \tilde{\phi}_{L,j} \rangle &= 0 \\ \langle F, \tilde{\phi}_{L,j} \rangle - \langle P_L F, \tilde{\phi}_{L,j} \rangle &= 0 \\ \langle F, \tilde{\phi}_{L,j} \rangle &= \langle P_L F, \tilde{\phi}_{L,j} \rangle \\ &= \langle \sum_k \langle F, \tilde{\phi}_{L,k} \rangle \phi_{L,k}, \tilde{\phi}_{L,j} \rangle \\ &= \sum_k \langle \langle F, \tilde{\phi}_{L,k} \rangle, \langle \phi_{L,k}, \tilde{\phi}_{L,j} \rangle \\ &= \langle F, \tilde{\phi}_{L,j} \rangle \end{aligned} \tag{63}$$

□

When using orthogonal wavelets, the basis functions are their own duals, and the associated projection is orthogonal. When using semi-orthogonal wavelets, the natural duals lie in the same space, and so the natural projection is the orthogonal projection. When the wavelet basis is merely bi-orthogonal (and not semi-orthogonal), then the natural duals do not lie in the same space, and so the natural projection is not the orthogonal projection [8].

2.5.1 Vanishing Moments

Since taking inner products is one way to express the projection of some function into a basis, it is important to study the inner product properties of basis functions. In the Haar basis, a wavelet coefficient is near zero if the represented function is nearly constant over that region since for any constant C,

$$\begin{aligned} \int dt\ \psi(t)C &= C\int dt\ \psi(t) \\ &= 0 \end{aligned} \tag{64}$$

Wavelets generalize this notion to higher order using the concept of *vanishing moments.* A function ψ has M vanishing moments if

$$\langle \psi(t), t^i \rangle = \int dt\ \psi(t)t^i = 0, \quad i = 0, \ldots, M-1 \tag{65}$$

If some function F is well represented by a low order polynomial ($\leq M$) over the support of the basis function $\psi_{i,j}$, and $\psi_{i,j}$ has M vanishing moments, then the corresponding wavelet coefficient $F_{\tilde{\psi}_{i,j}}$ will be nearly zero. The number of vanishing moments measures the compression ability of a wavelet expressed in polynomial degree.

The cubic B-spline wavelets discussed in this section have four vanishing moments.

2.6 Multidimensional Wavelets

Basis functions of two (or more) variables, are needed for the applications discussed in this thesis. Two dimensional functions are needed to describe surfaces, and four dimensional functions are needed to fully describe the light transport in 3D environments.

[8]In a geometric setting, such projections are called oblique.

Given some basis $\phi_{L,j}$ for V_L, a natural basis for the bivariate space $V_L \times V_L$ is the tensor product basis

$$\phi_{L,j,k}(s,t) = \phi_{L,j}(s)\phi_{L,k}(t) \tag{66}$$

2.6.1 Standard Basis

Given some wavelet basis $\{\phi_{0,j}, \psi_{i,j}\}$ for V_L, the natural bivariate basis is the standard tensor product wavelet basis

$$\begin{array}{ll} \phi_{0,j}(s)\phi_{0,l}(t) & \phi_{0,j}(s)\psi_{k,l}(t) \\ \psi_{i,j}(s)\phi_{0,l}(t) & \psi_{i,j}(s)\psi_{k,l}(t) \end{array} \tag{67}$$

Given some bivariate function $F(s,t)$ in $V_L \times V_L$, which is represented in the tensor product B-spline basis

$$F(s,t) = \sum_{i,j} f_{\phi_{L,i},\phi_{L,j}} \phi_{L,i}(s)\phi_{L,j}(t) \tag{68}$$

the standard tensor product wavelet coefficient can be obtained by performing the standard bivariate `stan_coef_pyrm_up` procedure. Given the original coefficients in a two dimensional matrix tableaux indexed by i and j, this procedure first performs the univariate `coef_pyrm_up` procedure on all the rows resulting in a new matrix of coefficients. The procedure then applies the univariate `coef_pyrm_up` procedure on all of the columns. The resulting numbers are the standard tensor product wavelet coefficients.

The tensor process described in this section can be repeated again, to obtain a 4variate standard tensor product basis.

2.6.2 Non-Standard Basis

One feature of the standard basis, is that the basis functions $\psi_{i,j}(s)\psi_{k,l}(t)$ mix univariate wavelet functions from different resolution levels $i \neq k$. For radiosity this is unfortunate because, as we will see, the resulting matrix is not as sparse as possible. Also for geometric modeling the standard representation is not very natural, because it does not allow one to decompose bivariate functions into separate resolution levels.

This inter-level coupling can be avoided if an L level bivariate basis is directly constructed. For univariate functions, the wavelet construction began with the two-part basis. For bivariate functions, the non-standard tensor product wavelet basis construction begins with the *four-part basis.*

To construct the four-part basis for the space $V_L \times V_L$ spanned by $\phi_{L,j}(s)\phi_{L,k}(t)$, one starts with $\phi_{L-1,j}(s)\phi_{L-1,k}(t)$ (tensor product B-spline functions that are twice as wide) and completes the basis by adding in the basis functions $\psi_{L-1,j}(s)\psi_{L-1,k}(t)$ and $\psi_{L-1,j}(s)\phi_{L-1,k}(t)$ and $\phi_{L-1,j}(s)\psi_{L-1,k}(t)$. Note that with univariate functions, only one special function $\psi(t)$ was needed to complete the basis, while with bivariate functions three types of functions need to be added. With this choice, transformations between the tensor B-spline and the tensor four-part bases can be accomplished with the procedures
`nonstan_coef_xform_up`,
`nonstan_coef_xform_down`,
`nonstan_basis_xform_up`, and
`nonstan_basis_xform_down`.

The procedure `nonstan_coef_xform_up` works as follows: The coefficients $f_{\phi_{L,i},\phi_{L,j}}$ are placed in a "matrix" according to their indices i and j. Each row of this matrix is transformed using `coef_xform_up` resulting in a new matrix of coefficients. Now each column of this resulting matrix is submitted to `coef_xform_up`, resulting in the four-part coefficients $f_{\phi_{L-1,j},\phi_{L-1,k}}$ and $f_{\phi_{L-1,j},\psi_{L-1,k}}$ and $f_{\psi_{L-1,j},\phi_{L-1,k}}$ and $f_{\psi_{L-1,j},\psi_{L-1,k}}$.

Just as with univariate functions, reapplication of this reasoning to the $\phi_{L-1,j}(s)\phi_{L-1,k}(t)$ functions results in another alternative basis, and after L applications, the non-standard wavelet basis is obtained

$$\begin{array}{ll} \phi_{0,j}(s)\phi_{0,k}(t) & \phi_{i,j}(s)\psi_{i,k}(t) \\ \psi_{i,j}(s)\phi_{i,k}(t) & \psi_{i,j}(s)\psi_{i,k}(t) \end{array} \tag{69}$$

(with i in $\{0 \ldots L-1\}$).

In this non-standard wavelet basis, all of the bivariate basis functions are products of univariate basis functions from the same resolution level i.

Again, as with univariate functions, one can transform between the bivariate tensor B-spline and the bivariate tensor non-standard wavelet bases bases using the

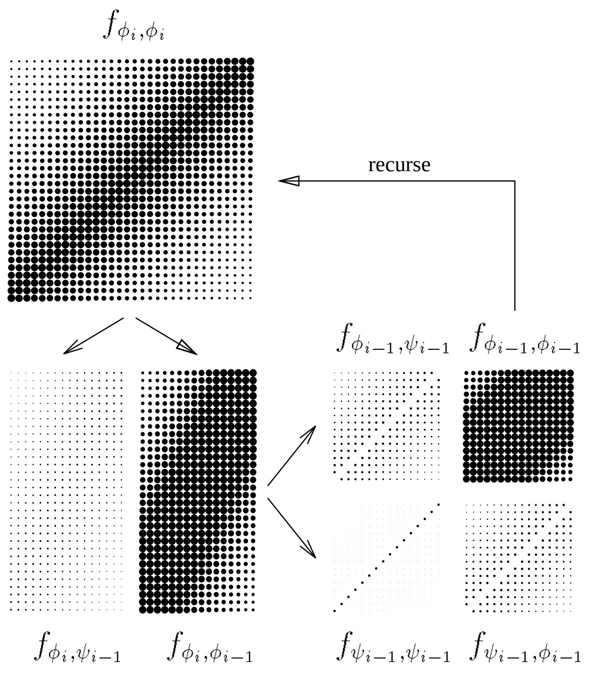

f_{ϕ_i,ϕ_i}
recurse
$f_{\phi_{i-1},\psi_{i-1}}$
$f_{\phi_{i-1},\phi_{i-1}}$
$f_{\phi_i,\psi_{i-1}}$
$f_{\phi_i,\phi_{i-1}}$
$f_{\psi_{i-1},\psi_{i-1}}$
$f_{\psi_{i-1},\phi_{i-1}}$

Figure 12: The 2D non-standard pyramid algorithm (from [40]).

procedures

`nonstan_coef_pyrm_up`,

`nonstan_coef_pyrm_down`,

`nonstan_basis_pyrm_up`, and

`nonstan_basis_pyrm_down`

where the procedure `nonstan_coef_pyrm_up` makes L calls to `nonstan_coef_xform_up`, each time with an input "matrix" of 1/4 the size (Figure 12).

A 4variate non-standard basis can be constructed using the method described in this section. There are other multivariate wavelet bases that are not based on univariate wavelet bases. This thesis will not discuss such bases.

Chapter 3

Radiosity

The next two chapters discuss how wavelets may be used to efficiently solve the integral equation arising in the radiosity formulation of global illumination problems [1]. In a global illumination problem, one is given the geometric description of an environment and wants to simulate the interreflection of light to create a realistic looking image of the environment that is faithful to the physics governing light transfer. This simulation is useful for architects who want to know how bright different regions of some planned structure will be. It is also useful for anyone interested in producing computer generated images that look realistic, for example people in the film industry making special effects. Although the physics of light transfer is well understood, a complete simulation, following the path of the innumerable photons, is intractable. Instead researchers have searched for practical methods that numerically approximate the solution. The radiosity method is one such approach, where an approximate global illumination solution is found by constructing and solving a linear system. Unfortunately, to obtain a detailed solution, one must solve a large dense matrix equation. This can become quite expensive. This chapter reviews the radiosity formulation. The following chapter will discuss how a wavelet basis can be used to more efficiently solve radiosity problems. In particular this thesis will describe how the wavelet basis gives rise to a sparse linear system, which can be computed and solved quickly.

[1]These chapters represent work done jointly with Peter Schröder and our advisors Professors Michael F. Cohen and Pat Hanrahan and is also reported in [40, 65].

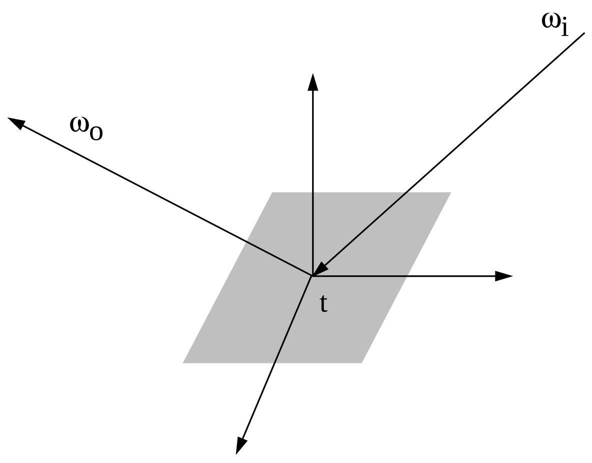


Figure 13: The BRDF is parameterized by a location t on a surface, as well as incoming and outgoing directions ω_i and ω_o.

3.1 A Little Background

The earliest attempts to render realistic looking images only focused on *local reflectance* properties. The goal was to mimic the way that real objects respond to real light sources. Formally speaking, the local reflectance properties of a surface are defined by its bidirectional reflectance function (BRDF)

$$f_r(\omega_i, s, \omega_o, \lambda) \tag{70}$$

which describes the amount of light from an incoming beam centered in the direction ω_i with wavelength λ [2] that is reflected in the outgoing direction ω_o at the surface point s (see Figure 13). (See [23] for a comprehensive discussion of photometric units). Thus given a point light source in some direction ω_i, and an eye placed in direction ω_o, the BRDF can be used to compute the the appearance of the surface point s to the eye. This can be used to compute the value of the corresponding pixel of a graphical image. Many simple lighting models, such as Phong and Torrance-Sparrow have been developed [29], and important research on BRDFs continues to the present time [50, 64].

[2] In computer graphics, color is often handled by sampling the visible spectrum at three locations for red, green, and blue values. This thesis will have no further discussion of wavelength, and λ will be dropped.

While the local reflectance methods helped introduce a great deal of realism to computer generated images, these local methods lack any global illumination effects. In a realistic environment, the light incident at a surface point results from complicated interactions involving all of the geometry of the environment. Important global illumination effects include occlusion and interreflection. An example of an occlusion effect is the complicated shadow patterns (umbrae and penumbrae) that arise when some light source is partially blocked by a surface. A simple example of an interreflection effect is color bleeding, where some surface (A) takes on the color of a non-emitting nearby colored surface (B), because of the color of light that (B) reflects. In this global perspective every surface point is responsible for receiving from and reflecting to every other surface point.

One of the first algorithms designed to account for some degree of global illumination was *ray tracing* [77]. The basic idea is simple. A ray is cast from the "eye" of the viewer through each pixel of the computer screen window into the 3D environment, until this ray hits some surface point. To calculate the brightness of that surface point, and hence the intensity of that pixel, *shadow* rays are cast towards point light sources. If a shadow ray intersects some other surface (before hitting the light), then that light source is blocked, and none of its light is propagated to the surface point in question. If the shadow ray is not blocked, then the BRDF is used to calculate the intensity of light reflected back in the direction of the eye. Using this method, sharp shadow regions, that arise from point light source, can be calculated.

If the ray cast from the eye, hits a surface with specular reflectance (such as a mirror, or suspended glass ball), then another ray is propagated in the "bounce" direction. This ray is then recursively treated as an eye ray, and the light calculated for that ray is then bounced back towards the eye.

Early ray tracing methods only accounted for direct reflection of point light sources at non-specular surfaces, and indirect reflection at specular surfaces. Later, these methods were extended to include area light sources and non-specular interreflection using the method of *distribution ray tracing* [24]. To account for area light sources, a large collection (distribution) of shadow rays are fired towards area light sources. Each of these shadow rays is associated with some fraction of the area light source, and propagates its energy back to the surface point. If all of the shadow rays are blocked, then the surface point is shadowed. If all of the shadow rays hit the light, then the surface point is fully illuminated. If only some fraction of rays are blocked, the surface point is in a penumbral region.

A similar method is employed by distribution ray tracing to account for indirect reflection at surfaces with directionally diffuse BRDFs (e.g., a glossy but not mirrored surface such as a waxed tile floor). When an eye ray hits a glossy surface, a large distribution of rays are sent to sample the cone of the bounce direction. These rays are recursively treated like new eye-rays, and the energy brought back by these rays is summed up using the BRDF to return the proper amount of energy to the pixel. Using distribution ray tracing, many stunning images have been created. In theory, these methods can be extended to include very general BRDFs such as diffuse reflectance, but this requires sending an immense amount of rays to sample the entire hemisphere of reflection (there being no preferred bounce direction).

The radiosity algorithm [38] was introduced as a method for rendering an image with primarily diffuse reflectance, and area light sources. In the radiosity paradigm, the BRDFs of the surfaces are assumed to be Lambertian diffuse (light is reflected with the same brightness in all directions over the hemisphere), and the light sources are assumed to be emitting light in a diffuse manner. With these assumptions, the brightness at every point on a surface can be described by one value which is called *radiosity* [3].

The environment is meshed into a set of surface regions, called elements, and the assumptions is made that the radiosity is constant over each element, and that the environment is in energy equilibrium. This gives rise to the following linear system.

$$\forall i \quad b_i = e_i + \rho_i \sum_j b_j F_{i,j} \tag{71}$$

[3] Radiosity has units $\frac{watts}{meters^2}$.

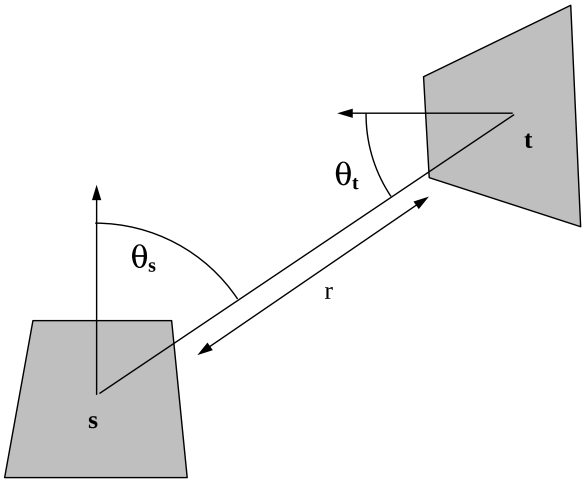


Figure 14: The configuration of two polygons. The lines with arrowheads are normals of the polygons.

or in matrix form:

$$\begin{bmatrix} 1-\rho_1F_{1,1} & -\rho_1F_{1,2} & -\rho_1F_{1,3} & . & . & -\rho_1F_{1,n} \\ -\rho_2F_{2,1} & 1-\rho_2F_{2,2} & -\rho_2F_{2,3} & . & . & -\rho_2F_{2,n} \\ . & & & & & . \\ . & & & & & . \\ -\rho_{n-1}F_{n-1,1} & & & & & . \\ -\rho_nF_{n,1} & . & . & . & . & 1-\rho_nF_{n,n} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_{n-1} \\ b_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ . \\ . \\ e_{n-1} \\ e_n \end{bmatrix}$$

where:

b_i	= the radiosity of the i^{th} element
ρ_i	= the reflectivity of the i^{th} element
e_i	= the emission of the i^{th} element
$F_{i,j}$	= the form factor from element i to element j
	= the fraction of power leaving element i arriving directly at element j

This equation states that the radiosity of an element is equal to the amount of emitted radiosity of that element, plus the reflected (ρ) radiosity that is gathered from the entire environment ($\sum_j$). The form factor, $F_{i,j}$, defines the fraction of power leaving element i that arrives at element j, and can be calculated as the definite integral [4] over the elements , $_i$, $_j$,

$$\frac{1}{a_i} \int_{\Gamma_i} dt \int_{\Gamma_j} ds \, \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} \, V_{st} \tag{72}$$

Where a_i is the area of element i, r_{st} is the distance between point t and point s, and the visibility, V_{st} is 1 if point s can see point t and 0 otherwise, (see Figure 14) [5].

This algorithm raises several computational issues that have motivated a great deal of research. The form factor integrals Equation (72), must be computed, and many methods to do this have since been developed [23]. It should also be clear that the mesh of elements chosen has a great impact on the quality of the solution; in regions where the true radiosity varies greatly (for example shadow boundaries) many small elements are needed to express the variation, thus methods for mesh generation have also generated a great deal of research[23]. Once the solution (b) is computed one can simply output flat shaded surface elements to the screen, but one may also use more sophisticated methods such displaying Gouraud shaded polygons, or employing some more complicated method of *reconstruction*[63].

Some research has focussed on methods for solving the linear system (Equation (71)). Obviously one could use a direct matrix inversion technique such as Gaussian elimination, but this takes $O(n^3)$ time. In [21] it was noted this matrix is *diagonally dominant*, and hence can be solved by relaxation methods such as Gauss-Seidel or Jacobi iteration, which converge to acceptable solutions much quicker than direct inversion methods. This was improved upon in [20] where the progressive radiosity (PR) solution method was introduced. PR operates by keeping track of unshot energy and "shooting" it around the environment until convergence occurs. In [39], it was shown that PR is related to the Southwell method of solving linear systems.

[4]In all of the integrals presented in this chapter are parameterized with respect to geometric size. For 3D radiosity then, dt and ds are unit area elements. For 2D Flatland radiosity dt and ds are unit length elements.

[5]It may seem counterintuitive that the we use $F_{i,j}$ in the linear system and not $F_{j,i}$. But this is a result of measuring in units of radiosity (power/area) instead of power and using the reciprocity relationship $A_iF_{i,j} = A_jF_{j,i}$ [38].

Just as ray tracing was first designed for specular reflectance, and then extended to more diffuse BRDFs, radiosity which was first designed for specular reflectance has since been extended to handle more specular-like BRDFs [69].

3.2 Integral Equations

Kajiya [49] put this whole topic in formal terms when he explained that all these methods were really trying to solve the integral *rendering equation*.

$$L(s,\omega_o) = L_e(s,\omega_o) + \int dt\; V_{st}\; L(t,\omega_{ts})\, \frac{\cos\theta_s \cos\theta_t}{r_{st}^2}\, f_r(\omega_{st}, s, \omega_o) \tag{73}$$

$L(s,\omega_o)$ is the outgoing radiance from surface point s in the direction ω_o. L_e is the out going emitted radiance and ω_{st} is the direction of the ray from point s to point t. The angles θ and the radius r are given in Figure 14 [6]. This equation states that the outgoing radiance L at surface point s in the outgoing direction ω_o is equal to the emitted radiance L_e at that point, plus the reflected radiance at that point. To compute the reflected radiance, one must integrate over the rest of the environment $\int dt$, check if the two points are mutually visible V_{st}, collect the radiance at these points $L(t,\omega_{ts})$, attenuate by a geometric term (the term with the cos and $1/r^2$), and multiply by the BRDF f_r. The unknown of this equation is the function L.

If the BRDF is assumed to be Lambertian diffuse, then the equation reduces to the radiosity integral equation [7].

$$B(s) = E(s) + \rho(s) \int dt\; V_{st}\; B(t)\, \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} \tag{74}$$

[6] L has units of $\frac{watts}{meters^2\, steradians}$.
$f_r \cos\theta_s$ has units of $\frac{1}{steradians}$.
dt has units of $meters^2$.
$\frac{\cos\theta_t dt}{r^2}$ has units $\frac{steradians}{meters^2}$.

[7] By making the assumption of diffuse reflectance, the BRDF becomes independent of the directions ω_i and ω_o. This diffuse BRDF is only a function of s and can be expressed as $\frac{\rho(s)}{\pi}$. Since the outgoing radiance is equal in all directions, the radiance can be expressed as $\frac{B(s)}{\pi}$. $B(s)$ is only a function of s and it measures radiosity which has units $\frac{watts}{meters^2}$. $\frac{1}{\pi}$ has units $\frac{1}{steradians}$. Similarly the emitted radiance can be expressed as $\frac{E}{\pi}$. This $\frac{1}{\pi}$ can be multiplied out of the integral equation leaving an equation for the radiosity B.

where $B(s)$ is the radiosity at point s and $\rho(s)$ is the diffuse reflectivity fraction. The unknown function to be solved for is B, the radiosity function. This equation is often written as

$$B(s) = E(s) + \int dt\ K(s,t)B(t) \tag{75}$$

where the kernel function is

$$K(s,t) = \rho(s)\frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2}V_{st} \tag{76}$$

This can be written in operator notation as

$$B = E + \mathcal{K}B \tag{77}$$

where $\mathcal{K}$ is the associated integral operator [8].

3.3 Finite Element/Galerkin Radiosity

From this perspective, Kajiya explained that distribution ray tracing was just an application of the *Monte Carlo* method for solving solving the rendering integral equation (Equation (73)). In [45], Heckbert explained how the radiosity method, was actually an application of the *Galerkin/finite element* method for solving the radiosity integral equation (Equation (74)).

The finite element method begins by orthogonally projecting E into some finite dimensional function space V_L, spanned by n basis functions $\phi_{L,j}$.

$$E \approx \hat{E} = \sum_j e_{\phi_{L,j}}\phi_{L,j} \tag{78}$$

and then assuming that the unknown function B also lies in V_L

$$B \approx \hat{B} = \sum_j b_{\phi_{L,j}}\phi_{L,j} \tag{79}$$

This results in the integral equation:

$$\hat{B}(s) = \hat{E}(s) + \int dt K(s,t)\hat{B}(t) \tag{80}$$

[8]For ease of exposition, B will be described as a univariate function and K as a bivariate function. In 3D radiosity, B is a function defined over the domain of surfaces, and so is bivariate, while K is 4variate.

Unfortunately this equation does not in general have a solution $\hat{B}$ that resides in V_L. Heckbert describes two ways that the equality can be relaxed. In *point collocation*, equality is only enforced at a finite discrete set of points. This gives rise to a linear system with n variables. In the *galerkin method*, it is assumed that the residual is orthogonal to V_L.

$$\left(\hat{B}(s) - \hat{E}(s) + \int dt K(s,t)\hat{B}(t)\right) \perp V_L \tag{81}$$

This is also a set of linear constraints, and also results in a linear system.

The Galerkin orthogonality constraint can be expressed in the language of projections (Section 2.5), given the unique basis $\tilde{\phi}_{L,j}$ that is dual to $\phi_{L,j}$ and also lies is V_L (i.e., $V_L = \tilde{V}_L$) as follows

$$\hat{B}(s) = \hat{E}(s) + \sum_i \left\langle \int dt K(s,t)\hat{B}(t), \tilde{\phi}_{L,i}(s) \right\rangle \phi_{L,i}(s) \tag{82}$$

Using projection notation, this equation can be written as

$$\hat{B} = \hat{E} + P_L \mathcal{K} \hat{B} \tag{83}$$

and since $\hat{B}$ by definition lies in V_L, $\hat{B} = P_L \hat{B}$, so the equation can also be written as

$$\hat{B} = \hat{E} + P_L \mathcal{K} P_L \hat{B} \tag{84}$$

In words, we *operate* on (integrate against the kernel) the finite dimensional function $\hat{B}(t)$. After having been operated on, the resulting function generally no longer lies in the finite dimensional function space, so the function is orthogonally reprojected into V_L using the dual basis $\tilde{\phi}_{L,i}(s)$.

This sets up the n constraints ($\forall i$)

$$\begin{aligned} b_{\phi_{L,i}} &= e_{\phi_{L,i}} + \left\langle \int dt K(s,t)\hat{B}(t), \tilde{\phi}_{L,i}(s) \right\rangle \\ &= e_{\phi_{L,i}} + \left\langle \int dt K(s,t) \sum_j b_{\phi_{L,j}} \phi_{L,j}(t), \tilde{\phi}_{L,i}(s) \right\rangle \\ &= e_{\phi_{L,i}} + \sum_j b_{\phi_{L,j}} \left\langle \int dt K(s,t) \phi_{L,j}(t), \tilde{\phi}_{L,i}(s) \right\rangle \\ &= e_{\phi_{L,i}} + \sum_j b_{\phi_{L,j}} k_{i,j} \end{aligned} \tag{85}$$

where the $b_{\phi_{L,i}}$ are the n unknowns and

$$\begin{aligned} k_{i,j} &= \left\langle \int dt K(s,t)\phi_{L,j}(t), \tilde{\phi}_{L,i}(s) \right\rangle \\ &= \int ds \int dt K(s,t)\tilde{\phi}_{L,i}(s)\phi_{L,j}(t) \end{aligned} \tag{86}$$

and

$$e_{\phi_{L,i}} = \langle E, \tilde{\phi}_{L,i} \rangle = \int ds E(s)\tilde{\phi}_{L,i} \tag{87}$$

Or in matrix terms

$$\mathbf{b} = \mathbf{e} + \mathbf{K}\mathbf{b} \tag{88}$$

There are n^2 matrix terms, where n is the number of basis functions used to describe B. Computing these terms is the most expensive part of radiosity calculations. (Because the system is well conditioned, very few relaxation iterations are required for convergence and so solving the system is somewhat less expensive.)

When the box functions are chosen as the basis $\phi_{L,i}$, then the dual function $\tilde{\phi}_{L,i}$ is a box function with height $1/a_i$, and this linear system is identical to the classical radiosity linear system (Equation (71)).

When relaxation methods, such as Jacobi iteration are used to solve a radiosity linear system, the main operation of the iteration is multiplying some vector by the matrix $\mathbf{K}$, or in operator notation, performing the operation $P_L \mathcal{K} P_L B = P_L \int dt K(s,t) P_L B(t)$ on some function B. This operation is often called "shooting". If the input is a radiosity function B, the output function is the result of shooting that radiosity once through the environment.

Note that the $k_{i,j}$ terms can also be viewed as the coefficients of the function K projected into the bivariate basis $\phi_{L,i}(s)\tilde{\phi}_{L,j}(t)$.

$$\hat{K} \equiv P_{Ls}\tilde{P}_{Lt}K(s,t) = \sum_{ij} k_{i,j}\phi_{L,i}(s)\tilde{\phi}_{L,j}(t) \tag{89}$$

(where the s and t projection subscripts designate projections of K with respect to the correct variable).

This projected K can be used as another way to write out the projected integral operator:

$$\begin{aligned}\int dt\hat{K}(s,t)B(t) &= \int dt(P_{Ls}\tilde{P}_{Lt}K(s,t)) * B(t) && (90)\\ &= \int dt(P_{Ls}K(s,t)) * (P_L B(t)) \\ &= P_L\left(\int dtK(s,t)(P_L B(T))\right) && (91)\end{aligned}$$

This gives two identical ways to express the same operation: either project B, integrate K with it, and reproject the result (Equation (90)), or project K and then integrate the resulting function with B (Equation (91)).

Given that $V_L = \tilde{V}_L$, and $P_L = \tilde{P}_L$, one can also obtain the same projected kernel $\hat{K}$ by expanding K into the symmetric basis $\tilde{\phi}_{L,i}(s)\tilde{\phi}_{L,j}(t)$.

$$k_{i,j} = \int ds \int dtK(s,t)\phi_{L,j}(t)\phi_{L,i}(s) \qquad (92)$$

This symmetric representation has the advantage of only one type of basis function under the integral. The matrix made up of these terms will be denoted as $\tilde{\mathbf{K}}$ which is a symmetric matrix (modulo ρ). When this form of the integral operator

$$\tilde{P}_L\mathcal{K}P_L \qquad (93)$$

is applied to the function B expressed with respect to the primal basis $\phi_{L,j}(t)$, the resulting function is expressed with respect to the dual basis $\tilde{\phi}_{L,i}(s)$, and must be transformed back to the primal basis for further computation.

3.3.1 Choosing a Basis

It is important to remember that the projected equation is only an approximation to the original integral equation. Projections into different finite dimensional spaces will result in different approximations with differing amounts of error and different types of error. In general, the projection error is $O(h^{p+1})$ where h is the size of the grid elements, and p the degree of the polynomial used. Thus one can reduce the error by using higher order basis functions or smaller elements. Basis functions of higher order polynomials can also result in smoother reconstructed radiosity solutions leading to fewer visual artifacts. However, higher order basis functions require more work to evaluate the associated inner products, possibly offsetting potential savings.

The original classical radiosity algorithms (CR) used the box basis. Heckbert studied the use of a linear basis (in the context of 2D radiosity in a plane)[45]. Troutman and Max discuss the use of a linear basis in 3D radiosity[75]. And Zatz has studied using higher order Galerkin methods[82]. These higher order methods are referred to as Galerkin Radiosity (GR).

This thesis discusses using a variety of wavelet bases for radiosity. Some wavelet bases offer alternative bases for familiar spaces that have been used in finite element methods (e.g., Haar basis), while others span new spaces that have not been explored.

The advantage of a wavelet basis is that its hierarchical structure allows the kernel function and the associated integral operator to be expressed with a sparse matrix representation. A hierarchical representation for radiosity was explored in the hierarchical radiosity method of Hanrahan et al. to obtain a fast radiosity method[44]. As will be discussed, the hierarchical radiosity method can be understood as a first order wavelet method.

Chapter 4

Wavelet Radiosity

This chapter will discuss using a wavelet basis to represent the *illumination function* and the *kernel function*. The illumination function is a function that describes the light intensity over all of the surface points in a geometric environment and the kernel function is a function that describes the energy interaction between pairs of points in the environment. Because in many regions (over the space of pairs of points), the kernel function is smooth, the wavelet basis is able to compress the representation resulting in a sparse radiosity matrix. The sparseness of this matrix allows for a more efficient solution method then possible with classical radiosity approaches where a dense matrix of equations must be solved. This represents work done jointly with other researchers at Princeton University, and the presentation here is adapted from the papers [40, 65]. The use of wavelet functions to quickly solve integral equations was discovered by [9, 2, 3]. This important discovery allows a large class of integral equations to be expressed in a wavelet basis by a sparse matrix with only $O(n)$ significant entries. When a classical finite element basis is used, the resulting matrix generally has $O(n^2)$ significant entries. Much of the theoretical discussion given in this chapter is based on that work.

4.1 Contribution

The contribution of the wavelet approach to radiosity described in this chapter is both theoretical and practical.

The use of a wavelet formulation gives rise to a more elegant formulation of the hierarchical radiosity method (HR) of Hanrahan et al. [44]. hierarchical radiosity uses a hierarchical representation of the geometric environment in order to simulate the energy transfer of light in the environment using only $O(n)$ interaction terms instead of the usual $O(n^2)$. This makes it the most efficient method know to solve radiosity problems. This chapter explains how wavelets can be used to cast hierarchical radiosity in the context of the more traditional Galerkin/finite element method. In particular hierarchical radiosity can be viewed as a wavelet approach based on the simplest wavelet, the Haar wavelet.

This formulation not only expresses hierarchical radiosity in an elegant theoretical framework, but it also gives rise to more efficient radiosity algorithms that can produce more accurate solutions with less computation. By posing the hierarchical methods in the wavelet context, one is free to move beyond the Haar basis and explore the tradeoff's of using a variety of different wavelet bases. In particular this thesis describes a new set of basis functions dubbed "flatlets" that have been developed specifically for radiosity. This chapter explores some of these tradeoffs, describes an implementation, and discusses some experimental results.

4.2 Hierarchical Radiosity

This section briefly describes a hierarchical radiosity method described by Hanrahan et al. [44]. Later sections will discuss the relationship between this algorithm and wavelet based methods.

In classical radiosity implementations, the radiosity integral equation is solved by an iterative method such as Jacobi iteration. Each iteration can be thought of as propagating some energy through the environment, applying the operator $P_L \mathcal{K} P_L$ to some current radiosity solution B. To facilitate this energy propagation, each of the surfaces in the environment propagates its energy to all of the other surfaces. In order for one surface (called the "shooter") to propagate its energy to some other surface (called the "receiver"), each of the two surfaces is meshed into a set of smaller surface elements. The energy distribution is represented using one coefficient for each mesh element. Energy is then propagated from each of the mesh elements of the shooting surface, to all of the mesh elements of the receiving surface. This results in a new energy distribution on the receiving surface.

In contrast to this classical radiosity implementation, hierarchical radiosity uses a hierarchy of meshes to describe the energy distributions at each surface at a variety of resolutions. For different portions of the energy propagation, different resolutions of the meshes are used. This novelty combines two different meshing strategies, multilevel meshing of surfaces receiving energy, and multilevel meshing of surfaces shooting energy. Each of these strategies have been used and justified in previous implementations of radiosity.

Multilevel receivers: It is clear from the linearity of the integral operator, that one can shoot from each the surface elements independently. Therefore one is free to use a different mesh resolution over the receivers for different shooting element. For some shooting elements, one may need to use a fine mesh for the receiving surface, while for other shooting elements one may only need some coarse mesh for the receiver. After all of the shooting elements have been accounted for, the received energy can then be summed up at finest resolution mesh of the receiver.

Multilevel shooters: Cohen et al. [22] discuss the converse strategy. In their method, one always uses the finest resolution mesh to describe the receiving surface, but one uses a coarser description to represent the energy distribution of the shooting surface. They argue that during light exchange, little error is incurred by using this coarse description for the shooters. The coarser distribution of energy at the shooting surface, is obtained by averaging the fine level description of energy.

Hierarchical radiosity combines these two strategies in a systematic way. Roughly speaking, in hierarchical radiosity, when two nearby surface regions of the environment interact, a fine description of the environment is used on both the shooter and the receiver. When two distant regions (relative to the size of the regions) interact,

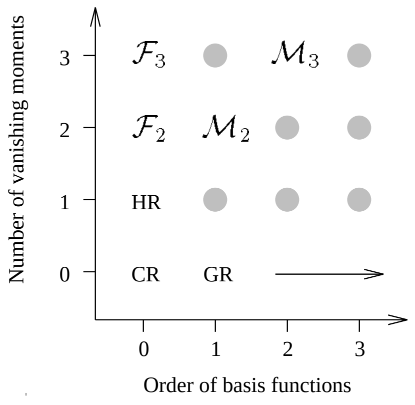


Figure 15: The space of projection methods for radiosity (from [40]).

the kernel function can be well represented as a constant (because the geometric terms, such as the cosines and distance do not vary greatly) and therefore a coarser description is used on both the shooter and receiver. When two regions are partially visible (mutually penumbral), then it is concluded that the kernel function is not constant, and a finer description is used. The hierarchical description of the energy distribution on shooting surface is obtained using an averaging pyramid procedure called `pull` that works from the finest resolution to the coarsest. The radiosity received at the various resolutions of the mesh are summed together with a procedure named `push` that works from the coarsest resolution to the finest [44].

Because many of the interactions can be done between coarse levels of the hierarchy, hierarchical radiosity is able to use interaction patterns with only $O(n)$ interactions, where n is the number of elements in the finest level description of the hierarchy.

Hierarchical methods for radiosity can be formally understood by using a wavelet basis, where the efficiency is expressed by the sparseness of the matrix, and this sparseness is due to the vanishing moments of the wavelets. In this context, it will be explained how hierarchical radiosity is related the Haar basis. By doing so, hierarchical radiosity can be understood as a first order wavelet method. Figure 15 places earlier algorithms plus the wavelet methods into a matrix relating hierarchy versus the order of the underlying basis. Classical radiosity (CR) uses zero order polynomials, while galerkin radiosity (GR) uses higher order polynomials (indicated by the arrow). The vertical axis represents the sparseness obtained by exploiting smoothness of some order in the kernel. Hierarchical radiosity (HR) exploits "constant" smoothness in the kernel. Within this context, we recognize hierarchical radiosity as a first order wavelet. Higher order wavelets can be used that result in an even sparser matrix. One such family of higher order wavelets is the multiwavelet family of [3] ($\mathcal{M}_{2,3}$ in Figure 15). We will also introduce a new family of wavelets, which we have dubbed *flatlets* ($\mathcal{F}_{2,3}$ in Figure 15) that require only low order quadrature methods while maintaining most of the benefits of other wavelet sets.

4.3 Compression of Integral Operators

The theoretical basis for the wavelet radiosity method comes from the discovery by Beylkin et al. [8] that a large class of functional operators can be realized in the wavelet basis with a sparse set of matrix entries. In particular they studied Calderon-Zygmund Integral operators $\mathcal{K}$; these are operators with the form

$$(\mathcal{K}B)(s) = \int dt\ K(s,t)B(t) \tag{94}$$

where the kernel function K is singular along the diagonal $K(s,s)$, and is sufficiently smooth away from the diagonal. The required smoothness is defined by

$$\begin{aligned} \mid K(s,t) \mid &\leq \frac{1}{\mid s-t \mid^{d}} \\ \mid \partial_s^M K(s,t) \mid + \mid \partial_t^M K(s,t) \mid &\leq \frac{C_M}{\mid s-t \mid^{d+M}} \end{aligned} \tag{95}$$

for some $M \geq 1$, where d is the dimension of the parameters s and t, and C_M is some constant. One simple example of an operator which meets these conditions is

$$(\mathcal{K}B)(s) = \int dt\ \frac{1}{s-t}B(t) \tag{96}$$

If the kernel function K is expanded in a multivariate wavelet basis with M vanishing moments, and K can be well described by a polynomial of low degree (relative to the number of vanishing moments of the chosen wavelet function) over the support of some particular basis function $\psi_{i,j}(s)\psi_{k,l}(t)$, then the corresponding coefficient will be correspondingly small. This occurs because the integral of a wavelet with enough vanishing moments, against a polynomial of correspondingly low degree, is zero.

In particular Beylkin et al. show that for a Calderon-Zygmund kernel, by dropping the first M terms of the kernel's taylor series, the magnitude of a kernel coefficient with respect to a wavelet with M vanishing moments

$$\int ds \int dt \; K(s,t)\, \psi_{i,j}(s)\psi_{k,l}(t) \tag{97}$$

can be bounded by

$$C_M \left(\frac{\max(I_s, I_t)}{\mathrm{singDist}(\psi_{i,j}, \psi_{k,l})} \right)^{d+M} \tag{98}$$

where *singDist* measures the minimum distance between the singularity in (s,t) and the support of the basis function $\psi_{i,j}(s)\psi_{k,l}(t)$. I_s is the support length of the basis function $\psi_{i,j}(s)$, and I_t is the support length of the basis function $\psi_{k,l}(t)$. For completeness, a derivation of this theorem is given in Appendix B.

This implies that coefficients are small when the support of the basis functions are small, or if the basis functions are far from the singularity of the kernel function [1]. Given that the coefficient of a particular basis function is less than epsilon, if one moves twice as far from the singularity, then the basis functions that are twice as wide will also have coefficients less than epsilon.

Using this reasoning, Beylkin et al. show that for Calderon-Zygmund operators, when the standard bivariate wavelet basis is used to expand the kernel function, only $O(n \log n)$ coefficients are larger than a user defined epsilon. Moreover, only $O(n)$ coefficients are significant if the non-standard bivariate wavelet basis is used. In these complexity bounds, n is the number of basis functions used to expand functions of the single parameter s or t, such as the function B in Equation (94). By discarding the non-significant coefficients, a sparse matrix representation of the integral operator is obtained.

The framework developed by Beylkin et al. is general enough to include the light transfer operator of radiosity problems. For 3D radiosity, the parametric dimension d

[1] If the support size is less than the distance to the singularity, then the coefficients can also be made smaller by raising the number of vanishing moments M.

is two and the denominator of the 3D radiosity kernel falls as $1/r^2$, thus it meets the criteria of Equation (95). Roughly speaking, the radiosity kernel is smooth between two distant, small regions. The kernel has a singularity, and thus is not smooth, where two surfaces touch. The kernel has a discontinuity, and is thus is also not smooth, where two surfaces undergo a visibility change from mutually visible to mutually not visible.

The radiosity kernel for particular geometric configurations can be analyzed in this framework. The radiosity configuration giving rise to an operator analogous to Equation (96) is the environment consisting of two parallel polygons facing each other with a short distance between them. This case is in fact better behaved (with smaller coefficients) than those of Equation (96) because there is no actual singularity for any values of s and t. (There is a singularity when viewed as a function in complex plane. For a discussion, see the Appendix of [65]). An example of a radiosity configuration with a singular kernel is one where two polygons intersect (such as a wall meeting the ceiling). But this case is also better than Equation (96) because this singularity is smaller in dimension, (it is not along the diagonal of parameter space, but just along a corner). A discontinuity in K is similar to a singularity, in that basis functions that cross the discontinuity have large magnitude, but is better behaved in that coefficients than are arbitrarily close to the discontinuity may still be small.

There is a theoretical disadvantage to the standard decomposition, which comes from the coupling of univariate basis functions of different scales i and k in constructing the basis functions $\psi_{i,j}(s)\psi_{k,l}(t)$. This coupling gives the basis functions wider than necessary supports, resulting in more basis functions crossing a discontinuity, or not being well separated from a singularity. In comparison, the non-standard basis functions have square support, and have more basis functions of smaller support.

Consider for example the Haar basis with $L = 1$. In the bivariate standard basis, there are 4 functions at level 0 with support 4×4, there are 4 basis functions $\{\psi_{1,j}(s)\psi_{0,l}(t), \psi_{1,j}(s)\phi_{0,l}(t)\}$ with support 2×4, there are 4 basis functions $\{\psi_{0,j}(s)\psi_{1,l}(t), \phi_{0,j}(s)\psi_{1,l}(t)\}$ with support 4×2, and there are 4 bivariate basis functions $\{\psi_{1,j}(s)\psi_{1,l}(t)\}$ with support size 2×2.

In the non-standard basis, there are the 4 functions at level 0 with support 4×4, while the remaining 12 basis functions only involve functions at level 1 and all have support 2×2. For the non-standard basis, the arguments in [9] arrive at an $O(n)$ sparseness bound on the number of significant coefficients.

4.4 Wavelet Radiosity

4.4.1 Standard Decomposition

The idea behind the standard wavelet method is simple. All the reasoning of section 3.3 is applied, but instead of using a standard finite element basis to represent $B(s)$, a wavelet basis is used.

In this case, the matrix has elements of the form

$$\begin{aligned}
k^{\alpha}_{ijkl} &= \int ds \int dt\ K(s,t)\tilde{\psi}_{i,j}(s)\psi_{k,l}(t) \\
k^{\beta}_{jkl} &= \int ds \int dt\ K(s,t)\tilde{\phi}_{0,j}(s)\psi_{k,l}(t) \\
k^{\gamma}_{ijl} &= \int ds \int dt\ K(s,t)\tilde{\psi}_{i,j}(s)\phi_{0,l}(t) \\
k^{\phi}_{jl} &= \int ds \int dt\ K(s,t)\tilde{\phi}_{0,j}(s)\phi_{0,l}(t)
\end{aligned} \tag{99}$$

This matrix can be defined relative to the matrix $\mathbf{K}$ of section 3.3. as $\mathbf{WKW^{-1}}$. (Recall that $\mathbf{W}$ is the matrix representation of `coef_pyrm_up`). If one is using the symmetric expansion, then the standard matrix has the terms

$$\begin{aligned}
k^{\alpha}_{ijkl} &= \int ds \int dt\ K(s,t)\psi_{i,j}(s)\psi_{k,l}(t) \\
k^{\beta}_{jkl} &= \int ds \int dt\ K(s,t)\phi_{0,j}(s)\psi_{k,l}(t) \\
k^{\gamma}_{ijl} &= \int ds \int dt\ K(s,t)\psi_{i,j}(s)\phi_{0,l}(t) \\
k^{\phi}_{jl} &= \int ds \int dt\ K(s,t)\phi_{0,j}(s)\phi_{0,l}(t)
\end{aligned} \tag{100}$$

and the matrix can be defined as $\mathbf{W^{-T}\tilde{K}W^{-1}}$.

This decomposition is referred to as standard, as it can be interpreted as an application of Equation (90) where $K(s,t)$ is projected and expressed with respect to the bivariate standard wavelet basis. Here is the symmetric version of that expansion:

$$\begin{aligned}\hat{K}(s,t) &= \sum_{ijkl} k^{\alpha}_{ijkl}\tilde{\psi}_{i,j}(s)\tilde{\psi}_{k,l}(t)\\ &+ \sum_{jkl} k^{\beta}_{jkl}\tilde{\phi}_{0,j}(s)\tilde{\psi}_{k,l}(t)\\ &+ \sum_{ijl} k^{\gamma}_{ijl}\tilde{\psi}_{i,j}(s)\tilde{\phi}_{0,l}(t)\\ &+ \sum_{jl} k^{\phi}_{jl}\tilde{\phi}_{0,j}(s)\tilde{\phi}_{0,l}(t) \qquad (101)\end{aligned}$$

One can also express this decomposition, as the operator $\mathcal{K}$ operating on the finite dimensional function spaces and using the identity $P_L = P_0 + \sum_{i=0}^{L-1} Q_i$. (Here is the symmetric version)

$$\begin{aligned}&\tilde{P}_L\mathcal{K}P_L\\ &= (\tilde{P}_0 + \sum_{i=0}^{L-1}\tilde{Q}_i)\mathcal{K}(P_0 + \sum_{i=0}^{L-1} Q_i)\\ &= \tilde{P}_0\mathcal{K}P_0 + \sum_{i=0}^{L-1}\tilde{P}_0\mathcal{K}Q_i + \sum_{i=0}^{L-1}\tilde{Q}_i\mathcal{K}P_0 + \sum_{i,l=0}^{L-1}\tilde{Q}_i\mathcal{K}Q_l\end{aligned} \qquad (102)$$

Each one of the terms in the above expansion describes the action of the integral operator between the two spaces on its left and right side [3].

4.4.2 Non-Standard Decomposition

A sparser set of coefficients representing the kernel can be obtained using the non-standard wavelet decomposition. This representation must be used to obtain a method to implement the integral operator $\mathcal{K}$. This can be done using the method described in [9], which may be described from three different points of view.

Algebraic

In Equation (101), $\hat{K}$ is expressed with respect to the standard bivariate wavelet basis. As explained in Section 2.6.2, an alternative bivariate basis is the non-standard basis, and $\hat{K}$ can be expressed as

$$\begin{aligned}\hat{K}(s,t) = &\sum_{ijk} k^{\alpha}_{ijk}\tilde{\psi}_{i,j}(s)\tilde{\psi}_{i,k}(t) + \sum_{ijk} k^{\beta}_{ijk}\tilde{\phi}_{i,j}(s)\tilde{\psi}_{i,k}(t) \\ &+ \sum_{ijk} k^{\gamma}_{ijk}\tilde{\psi}_{i,j}(s)\tilde{\phi}_{i,k}(t) + \sum_{jk} k^{\phi}_{jk}\tilde{\phi}_{0,j}(s)\tilde{\phi}_{0,k}(t) \qquad (103)\end{aligned}$$

where

$$\begin{aligned} k^{\alpha}_{ijk} &= \int ds \int dt\ K(s,t)\psi_{i,j}(s)\psi_{i,k}(t) \\ k^{\beta}_{ijk} &= \int ds \int dt\ K(s,t)\phi_{i,j}(s)\psi_{i,k}(t) \\ k^{\gamma}_{ijk} &= \int ds \int dt\ K(s,t)\psi_{i,j}(s)\phi_{i,k}(t) \\ k^{\phi}_{jk} &= \int ds \int dt\ K(s,t)\phi_{0,j}(s)\phi_{0,k}(t) \qquad (104)\end{aligned}$$

In this expansion, all bivariate basis functions are combined from univariate basis functions of the same scale i. Because all of the bivariate basis functions have square support, the well-separated arguments of [9], show that only $O(n)$ k terms will be above ϵ.

The only question remaining is how does one use this expansion of the kernel function to expand the integral operator. To expand the operator, it is best to begin with Equation (90) and not Equation (91). From this we can derive:

$$\begin{aligned}&\int dt\hat{K}(s,t)B(t) \\ &= \sum_{ij}\tilde{\psi}_{i,j}(s)\sum_{k} k^{\alpha}_{ijk}b^{\alpha}_{ik} + k^{\gamma}_{ijk}b^{\gamma}_{ik} + \sum_{ij}\tilde{\phi}_{i,j}(s)\sum_{k} k^{\beta}_{ijk}b^{\beta}_{ik} + \sum_{j}\tilde{\phi}_{0,j}(s)\sum_{k} k^{\phi}_{jk}b^{\gamma}_{0,k} \\ &\qquad (105)\end{aligned}$$

where

$$\begin{aligned} b^{\alpha}_{ik} = b^{\beta}_{ik} = b_{\psi_{i,k}} &= \int dt\ \tilde{\psi}_{i,k}(t)B(t) \\ b^{\gamma}_{ik} = b_{\phi_{i,k}} &= \int dt\ \tilde{\phi}_{i,k}(t)B(t) \qquad (106)\end{aligned}$$

Proof:

$$\begin{aligned}
&\int dt\hat{K}(s,t)B(t) \\
&= \int dt \sum_{ijk} k^{\alpha}_{ijk}\tilde{\psi}_{i,j}(s)\tilde{\psi}_{i,k}(t)B(t) + \int dt \sum_{ijk} k^{\beta}_{ijk}\tilde{\phi}_{i,j}(s)\tilde{\psi}_{i,k}(t)B(t) \\
&\quad + \int dt \sum_{ijk} k^{\gamma}_{ijk}\tilde{\psi}_{i,j}(s)\tilde{\phi}_{i,k}(t)B(t) + \int dt \sum_{jk} k^{\phi}_{jk}\tilde{\phi}_{0,j}(s)\tilde{\phi}_{0,k}(t)B(t) \\
&= \sum_{ij}\tilde{\psi}_{i,j}(s)\int dt \sum_{k} k^{\alpha}_{ijk}\tilde{\psi}_{i,k}(t)B(t) + \sum_{ij}\tilde{\phi}_{i,j}(s)\int dt \sum_{k} k^{\beta}_{ijk}\tilde{\psi}_{i,k}(t)B(t) \\
&\quad + \sum_{ij}\tilde{\psi}_{i,j}(s)\int dt \sum_{k} k^{\gamma}_{ijk}\tilde{\phi}_{i,k}(t)B(t) + \sum_{j}\tilde{\phi}_{0,j}(s)\int dt \sum_{k} k^{\phi}_{jk}\tilde{\phi}_{0,k}(t)B(t) \\
&= \sum_{ij}\tilde{\psi}_{i,j}(s)\sum_{k} k^{\alpha}_{ijk}\int dt\tilde{\psi}_{i,k}(t)B(t) + \sum_{ij}\tilde{\phi}_{i,j}(s)\sum_{k} k^{\beta}_{ijk}\int dt\tilde{\psi}_{i,k}(t)B(t) \\
&\quad + \sum_{ij}\tilde{\psi}_{i,j}(s)\sum_{k} k^{\gamma}_{ijk}\int dt\tilde{\phi}_{i,k}(t)B(t) + \sum_{j}\tilde{\phi}_{0,j}(s)\sum_{k} k^{\phi}_{jk}\int dt\tilde{\phi}_{0,k}(t)B(t) \\
&= \sum_{ij}\tilde{\psi}_{i,j}(s)\sum_{k} k^{\alpha}_{ijk}b^{\alpha}_{ik} + k^{\gamma}_{ijk}b^{\gamma}_{ik} + \sum_{ij}\tilde{\phi}_{i,j}(s)\sum_{k} k^{\beta}_{ijk}b^{\beta}_{ik} + \sum_{j}\tilde{\phi}_{0,j}(s)\sum_{k} k^{\phi}_{jk}b^{\gamma}_{0,k}
\end{aligned} \tag{107}$$

□

Equation (105) can be implemented in the following three phase algorithm:
Pull: In this phase, the $b^{\alpha\beta\gamma}$ scalars are computed. If we begin with the $b_{\phi_{L,i}}$ coefficients, the new scalars can be obtained with the procedure `pull`. This procedure is just like `coef_pyrm_up(` bs_{in}`[] ,` bs_{out}`[] ,` w_{out}`[][] ,` L `)` except that all of the $b_{\phi_{i,j}}$ coefficients are saved on the way up, and output after the pyramid.

```
pull( $bs_{\text{in}}$[] , $bs_{\text{out}}$[][] , $w_{\text{out}}$[][] , $L$ )
    $bs_{\text{temp}}[L][] = bs_{\text{in}}[L]$ ;
    for( $i = L$; $i \geq 1$; $i --$ )
        coef_xform_up( $bs_{\text{temp}}[i][]$, $bs_{\text{temp}}[i-1][]$, $w_{\text{out}}[i-1][]$, $i$ ) ;
        $bs_{\text{out}}[i-1][] = bs_{\text{temp}}[i-1][]$ ;
```

Gather: In this phase index k of Equation (105) is summed over for all i and j. This can be thought of as multiplying a matrix with a vector.

Push: At this point, the resulting function is expressed as a combination of $\tilde{\phi}_{i,j}$ and $\tilde{\psi}_{i,j}$. To obtain an expression of this function with respect to the basis $\tilde{\phi}_{L,j}$, the procedure `dual_push` can be used. This procedure is like the dual `coef_pyrm_down(` bs_{in}`[],` w_{in}`[][] ,` bs_{out}`[],` L `)` except that the after each call to the dual `coef_xform_down`, the returned vector $b_{\text{temp}}[i][]$ has $b_{\text{in}}[i][]$ added to it.

```
dual_push( $bs_{\text{in}}$[][], $w_{\text{in}}$[][] , $bs_{\text{out}}$[], $L$ )
    $bs_{\text{temp}}[0][] = bs_{\text{in}}[0][]$ ;
    for( $i = 1$; $i \leq L$; $i$++ )
        dual_coef_xform_down( $bs_{\text{temp}}[i-1][]$, $w_{\text{in}}[i-1][]$, $bs_{\text{temp}}[i][]$, $i$ ) ;
        $bs_{\text{temp}}[i][]$ += $bs_{\text{in}}[i][]$ ;
    $bs_{\text{out}}[L] = bs_{\text{temp}}[L][]$ ;
```

Projections

The non-standard decomposition of $\mathcal{K}$ can also be described completely as the following operator decomposition.

$$\tilde{P}_L \mathcal{K} P_L = \tilde{P}_0 \mathcal{K} P_0 + \sum_{i=0}^{L-1} \tilde{Q}_i \mathcal{K} P_i + \sum_{i=0}^{L-1} \tilde{P}_i \mathcal{K} Q_i + \sum_{i=0}^{L-1} \tilde{Q}_i \mathcal{K} Q_i \tag{108}$$

This expression states nothing about expanding the kernel function K, it merely describes the operator as the sum of "smaller" operators. And given bases to span the V_i and W_i, each of these smaller operators can be expressed as a matrix. This gives rise to the same three phase algorithm.

This decomposition can be derived using the following reasoning. Suppose one begins with the coarsest description of the operator

$$(\tilde{P}_0 + \tilde{Q}_0)\mathcal{K}(P_0 + Q_0) \tag{109}$$

Suppose one now wishes to include yet another level of detail to obtain

$$(\tilde{P}_0 + \tilde{Q}_0 + \tilde{Q}_1)\mathcal{K}(P_0 + Q_0 + Q_1) \tag{110}$$

There are two ways to do this. One could add to the terms of Expression (109) the following terms:

$$\tilde{Q}_1 \mathcal{K} P_0 + \tilde{Q}_1 \mathcal{K} Q_0 + \tilde{Q}_1 \mathcal{K} Q_1 + \tilde{P}_0 \mathcal{K} Q_1 + \tilde{Q}_0 \mathcal{K} Q_1 \tag{111}$$

If this form is used up to include the full resolution finite dimensional function space, then one obtains the standard operator decomposition (Equation (102)). On the other hand, due to $P_0 + Q_0 = P_1$ and could instead add in to the terms of the following terms: Expression (109).

$$\tilde{Q}_1 \mathcal{K} P_1 + \tilde{Q}_1 \mathcal{K} Q_1 + \tilde{P}_1 \mathcal{K} Q_1 \tag{112}$$

If this form is used to include the full resolution finite dimensional function space, then one obtains the non-standard operator decomposition (Equation (108)).

Preconditioning

The standard decomposition can also be viewed as a matrix preconditioning process wherein the vector matrix multiply $\tilde{\mathbf{K}}\mathbf{b}$ is expanded as

$$\tilde{\mathbf{K}}\mathbf{b} = \mathbf{W}^{\mathbf{T}}\mathbf{M}\mathbf{W}\mathbf{b} \tag{113}$$

and where $\mathbf{M} = \mathbf{W}^{-\mathbf{T}}\tilde{\mathbf{K}}\mathbf{W}^{-\mathbf{1}}$ is the preconditioned sparse matrix.

Similarly, the non-standard method can also be viewed as a matrix preconditioning process, except that *rectangular* matrices must be employed. In this case $\mathbf{W}$ is the rectangular matrix that represents the `pull` procedure, and $\mathbf{W}^{\mathbf{T}}$ is the matrix representing `push`. Because the $\mathbf{W}$ matrix is rectangular, and has no unique inverse, there are many matrices $\mathbf{M}$ that represent the same operator. In particular the matrix with the non-standard k terms (Equation (104)) is one such inverse.

4.5 Implementation

This section discusses the implementation of a wavelet radiosity system that was implemented by Peter Schröder and this author, and was reported in [40]. This section discusses some of the issues that arose in implementing that system, and the choices that were made.

4.5.1 Choice of Basis

The 3D implementation of wavelet radiosity, includes only the non-standard decomposition due to the theoretical properties of this basis, and due to some implementation issues. In particular, the arguments in [9] show an $O(n)$ sparseness for the non-standard decomposition, as compared to $O(n \lg n)$ sparseness for the standard decomposition. In addition, our implementation of wavelet radiosity was built on top of an already existing hierarchical radiosity program[44], and the non-standard basis was a more natural extension. In the standard decomposition, each basis function interacts with basis functions from all resolution levels, while in the non-standard decomposition, only basis functions from the same resolution level interact. This is similar to the hierarchical radiosity implementation where an emitter/receiver pair of polygons are recursively subdivided until the desired error tolerance is achieved, and then interactions are computed at that single resolution level.

With regard to the actual type of wavelet used, there are an infinite variety of wavelet bases to choose from with various desirable properties.

- *Small support* is important. Wider basis functions are more likely to cover a non smooth portion of the kernel.
- *Vanishing moments* are important to capture the coherence of energy transfer operator with few significant coefficients leaving a sparser matrix.
- *Continuity* is convenient if the basis is used to represent continuous functions. If a discontinuous basis functions are used, the projected function is likely to display discontinuities.
- *Orthogonality* is convenient in that $\phi = \tilde{\phi}$ and $\psi = \tilde{\psi}$. Non-orthogonality raises many complicated issues. If the non-symmetric expansion (Equation (86)) is used, then one has to keep track of two kinds of basis functions under the integrals. If the symmetric expansion (Equation (92)) is used then one must perform a basis change between iterations, going from the dual to primal representation. If the symmetric expansion is used, but the primals and duals do not span the same space, then the projection of Equation (92) is not orthogonal,

and one does not obtain a true Galerkin method. (One obtains a more general weighted residual method [23].)

- The *tree property* of the Haar basis is convenient for implementation. In tree wavelets, during the construction of the two-part basis, disjoint sets of scaling functions are combined to construct disjoint sets of wavelet and wider scaling functions. In the Haar case, two adjacent box functions are combined to make one wider box function, and one Haar function. The B-spline wavelets do not have this tree property. In the B-spline example, five B-splines are combined to make one wider B-spline. Some of these five are also needed to make the adjacent translated wider B-spline.

In the implementation described in this thesis, only tree wavelets were considered because the wavelet radiosity program was written as an extension to an already existent hierarchical radiosity program [2]. Two families of bases were considered, the multiwavelets that were developed by Alpert [3], and the *flatlets* which were developed specifically for the radiosity problem. To maintain the tree property, while increasing the number of vanishing moments, one must surrender the wavelet property of having the construction based on a single ϕ and a single ψ. Instead multiple ϕ^m and ψ^m shapes are used.

Multiwavelets

The M^{th} multiwavelet basis, $\mathcal{M}_M$, is a generalization of the Haar basis with M vanishing moments, that was developed by Alpert [3] to quickly approximate integral operators. Like the Haar basis, the domain is divided up into a set of elements, and the represented function is allowed to be any polynomial of order M over each of the elements. There are no continuity constraints between the elements. These functions can be represented by a set of orthogonal Legendre Polynomials over each element.

[2]With tree wavelets, one can replace the scaling functions and wavelet functions over a single element with the scaling functions over two elements on the next finer resolution level, without affecting any basis functions over other elements. If this is done, the interaction matrix terms only deal with scaling functions and not wavelet functions. This scheme corresponds closely to hierarchical radiosity where only constant polygonal elements (box basis functions) interact.

To construct the "two-part" $\mathcal{M}_2$ basis, the $2M$ Legendre basis functions over two adjacent elements are replaced with a different set of $2M$ basis functions: M Legendre Polynomials that are twice as wide, and M piecewise polynomials that have M vanishing moments and are mutually orthogonal. This transformation is applied pairwise across the whole basis to construct the two-part basis. This basis is orthonormal.

Figure 16 shows this `basis_xform_up` transformation over two elements in the $\mathcal{M}_2$ basis. The transformation shown in the figure can be expressed as

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 2 & 0 & 2 & 0 \\ -\sqrt{3} & 1 & \sqrt{3} & 1 \\ 0 & -2 & 0 & 2 \\ 1 & \sqrt{3} & -1 & \sqrt{3} \end{bmatrix}\begin{bmatrix} \phi^1_{L,2j} \\ \phi^2_{L,2j} \\ \phi^1_{L,2j+1} \\ \phi^2_{L,2j+1} \end{bmatrix} = \begin{bmatrix} \phi^1_{L-1,j} \\ \phi^2_{L-1,j} \\ \psi^1_{L-1,j} \\ \psi^2_{L-1,j} \end{bmatrix} \qquad (114)$$

As this is an orthogonal transformation, the same matrix can be used to describe the `coef_xform_up` relationship. The transpose matrix can be used to describe the `basis_xform_down` and `coef_xform_down` relationship. Like in a wavelet construction, the L level hierarchy is then constructed by reapplying this transformation L times each time on the twice as wide Legendre Polynomials.

Computation with a multiwavelet basis is best done with a binary tree. Each leaf represents an element on the finest level, and thus stores the M Legendre coefficients. To perform `coef_xform_up`, the coefficients from two leaves are transformed using a linear combination (Equation (114) in the $\mathcal{M}_2$ case). The computed ψ coefficients are saved, and the ϕ Legendre coefficients are exposed to the next level of the binary tree. Similarly, one may compute with the bivariate non-standard multiwavelet construction using a quad-tree instead of a binary tree.

The functions spanned by the the $\mathcal{M}_2$ basis are piecewise linear over each element with no continuity between element boundaries. In contrast to this basis, the hat basis (first order B-splines) generates piecewise linear functions with C^0 connections between the elements. This lack of continuity in the multiwavelets is most likely a liability, since the number of basis functions must double to express this extra

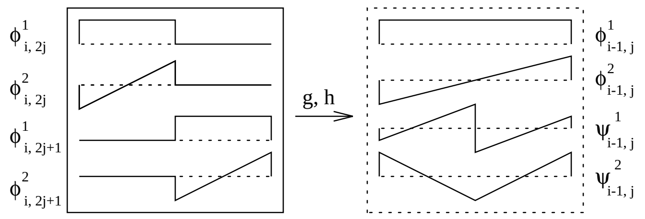


Figure 16: The $\mathcal{M}_2$ wavelet construction whose smooth shapes are the first two Legendre polynomials. Both of the detail shapes (lower right) have two vanishing moments (from [40]).

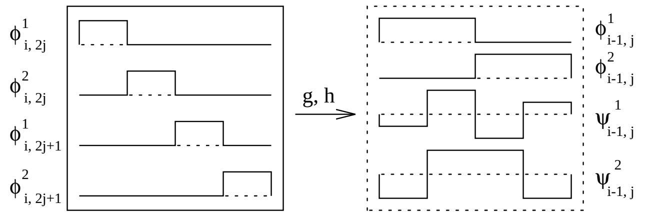


Figure 17: The $\mathcal{F}_2$ wavelet construction. $\mathcal{F}_2$ bases have two different detail shapes. Both of the detail shapes have two vanishing moments (from [40]).

freedom, and the freedom to be discontinuous gives rise to solutions that display unwanted discontinuities. However this is the price that must be paid if one is to use a tree-wavelet.

Flatlets

The *flatlets* are a family of basis functions that were developed specifically for radiosity problems. Flatlets with any desired number of vanishing moments can be constructed to obtain a sparse radiosity matrix. Flatlets are also tree wavelets and so the computation may be performed using a binary tree. Unlike the multiwavelets, the flatlet functions are piecewise constant. This may be desirable in that the associated $k_{i,j}$ integral terms are essentially form factors (Equation (72)). This allows the use of previously developed algorithms and programs for calculating form factors. The flatlet basis combines the piecewise constant functions so that the ψ have higher vanishing moments. To illustrate these principles we give the hierarchical basis with two vanishing moments ($\mathcal{F}_2$) (see Figure 17 and show how to construct bases with even more vanishing moments.

Begin with ϕ^1 and ϕ^2 which are adjacent box functions and define the hierarchy of basis functions using the `basis_xform_up` defined by

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 3 & -3 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \phi^1_{i,2j} \\ \phi^2_{i,2j} \\ \phi^1_{i,2j+1} \\ \phi^2_{i,2j+1} \end{bmatrix} = \begin{bmatrix} \phi^1_{i-1,j} \\ \phi^2_{i-1,j} \\ \psi^1_{i-1,j} \\ \psi^2_{i-1,j} \end{bmatrix} \tag{115}$$

The top two rows of this matrix are chosen to create box functions twice as wide while the bottom two rows are orthogonal to constant and linear variations (the vectors $[1, 1, 1, 1]$, $[0, 1, 2, 3]$). The inverse transpose matrix defines the dual `basis_xform_up`

$$\frac{1}{8} \begin{bmatrix} 5 & 3 & 1 & -1 \\ -1 & 1 & 3 & 5 \\ -1 & 1 & -1 & 1 \\ -2 & 2 & 2 & -2 \end{bmatrix} \begin{bmatrix} \tilde{\phi}^1_{i,2j} \\ \tilde{\phi}^2_{i,2j} \\ \tilde{\phi}^1_{i,2j+1} \\ \tilde{\phi}^2_{i,2j+1} \end{bmatrix} = \begin{bmatrix} \tilde{\phi}^1_{i-1,j} \\ \tilde{\phi}^2_{i-1,j} \\ \tilde{\psi}^1_{i-1,j} \\ \tilde{\psi}^2_{i-1,j} \end{bmatrix}$$

These two matrices define all of the (dual/primal) (basis/coefficient) `xform` procedures.

An L level hierarchical basis is applied by cascading this transformation L times each time on the twice as wide box functions.

There is a choice to be made for the $\tilde{\phi}^m_{L,j}$ functions. One could choose the natural duals (which in this case would be linear functions), and then the $\tilde{\phi}$ would look the same on every level. In this case though $V_L \neq \tilde{V}_L$, and we could not use them to perform the orthogonal projection required for the Galerkin method (Equation (82)). For this reason, at the level L, we define the dual basis to also be the box basis $\phi^m_{L,j} = \tilde{\phi}^m_{L,j}$ [3].

[3]This is also convenient because when the symmetric expansion of the operator is used, the result is expressed in the dual wavelet basis. But after a `push` operation, the result is expressed at level L, where the dual and primal coefficients are the same.

If a linear function is projected into the dual basis, many coefficients will be near zero due to the two vanishing moments of the wavelet functions. The representation of linear variation is possible since the dual functions are made up of ramp like functions that are piecewise constant only on the finest level.

This construction can be generalized for any number of vanishing moments [65]. To construct the two-scale relationship for $\mathcal{F}_3$ we would use a matrix, like that of Equation (115), where the first 3 rows are chosen to give us box functions of width 2 and the bottom 3 rows are chosen to be orthogonal to constant, linear, and quadratic variation. This implies that the last three rows will be a basis for the null space of

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 4 & 9 & 16 & 25 \end{bmatrix}$$

Clearly there are many bases for this space each giving us a set of wavelets and associated two-scale relationship. To fix these degrees of freedom we give the wavelets even more vanishing moments. We require the first wavelet to have 5 vanishing moments, giving us the row $(-1, 5, -10, 10, -5, 1)$. The next wavelet is required to have 4 vanishing moments and to be orthogonal to the first wavelet, giving us $(1, -3, 2, 2, -3, 1)$. Finally, the third wavelet is required to have 3 vanishing moments and be orthogonal to the first 2 wavelets, yielding the final row $(-5, 7, 4, -4, -7, 5)$ and the matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ -1 & 5 & -10 & 10 & -5 & 1 \\ 1 & -3 & 2 & 2 & -3 & 1 \\ -5 & 7 & 4 & -4 & -7 & 5 \end{bmatrix}$$

After normalizing each row we have the two-scale relationship for $\mathcal{F}_3$. This procedure is very similar to the one used in [1].

4.5.2 Calculating The Wavelet Matrix

At the heart of the wavelet representation of the radiosity operator is a matrix of k terms defined by Equation (104). Due to the vanishing moments of the wavelets, many of these terms will be negligible and can be ignored, but first one must obtain this matrix.

One possible way of doing this is a bottom up approach. To start, the $k_{\phi_{L,i},\phi_{L,j}}$ terms are computed, using some numerical quadrature. Then the bivariate `nonstan_coef_pyrm_up` is run on this matrix of terms. Finally the negligible entries are discarded, leaving a sparse matrix.

This method is not efficient though, because it first requires computing the n^2 terms of the original matrix, before the sparse matrix is obtained. Computing the matrix terms is the main bottleneck of computation (as opposed to solving the system once the matrix is in hand).

Since there are a small number significant entries in the wavelet matrix (only $O(n)$) if it were known which ones they were, they could be directly computed in the wavelet basis using some quadrature. Unfortunately, this is not the case. This process can be approximated however, by working in a top down fashion. First the $\phi\phi$ coefficients at level 0 are computed. If the kernel is smooth over its entire support, then no more terms need to be computed. Otherwise the wavelet coefficients are computed at that level (in this case 0), and the next level (in this case 1) is investigated. At level 1, if the kernel is not smooth over the support of some of the bivariate basis functions, those coefficients are computed, and the basis functions under the same region of support on the next level are investigated. If, however, the kernel is smooth over the support of some of the bivariate basis functions, then the finer basis functions on the next level, with support under these regions do not need to be investigated. The smoothness is measured by an `Oracle` function which is discussed later. This can be summarized with the following recursive procedure: [4]

[4]Once again, to ease the exposition, the procedures are written out as if the patches were univariate and the kernel bivariate

```
ProjectKernel( i, patch p, patch q )
  smooth = Oracle( p, q );
  if( smooth ) return;
  else
    ($k^{\alpha}_{i,j(p),k(q)}$, $k^{\beta}_{i,j(p),k(q)}$, $k^{\gamma}_{i,j(p),k(q)}$) = Quadrature( k, p, q );
  if( i == L-1 ) return;
  else
    ProjectKernel( i+1, left(p), left(q) );
    ProjectKernel( i+1, left(p), right(q) );
    ProjectKernel( i+1, right(p), left(q) );
    ProjectKernel( i+1, right(p), right(q) );
```

The use of tree-wavelets in this implementation allows for a number of simplifications. Starting at level 0, if the oracle determines that the kernel is not smooth, instead of computing the $\phi\phi$, $\phi\psi$, $\psi\phi$ and $\psi\psi$ coefficients at level 0, we compute the $\phi\phi$ coefficients on the next level down (1). Since this is a tree wavelet, this is an exact n function to n function replacement, and so represents the same operator decomposition (equivalently the same projection of K). Given that we are computing the $\phi\phi$ coefficients at level 1, this level can be treated just like four copies of the problem at level 0. With this, `ProjectKernel` can be implemented as

```
ProjectKernel( i, patch p, patch q )
  ParentLevelsmooth = Oracle( p, q );
  if( ParentLevelsmooth || i == L )
    $k_{\phi,\phi}$ = Quadrature( k, p, q );
  else
    ProjectKernel( i+1, left(p), left(q) );
    ProjectKernel( i+1, left(p), right(q) );
    ProjectKernel( i+1, right(p), left(q) );
    ProjectKernel( i+1, right(p), right(q) );
```

When this technique is being employed, none of the kernel coefficients are computed until the basis functions cover smooth regions. This makes the numerical integration of those coefficients well behaved.

In this light, this implementation of wavelet radiosity is closely related to the hierarchical radiosity algorithm of [44]. In 3D radiosity where B is a bivariate function, the bivariate non-standard flatlet or multiwavelet coefficients are stored in a quad-tree. The flatlet or multiwavelet coefficients are transformed into the box or piecewise Legendre representation using a quad-tree `push` procedure similar to that of [44]. The flatlet or multiwavelet representation is obtained using a quadtree procedure `pull` like that in [44]. The $k_{i,j}$ entries of the sparse matrix are stored on *links* that have pointers to the i and j flatlet or multiwavelet coefficients. The `gather` operation is performed by traversing this set of links.

4.5.3 Oracle

The job of the `Oracle` function is to determine if the kernel function can be represented by a low order polynomial over some region of support. If this is true, the kernel wavelet coefficients over that region on this and finer levels are negligible, and need not be computed. If the oracle is too stringent, then too many coefficients will be computed. If the oracle is too lenient, then too few coefficients will be computed and an unsatisfactory result may be computed. There is clearly a work/accuracy tradeoff here, and so the oracle should be driven by a user defined ϵ.

In hierarchical radiosity, the actual magnitude of the form factor is used for the oracle determination. This can be viewed as a simple way of determining the variation in the kernel. When the form factor is large, this implies that one is dealing with large nearby polygons, over which the kernel is likely not to be constant. When the form factor is small, one is dealing with small distant polygons, over which the kernel is probably constant. It would be nice to derive simple geometric considerations (based on such things as size, distance, and orientation) for the oracle to use in cases besides hierarchical radiosity (higher vanishing moments), but this has not been done to date.

One way of implementing a general `Oracle` is to numerically compute the appropriate kernel wavelet coefficients on the present level and examine their magnitude.

Coefficients with small magnitudes imply that the kernel is smooth in the corresponding region. This implementation uses a somewhat different approach. First the kernel is sampled at one set of points to fit an interpolating multivariate polynomial. Then the kernel is sampled at another set of points to see how accurate the interpolating polynomial is. The accuracy of the fit is used to determine the smoothness of the kernel.

The oracle, as implemented, treats the visibility portion of the kernel separately. A simple visibility test is performed to determine total visibility, invisibility or partial visibility between two surfaces using jittered rays. If the two surface regions, corresponding to this interval of the 4variate kernel function, are mutually invisible, then it is quickly determined that the kernel is smooth (it is identically zero). If the two surface regions are completely, mutually visible, then the sampling test described above is used for the oracle. If the two surface regions are mutually partially visible, then the sampling test described in the previous paragraph is used, and a separate penalty is assessed due to the partial visibility.

Like in hierarchical radiosity, a *brightness refinement* oracle is used in this implementation. This means that the result of the oracle's kernel test is weighted by the brightness of the involved surface regions. More properly one could imagine a *variation refinement* strategy where the magnitude of the radiosity wavelet coefficients is used to weight the oracle's kernel test. In this case large surface regions with low order polynomial radiosity functions would not be subdivided even if the kernel coefficients are large. In the matrix multiply, those kernel wavelet coefficients are multiplied by negligible radiosity wavelet coefficients, and therefore do not matter. This strategy has not been implemented to date [5].

4.5.4 Quadrature

Some form of quadrature must be employed to compute the k matrix terms. The current implementation uses Gauss-Legendre (GL) quadrature. When a Gauss-Legendre

[5]When using a bi-orthogonal wavelet construction, the dual wavelets may not have the same number of vanishing moments as the primal ones. In the basis expansion, the kernel is integrated against primal functions, while the radiosity function is integrated against the duals, and so the same degrees of smoothness may not vanish in the radiosity function.

quadrature with p sample points is chosen, the resulting method returns the exact definite integral for polynomial functions with order up to $2p-1$.

A greater number of sample points is used as the number of vanishing moments M increases. This is because with more vanishing moments, the `Oracle` will consider the kernel smooth even if it is a higher order polynomial, and so a more costly integration (due to the higher order polynomial) needs to be computed. Also, the integrand consists of the kernel multiplied by a basis function, and in the case of multiwavelets, as M increases, the basis functions are piecewise polynomial of higher order, resulting in a more expensive integration [6]. For multiwavelets, an M point Gauss-Legendre is employed rule for the multiwavelet basis with M vanishing moments. For flatlets, a two point rule is used for each of the M piecewise constant sections of the basis functions [7].

When the integration includes a singular region of the kernel (e.g., the form factor between two intersecting polygons), Gauss-Legendre rules tend to produce inaccurate results. For flatlets, the solution has been to use an analytic form factor solution described in [66]. For multiwavelets, the solution has been to have the oracle function subdivide these singular regions until their error is below a given threshold.

4.6 Results

4.6.1 Flatland Radiosity

To verify and illustrate some of the ideas of wavelet radiosity, 2 flatland radiosity [45] (radiosity in 2D) environments were studied: the configuration consisting of two perpendicular unit length line segments meeting in a corner, and the configuration of two unit length parallel line segments that are separated by unit length (Figure 18)

The kernel functions from these two configurations were projected into the bivariate box basis $\phi_{L,i}(s)\phi_{L,j}(t)$. The left column of Figure 19 shows these projections

[6]Special Gauss-Legendre rules can be developed for each basis function, so that its order is considered an "integral weighting function", not included in the complexity of the integrand. This has been attempted in [14].

[7]This choice is justified further in [65].

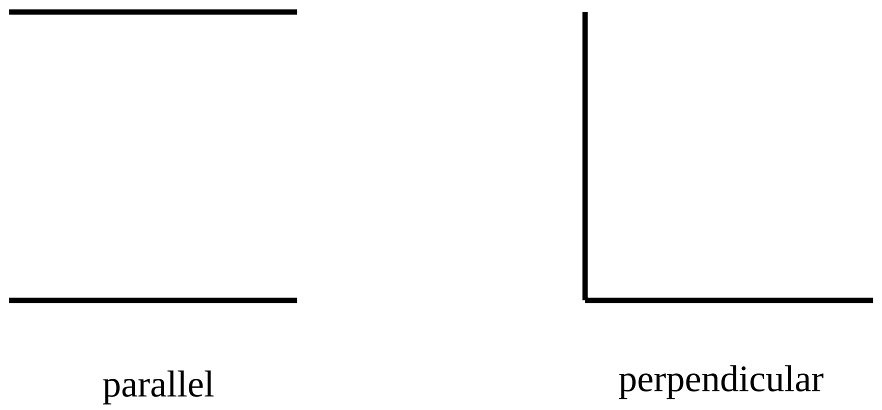


Figure 18: The 2D configurations of two parallel lines, and two perpendicular lines .

with $L = 5$ obtaining a 64×64 element matrix (1/4 of the full symmetric matrix is shown in these figures). In this figure, the size of a dot represents the magnitude of the associated matrix coefficient. In these (1/4) matrices, all n^2 coefficients are non negligible, but the matrices do show some coherence. The `stan_coef_pyrm_up` procedure was then applied to the matrices to obtain the representation with respect to the bivariate standard wavelet basis (the middle and right columns of Figure 19). The `nonstan_coef_pyrm_up` procedure was then applied to obtain the non-standard representation. These procedures were applied using both the Haar and the $\mathcal{F}_2$ bases (Figures 20 and 21). In each of the wavelet representations, many of the coefficients are small in magnitude, and it can also be seen that the basis with two vanishing moments has a greater number of negligible coefficients.

These different representations were also compared numerically. Beginning with a matrix with 256×256 elements, the standard and non-standard pyramid up procedures were applied. The largest m coefficients were kept, and the rest were discarded. To measure the accuracy of this sparse representation, the matrices were then subjected to the inverse pyramids to obtain an approximate kernel $\hat{K}$. The errror due to this sparse representation was computed as

$$\frac{\int dt \int ds \mid K(s,t) - \hat{K}(s,t) \mid}{\int dt \int ds \mid K(s,t) \mid} \tag{116}$$

This measures the accuracy of the kernel representation in general instead of just

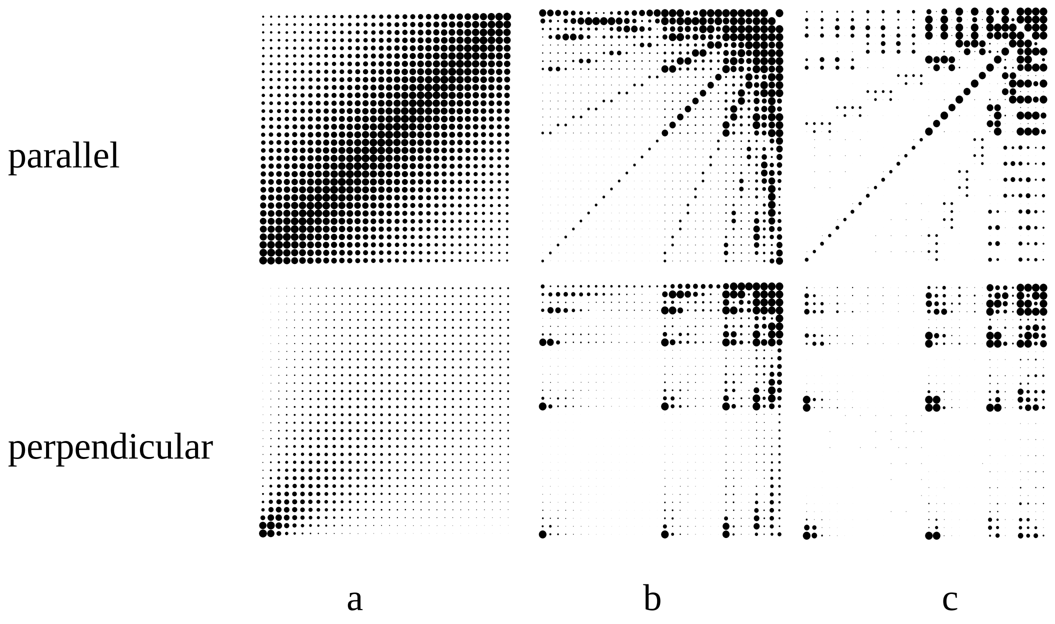


Figure 19: The top row shows 1/4 of the matrix of the flatland kernel for two parallel line segments in the finest level box basis, the standard Haar basis and the standard $\mathcal{F}_2$ basis (left to right). The bottom row shows the matrix of the flatland kernel for two line segments meeting at right angles in the same three bases (from [65]).

measuring the specific error incurred when using some fixed emission function E. This is also different from the $\|\|_2$ matrix norm used in [9]. The $\|\|_2$ matrix norm measures the worst possible error over the space of possible emission functions. This is probably too pessimistic for typical radiosity problems. The error for a series of full matrices with finer and finer resolution $n = 2^0, 2^1, \ldots, 2^8$ was also measured for comparison. No change to a wavelet basis was performed on these matrices.

The results of these numerical experiments are plotted in Figure 22. In both configurations the series of full matrix solutions were found to have the most error per amount of work, although in the parallel configuration the non-standard Haar basis was only better by a constant factor. The $\mathcal{F}_2$ basis performed asymptotically better as seen by the steeper slopes in the graphs. Notice that the steeper curves flatten out towards the end. This occurs because as the number of coefficients kept increases, the 256 by 256 full matrix solution is approached. In effect the finest level of the representation is artificially limited so that there is a common standard

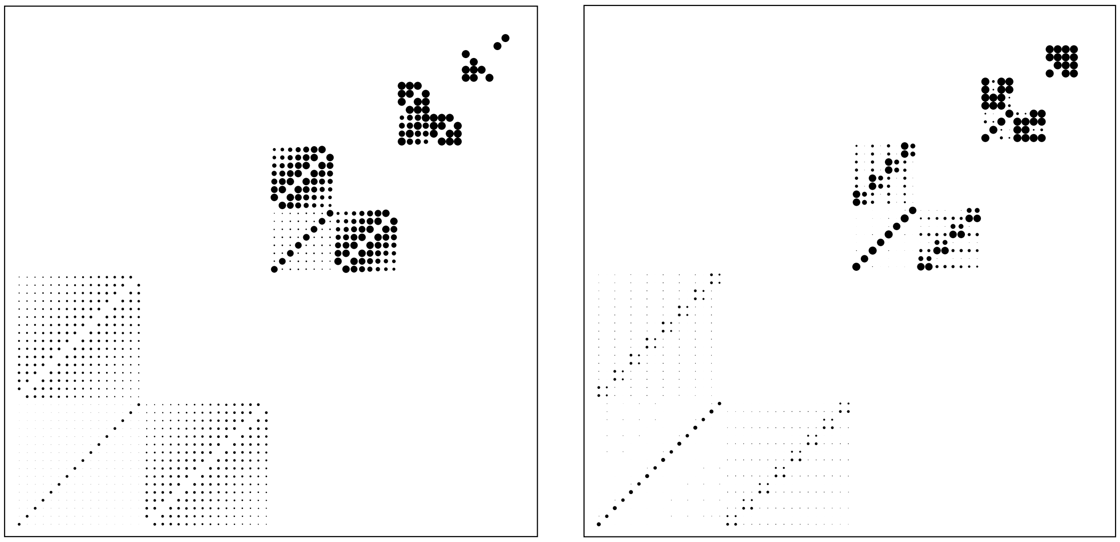


Figure 20: The kernel for two parallel lines realized in the non-standard Haar basis (left) and the non-standard $\mathcal{F}_2$ basis (right) (from [65]).

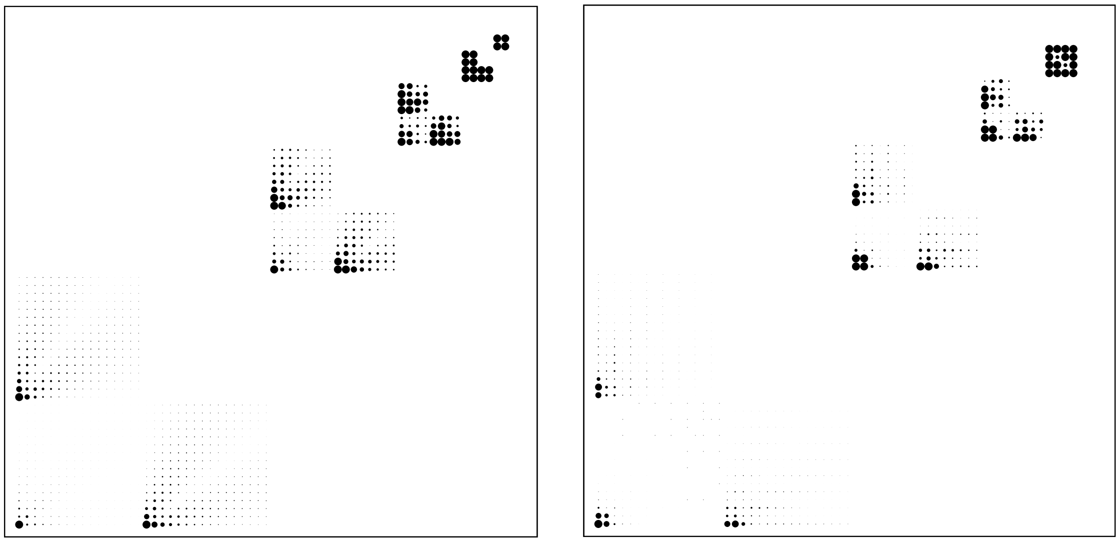
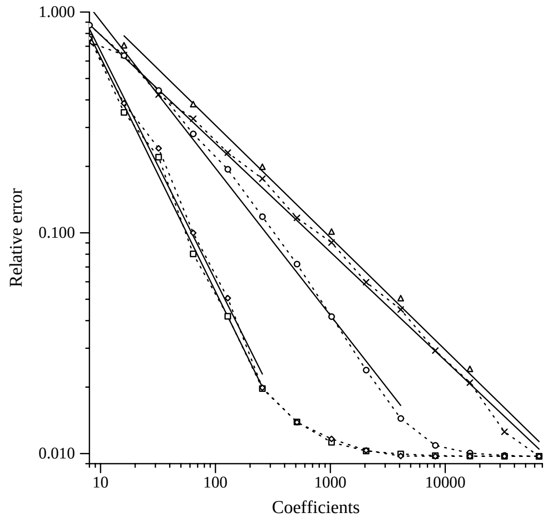


Figure 21: The kernel for two perpendicular lines realized in the non-standard Haar basis (left) and the non-standard $\mathcal{F}_2$ basis (right) (from [65]).

against which to measure the error. In practice, the wavelet radiosity implementation allows the scales to become finer in a dynamic fashion as higher accuracy is requested, preventing the "bottoming out" of the graphs.

Although the theoretical bounds are stronger for the non-standard basis than for



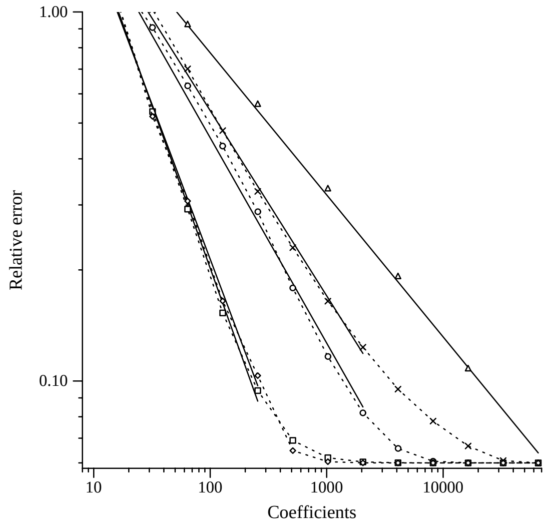


Figure 22: Parallel segments (left) and perpendicular segments (right). Relative L^1 error plotted against number of coefficients used with full matrices, non-standard haar, standard haar, non-standard $\mathcal{F}_2$, and standard $\mathcal{F}_2$ bases (top to bottom) (from [65]).

the standard basis [8], in these experiments the standard basis performed slightly better. This is consistent with the experimental results obtained in [47]. The explanation is that even though the non-standard functions have smaller support, the bivariate standard basis functions are combinations of univariate ψ functions in *both* variables s and t. These standard basis functions have vanishing moments in both s and t which results in a greater number of small coefficients than the non-standard basis functions which combine both univariate ψ and ϕ functions.

In these experiments, it is clear that raising the number M of vanishing moments gives rise to a sparser representation. Of course there are costs involved with raising M. Wavelets with more vanishing moments have wider support (in the case of tree wavelets this corresponds to more basis functions per element) and thus more basis functions can cross over a discontinuity or singularity.

These experimental results suggest that wavelet radiosity can obtain radiosity solutions more efficiently than hierarchical radiosity.

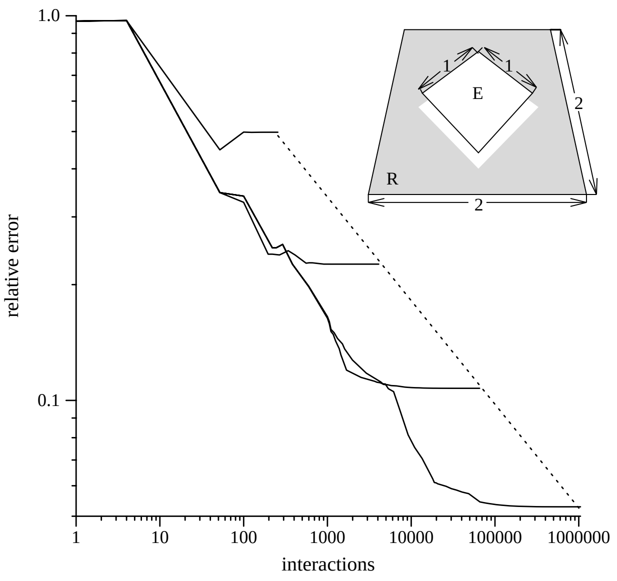


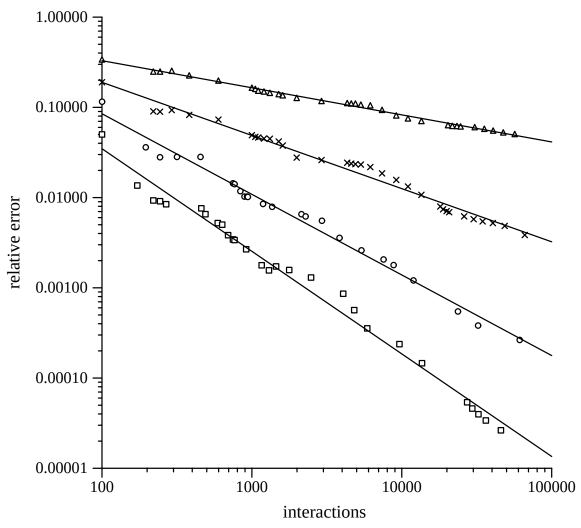
Figure 23: Relative L_1 error as a function of the number of interaction links for the haar basis with $h = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ (top to bottom). The test configuration is depicted in the upper right corner (from [40]).

4.6.2 3D Radiosity

This section discusses the experimental results using the 3D implementation of wavelet radiosity.

The first test configuration, depicted in the inset of Figure 23, consisted of a "shooting" polygon with side length 1, that emits constant radiosity and has zero reflectivity and a "receiving" polygon with side length 2. The distance between these polygons is 0.1. Because the shooter emits constant radiosity, and reflects none, the resulting radiosity at any point on the receiver can be analytically computed using the differential area to finite area form factor [7].

In the wavelet radiosity program, the user sets the stringency of the oracle using an input parameter. When the oracle is made more stringent, the work increases (more matrix terms are computed) as the error in the solution is forced to decrease. Figure 23 shows this work/error tradeoff using the Haar basis. In each of the four curves, a different smallest subdivision size h was imposed. The right end of each curve represents a full matrix of a particular resolution. Note that when using the



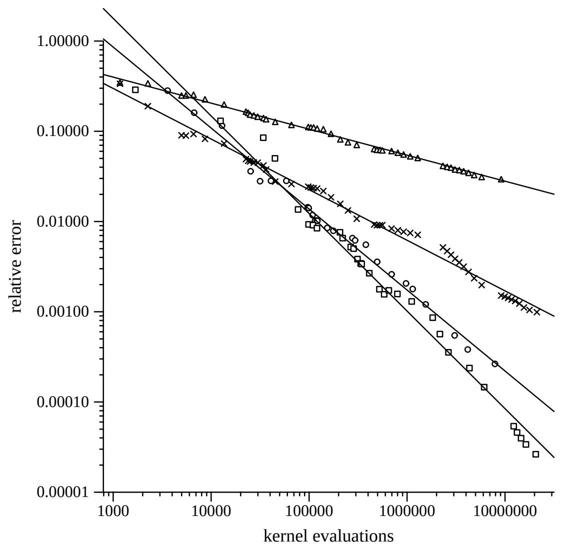


Figure 24: Left: Relative L_1 error as a function of the number of interactions for the wavelet bases $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ (top to bottom) using the same test configuration as in Figure 23. Here $h = \frac{1}{32}$. Right: Relative L_1 error as a function of work (from [40]).

Haar wavelet basis, the full accuracy in the solution is reached well before the full matrix is computed.

The left graph of Figure 24 shows the work/error tradeoff using the first four multiwavelet bases (h was set to $\frac{1}{32}$). As the number of vanishing moments is increased, the basis performs better, and has a steeper convergence slope. The right graph of Figure 24 shows the same experiment, but this time work is measured by the number of kernel samples used for the quadrature. This accounts for the extra overhead used in this implementation with higher order wavelets.

Experiments were also conducted with the configuration of perpendicular polygons (Figure 25) where the kernel has a singularity. Figure 26 shows the exact answer along the receiving floor, as well as error surfaces obtained using the first 3 flatlet bases. In all of these cases the number of interactions is approximately constant (8000). Figure 25 shows rendered images of the configuration using the Haar and $\mathcal{F}_2$ bases.

Figure 27 shows a picture of a more complicated environment, computed using

Figure 25: Computed image of perpendicular emitter and receiver. For the Haar basis (left), and $\mathcal{F}_2$ basis (right) using same amount of work. Note that any post processing such as Gouraud shading has not been performed. From [40].

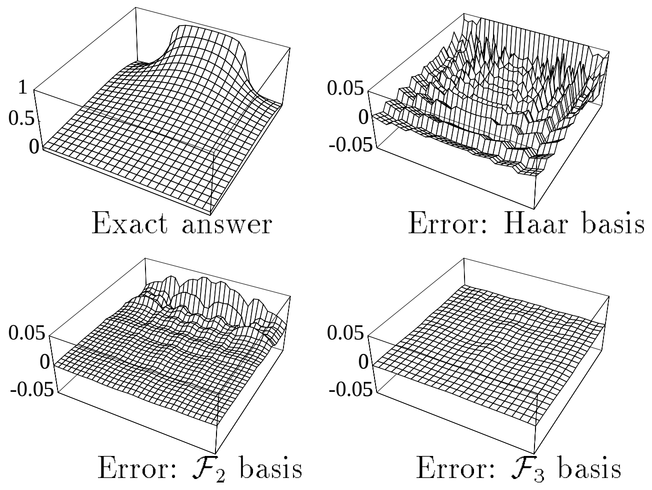


Figure 26: Heightfield error plots for perpendicular emitter and receiver (from [40]).

the $\mathcal{M}_2$ basis [8].

These experiments show how raising the number of vanishing moments allows for a better answer with less work, and thus demonstrates how wavelet radiosity improves over hierarchical radiosity.

[8]These images are displayed with no post processing such as Gouraud shading.

4.7 Conclusion

This section has discussed how wavelets can be used to efficiently solve radiosity problems. Classical radiosity matrices are dense, requiring the computation of n^2 coefficients, and solving a dense linear system. When a wavelet basis used, the associated matrix is sparse. Standard and non-standard wavelet decompositions have been discussed. The non-standard basis functions have smaller support, and so better theoretical bounds are possible, but in the 2D experiments, the standard basis fared better. The 3D system which extends an existent hierarchical radiosity program uses the non-standard basis, since this was the most direct extension.

Two families of wavelets were discussed (flatlets and multiwavelets), and 2D and 3D experimental results were shown. Bases with larger numbers of vanishing moments were shown to obtain better radiosity representations than classical methods and the hierarchical radiosity method. When bases with more than 2 vanishing moments are used, the inter-element discontinuities necessitated by the tree property, give rise to many basis functions per element, and this constant factor of overhead can swamp the computation. For example in the $\mathcal{M}_3$ basis there are 3 basis functions per 1D element, 9 per 2D element, and 81 resulting matrix terms for the interaction of two 2D elements. In practice, because of the high constant factor of using the bases with many vanishing moments, users of the program have had the best experience with the $\mathcal{M}_2$ basis [33]. Clearly it will be fruitful to experiment in 3D with a standard decomposition, using families of continuous wavelets.

The work described in this thesis has been recently been extended to elegantly use texture mapped emittance and reflection patterns [34]. More significantly, there is current research extending this wavelet method to solve the general illumination rendering equation with glossy reflectance [67, 14].

Figure 27: Four views of an architectural scene computed with the $\mathcal{M}_2$ basis and rendered directly from the basis functions (from [40]).

Chapter 5

Geometric Modeling

There are many computer applications where it is necessary for a computer to have an internal representation of some three dimensional geometric object. These objects may include a variety of real world objects, such as a mountain, an airplane or a chair. One may also desire a representation of some imagined object that does not exist in the real world, such as a dragon.

- In *computer graphics*, the representation of a geometric object can be used to render images. Examples of such images include flight simulation images, where the geometric objects may be some terrain, or a control tower. Other examples of such images include artistic images, in which case the geometric object may be anything from a flower to a robot.

- In *computer aided analysis*, the representation of a geometric object may be used by engineers to analyze the object. For example a geometric representation of an airplane wing may be used to finding the weakest point of that wing.

- In *computer aided manufacturing*, this representation may be used to to drive some manufacturing machine. For example, a computer representation of a nut or bolt could drive some sculpting device.

- In *computer aided design*, a designer may use an interactive *geometric modeling tool*, which allows him to manipulate the representation of a geometric model.

> Using such a tool, he may experiment with a virtual object, interactively changing its shape until he finds just the shape he is looking for.

Geometric modeling is the study of how such geometric objects can be represented in a computer. Geometric modeling is also concerned with algorithms and interfaces that allow a user create, manipulate and alter the geometric description of a three dimensional object.

This chapter reviews a variety of representations that have been used in geometric modeling. It also reviews some modeling paradigms that have been used to manipulate curves and surfaces.

5.1 Geometric Representations

In the real world, the shape of some arbitrary three dimensional solid objects can be thought of as binary function S of three variables x, y, z. This binary function denotes whether the solid object covers each spatial location x, y, z. Unfortunately this idealized function, S, cannot be directly used in a computer representation; in order to be used by a computer, the representation must be described by a *finite* set of numbers. This section will briefly describe some of the different representations that are used commonly in geometric modeling.

One way to obtain a finite representation of a geometric solid shape, is to simply *discretize* the ideal binary function S. A 3D spatial region is chopped up into an $N \times N \times N$ cube of *voxels* [46, 55]. A binary value at each voxel is then used to denote whether the object covers that spatial region or not. Using this representation, one can combine different modeled objects using the boolean operations "and" , "or", and "not". One can manipulate an object represented this way using a simple sculpting tool that allows a user to add or scrape material from the object. Although this is a very general representation, it also has some limitations. In particular, unless a very fine grid of voxels is used, the surfaces of the object will have a "blocky" nature. Also it is hard to determine the vector that is normal to some point on the surface of the object. The normal vector is necessary for many shading algorithms.

Another way to represent a solid object is to begin with a collection of simple shapes for which the ideal function S is already known. Such simple solids include spheres, cones and cylinders. Instances of these objects can be parameterized by a finite set of numbers, for example a sphere can be characterized by the location of its center, and its radius. More complicated geometric objects can then be created by combining these building blocks using boolean operations (and, or, not). This representation is known as *constructive solid geometry*. Because these objects can be described mathematically in closed form, one doesn't have to resort to a "blocky" voxel representation. Also surface normals can be computed directly from the mathematical representation. These models can be ray traced, or converted to polygons for display [36, 4]. The main drawback to constructive solid geometry is that one must build all geometric objects by combining a fixed set of simple shapes. At times a user wants to describe an arbitrary or *free-form* shape, and so a more general representation is often required.

For many applications, it is not important to have an actual representation of the 3D solid. Instead one only needs a representation for the surfaces bounding the object. For example, in order to render an image of a solid opaque object, such as a basketball, it is only important to know the shape of its outer surface. It is not important to represent the thickness of the ball's wall, or whether or not its center is hollow. For this reason, geometric modeling often uses a *surface* or *boundary* representation.

There are a variety of different ways to represent a surface. The *implicit* representation defines a surface as the zero set of some density function $F(x, y, z)$. F has positive values in the object's interior, negative values exterior to the object, and zero values at the surfaces bounding the object. For implicit surfaces, one needs a way to represent the density function, F, using a finite set of scalars. The most prevalent method is the *blobby object* method [10, 79], where the density function is defined by placing a collection of "field sources" in space. These field sources have simple density functions, such as a decaying Gaussian function that has the greatest value at the center, and decrease with distance. Instances of these field sources can be parameterized by the location of their centers and their rates of decay. By combining a variety of Gaussian density "blobs", a user can implicitly manipulate the surface (i.e., the zero set). The implicit method is useful for modeling certain types of objects that are "blobby" and have soft rounded features. For example, such methods are useful for modeling the human form. The main drawback of this method is that it does not give a user very precise control over the exact structure of the surface. It may also be expensive to render such a surface; one either has to convert the implicit surface representation to some approximate polygonal representation, or one has to ray trace the implicit surface. Computing the intersection of a ray with an implicit surface requires an expensive root finding operation.

Another way to represent a surface is using some *procedural* method. An example of a procedural representation is *subdivision surfaces* [13, 43]. In this representation, the surface is defined by an input mesh made up of vertices and edges. A transformation rule is applied to the input mesh resulting in a denser mesh. In the limit, as this transformation is applied again and again, the output mesh converges to a set of dense points. These dense points define a continuous surface. A user can manipulate such a surface by manipulating the original mesh. Alternatively, the user can define a set of constraints, such as interpolation constraints, and a satisfactory input mesh can then be solved for. Because this representation begins with some arbitrary mesh, the user is free to model a surface with arbitrary topology. One drawback of the subdivision surface representation is that the surface is defined procedurally, One does not have an explicit representation of the surface. This may make some operations on such a surface, such a ray tracing, difficult.

There are a variety of methods that *explicitly* represent the surfaces of an object. The simplest method is to define the object as a collection of polygons. The user can manipulate the object by moving the vertices of the polygons around in space. Polygonal objects can easily be rendered directly, or can be ray traced. The biggest drawback of polygons is that one cannot use them to model smoothly varying curved surfaces.

To explicitly model smooth curves and surfaces, one can use a *parametric representation*. In this representation, a curve is defined as a 3 dimensional trajectory parameterized by t,

$$\gamma(t) = (X(t), Y(t), Z(t)) \tag{117}$$

and a surface is defined as

$$\gamma(s,t) = (X(s,t), Y(s,t), Z(s,t)) \tag{118}$$

which defines a three dimensional location for every parameter pair (s,t). Using this simple parameterization, one can only construct surfaces that are topologically like a sheet, or torus. More complicated structures must be built by trimming and combining a collection of parameterized sections.

The parametric representation of a curve or surface is made up of three functions X, Y, Z, and so some practical method for representing each of these functions is required. This can be done using some set of basis functions. Just focusing on the X function, for curves this becomes

$$X(t) = \sum_j x_j \phi_{L,j}(t) \tag{119}$$

and for surfaces

$$X(s,t) = \sum_{j,k} x_{j,k} \phi_{L,j,k}(s,t) \tag{120}$$

where the x are scalar coefficients. In geometric modeling the univariate basis $\phi_{L,j}(t)$ is typically some "piecewise" basis, such as a cubic B-spline or the Bernstein (Bézier) basis, and the bivariate basis used for surfaces is the associated tensor product basis $\phi_{L,j,k}(s,t) \equiv \phi_{L,j}(s)\phi_{L,k}(t)$.

There are a variety of cubic B-spline bases that are commonly used in modeling. The uniform cubic B-spline basis is the simplest basis, where all of the basis functions are uniformly spaced translations of a single function. The non-uniform cubic B-spline basis allows for a non-uniform spacing of "knots". In particular, these knots can be coincident allowing for discontinuities of various degrees along a curve. The non-uniform rational B-spline basis (NURB), introduces a polynomial denominator which gives rise to rational functions that allow for expressing a greater variety of curves [6].

Parametric curves and surfaces, represented with combination of basis functions, are used in many popular modeling systems. They can be easily rendered, and are subject to a variety of useful manipulation methods. These parametric curves and surfaces will be the subject of this thesis. In particular, this thesis will discuss using

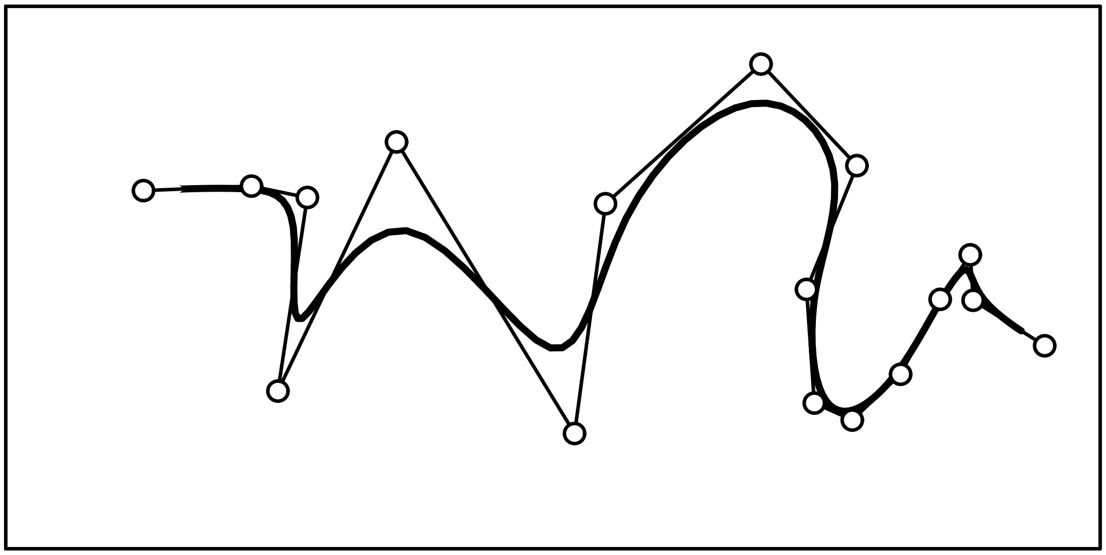


Figure 28: A piecewise cubic curve with defined by a set of B-spline control points (circles).

a wavelet basis based on cubic B-splines to represent smooth parametric curves and surfaces. The use of a wavelet basis allows a greater variety of modeling operations to be performed on the objects and it also allows some modeling operations to be performed more efficiently.

5.2 Geometric Specification

Given some set of basis functions to describe a curve/surface, some specification method is required so that a user can describe and design the desired object. The method must allow the user to manipulate the curve or surface in an intuitive manner, and must operate at interactive rates. This section will review three such methods.

Control Handles

In this simplest of schemes, the user is exposed to the actual representation parameters and allowed to manipulate them. For example, with cubic B-splines, the user is shown some current curve or surface, along with a control polyon or mesh whose vertices are the 3D coefficient points (x_i, y_i, z_i) (See Figure 28). The user can click and drag on these points with a mouse, thus reweighting the associated basis functions and altering the curve or surface. For such a method to work, the basis functions must be chosen so that manipulating their coefficients affects the solution in some intuitive way. Bernstein (and Hermite) basis functions are therefore often used because the resulting curve or surface interpolates the control points (and control tangents). Cubic B-splines are also commonly used because the resulting object is C^2, and the curve is related to the control polyon in some natural ways [6]. These control schemes are desirable because it is computationally inexpensive to calculate and display a curve or surface given the control points.

When controlling a curve or surface using a piecewise basis, the user is forced into manipulating the object at some fixed resolution. This is not fully desirable, because sometimes a user may want to specify some fine detail in a region of the object, while at other times he may wish to describe the broad sweep of it. This thesis will describe how hierarchical representations, such as the wavelet representation can be used to ameliorate this problem.

Least Squares Solving

The control handle method has the disadvantage that the user manipulates a set of parameters instead of the curve or surface itself. In order to affect some desired result, the doggedly determined user may have to try endless combinations of control point changes. It is more desirable to allow the user to directly manipulate the object, grabbing the curve itself at some point and dragging it around, or dragging around the tangent at any curve point. By doing this, the user introduces a *constraint* that the curve or surface must meet. In this context we will only be discussing linear constraints. For example, the constraint that a curve at $t = t_i$ interpolate the point b_i can be written in the form

$$\sum_j x_j \phi_{L,j}(t_i) = b_i \tag{121}$$

and constraining the curve to have derivative b_i can be written

$$\sum_j x_j \dot{\phi}_{L,j}(t_i) = b_i \tag{122}$$

Adding some set of constraints is not usually enough to uniquely describe the curve or surface, (i.e., it is likely to be underconstrained), and so some notion of optimality must be introduced. In *least squares* solving [5], given a current curve or surface defined by some current set of control points, when the user drags on the curve/surface or otherwise defines some constraint, the new curve or surface is found that meets the new constraints, while minimizing the change of the control points. The magnitude of the change is measured by the sum of the squares of the change in each control point. Least square solutions can be found very inexpensively using the pseudoinverse [32] or quadratic minimization techniques [76].

Clearly the semantics of this method is sensitive to the representation used. For example if the uniform cubic B-spline basis is used, the addition of an interpolation constraint has a very local effect. Only four basis functions overlap the parameter t_i, and so a least squares method never tries to move any distant basis functions. This has the disadvantage of locking the user into some fixed resolution. This thesis will describe how a hierarchical wavelet representation may be used to address this problem.

Variational Modeling

The least squares method is a special example of variational modeling. In variational modeling the solution desired is the curve or surface that meets some set of constraints, while having *minimal cost or energy.* There are numerous possible notions of cost, but in geometric modeling optimality is usually measured by smoothness. Given some set of constraints then, the desired solution is the one that satisfies them and has the least "wigglyness".

This notion, "wigglyness" is not well defined, and so there are many mathematical measures employed. One convenient measure is the *thin plate* functional related to the square of the second derivative. Thus, for curves, this is [1]

$$E(\gamma) = \int_{\Gamma} dt \parallel \ddot{\gamma}(t) \parallel^2 \tag{123}$$

and for surfaces is

$$E(\gamma) = \int_{\Gamma} dt \;\; ds \quad \parallel \gamma_{tt}(t,s) \parallel^2 + 2 \parallel \gamma_{ts}(t,s) \parallel^2 + \parallel \gamma_{ss}(t,s) \parallel^2 \tag{124}$$

This measure approximates the physical bending energy of a beam or sheet, that is bent from its original flat shape. Other measures include first derivative terms [73], which penalize stretching, while yet other measures include third derivative terms which penalize variation in curvature[57] (favoring circles, cylinders and spheres). Some measures, like the thin plate term, are sensitive to the particular parameterization of a curve or surface (there are many different parameterized trajectories $(X(t), Y(t), Z(t))$ that trace out the same curve in space). *Intrinsic* geometric measures are based solely on the geometry of the object, and give the same measurement under any s, t parameterization. For curves, the intrinsic equivalent to the thin plate measure method is based on the second derivative of the *arc-length parameterization* κ of the curve. For surfaces, curvature is measured with *gaussian* and *normal* curvature[27]. Other intrinsic functionals include the *minimum variation surface* measure of [57] and the measures given in [62]. All of these functionals can have variable local weights over s and t, allowing the modeling material to be "stiffer" in some regions, and more flexible (and even creasable) in others [73].

Clearly, the intrinsic measures are the most natural ones for a geometric setting, and they produce the most desirable results [57]. The disadvantage of these measures is that the resulting optization problems are highly non-linear and very costly to compute. On the other hand, parametric measures such as the thin plate (even allowing variable local weights) give rise to the much easier quadratic minimization problems. The work reported in this thesis will focus on the simpler thin plate measure, but will also discuss some issues of using more complicated measures.

Given some set of constraints and an energy functional E, the desired solution is the curve or surface that minimizes E while satisfying the constraints. Although

[1]the ℓ_2 norm brackets "$\parallel$" are used since γ is a 3-vector.

this is a well defined problem of variational calculus, it is often difficult to find a closed form solution. For some simple surface problems, such as point and tangent interpolation using the thin plate measure over the entire infinite s, t plane, there are sets of radially symmetric basis functions (where each basis function corresponds to one constraint) which, when properly combined, give rise to the optimal solution [56]. (These methods are the multivariate extension to natural spline optimization methods). While these methods are in many ways optimal, they are somewhat inflexible; special constructions must be done to allow for the thin plate domain to be some finite square of s, t, or to allow for locally variable weights. These methods are not well suited for infinite interpolation problems, such as having the surface interpolate some curve. These methods only work when the problem can be decoupled in to three separate X, Y, Z problems. This is not possible for the intrinsic functionals [2]. Finally, since these radial basis functions are infinite in support, one must solve a dense linear system. For a large number of constraints, this can be prohibitive.

In practice, therefore an approximation method is employed. In particular, this thesis will focus on the *Ritz* method, where a good approximate minimum solution is found by limiting the solutions to those that can be represented by a linear combination of a fixed set of basis function such as cubic B-splines. With this simplification, instead of searching for the minimum of the space of all possible curves or surfaces, the Ritz method only considers solutions that are linear combinations of the fixed set of basis functions. The variational problem then becomes an optimization problem over a discrete set of variables x_i, to which a variety of numerical optimization techniques can be applied.

Given a set of n variable control points $\mathbf{x}$, an objective function $E(\mathbf{x})$, and m constraints $C_i(\mathbf{x}) = 0$, the *Lagrangian* function is defined as

$$L(\mathbf{x}, \lambda) = E(\mathbf{x}) + (\mathbf{C}(\mathbf{x}))^T \lambda \tag{125}$$

Where λ are a set of m Lagrange multipliers.

A necessary condition for some point $\mathbf{x}_0$ to be a constrained minimum of E is that the gradient of L vanishes at $\mathbf{x}_0$, for some set of Lagrange multipliers λ_0. The

[2]It is also not possible for normal interpolation.

gradient with respect to the control points is

$$\nabla_x L(\mathbf{x}, \lambda) = \nabla_x E(\mathbf{x}) + (\mathbf{J}(\mathbf{C}(\mathbf{x})))^T \lambda \tag{126}$$

where $\mathbf{J}$ is the Jacobian matrix. The gradient with respect to the Lagrange multipliers is

$$\nabla_\lambda L(\mathbf{x}, \lambda) = \mathbf{C}(\mathbf{x}) \tag{127}$$

Setting the gradient of the L to 0 creates a set of $n+m$ simultaneous equations, with the $n+m$ variables $(\mathbf{x}, \lambda)$. If the energy functional (like the thin plate term) is a quadratic function

$$E(\mathbf{x}) = \mathbf{x}^{\mathbf{T}}\mathbf{H}\mathbf{x} \tag{128}$$

where $\mathbf{H}$ is the Hessian matrix

$$\mathbf{H}_{i,j} = \frac{\partial^2 E}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \tag{129}$$

and the constraints are linear

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \tag{130}$$

then the optimal solution in the given space can be computed as the solution to a single linear system [76].

$$\begin{vmatrix} \mathbf{H} & \mathbf{A}^{\mathbf{T}} \\ \mathbf{A} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \mathbf{x} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{131}$$

For thin plate curves the Hessian terms are

$$\mathbf{H}_{i,j} = \frac{\partial^2 E}{\partial x_i \partial x_j} = \int dt \;\; \ddot{\phi}_{L,i}(t)\ddot{\phi}_{L,j}(t) \tag{132}$$

If the functional is not quadratic, or the constraints not linear, then a non-linear optimization problem must be solved. Solution techniques such as sequential quadratic programming [35], lead to a series of linear systems with a form like Equation (131).

A common choice of a basis for variational modeling is the uniform cubic B-spline basis. An advantage to this basis is that the linear system obtained in Equation (131) is $O(n)$ sparse; for example the Hessian terms defined in Equation (132) are only non-zero when the support of $\phi_{L,i}$ and $\phi_{L,j}$ overlap. Thus, for cubic B-spline curves there are 7 non-zero entries per row of **H**, and for tensor product cubic B-spline surfaces there are 49 non-zero entries per row. This avoids the need for direct inversion of the matrix in $O(n^3)$ time, as one can use an iterative method such as the conjugate gradient method [59], where each iteration, comprised of a constant number of matrix-vector multiplies, can be performed in linear time. Unfortunately as the number of basis functions in the set grows, the system becomes poorly conditioned resulting in the need for an excessive number of iterations to find a solution [72, 71].

One of the positive qualities of the B-spline basis for modeling paradigms, local control, is a disadvantage in the optimization framework. Since a change in a single coefficient (control point) effects only a very limited section of the curve or surface, there is a conflict between moving a control point to meet a positional constraint and maintaining a smooth curve. This causes the iterative optimization procedures to perform a series of small changes to a sequence of control points. Fewer wider basis functions would provide a better means to control the curve or surface as a whole, but this would limit the ability to model fine detail in certain regions of the curve or surface. What is needed is a hierarchical description of the curve, where some of the basis functions represent a coarse/broad description of the solution, while other basis functions refine the curve and describe a higher degree of detail.

A hierarchical description is also useful for the iterative solution method to be able to dynamically adapt to the complexity of the solution and best approximate the true variational minimum. This adaptivity can be done easily when using a hierarchical description; a few wide (or coarse) basis functions can be used until the solution procedure decides to add detail in some region.

Wavelets offer such a hierarchical basis for representing a curve or surface. Unlike hierarchical B-splines [30], which over-represent a curve or surface by using B-spline functions of different widths, wavelets offer a unique representation of a curve or surface. The wavelet coefficients directly encode the necessary detail at various degrees of resolution, and thus allow solutions of variational problems to proceed adaptively and quickly.

Chapter 6

Multiresolution Modeling with Wavelets

In the following two chapters, three different curve and surface modeling paradigms will be explored. For each of these paradigms, wavelet methods, that make the interaction more intuitive or efficient, will be investigated. This chapter will discuss the control point and the least squares modeling paradigm.

6.1 Contribution

With a *control point* modeling tool, the user is shown some set of *shape handles* which he can drag around to affect the object. In a *least square* modeling tool, the user can directly manipulate the object, using the mouse to specify interpolation and tangent constraints. The modeling tool then returns a new object that meets the constraints by changing the current object by the "least" amount. In both of these paradigms, it will be discussed how the wavelet representation allows a user to manipulate a curve or surface at a variety of resolutions, instead of at some fixed B-spline resolution. This gives the user a more flexible manipulation method.

The wavelet representation allows the user to specify whether local detail changes are desired, or whether broader changes to the entire sweep of the object are intended. These ideas have been pursued independently by Finkelstein et al. [28]. The wavelet

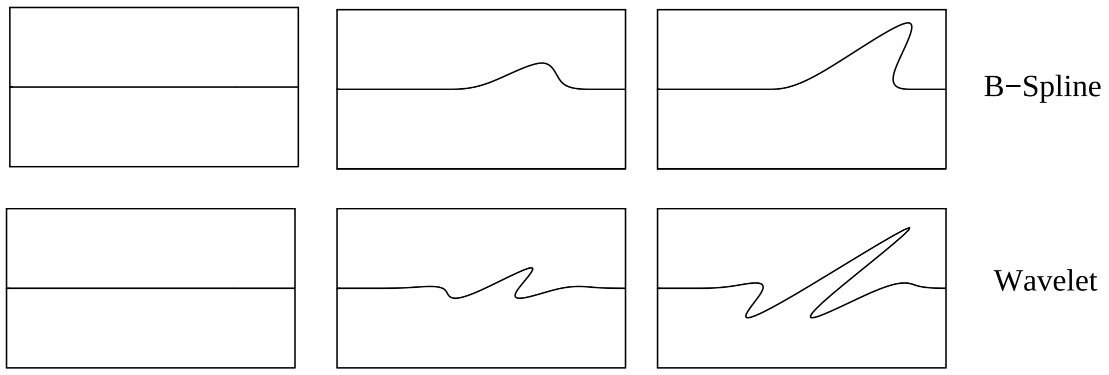


Figure 29: When B-spline coefficients are manipulated, the curve responds in a "hump" like fashion. When wavelet coefficients are manipulated, the curve responds in a "wave" like fashion.

approach may be considered as an extension to the Hierarchical B-spline approach described in [30, 31]. In the Hierarchical B-spline approach, there is no unique representation of any curve or surface. When a user modifies some "broad" B-spline control points, the affect on "finer" control points is computed using B-spline refinement rules. But when a user modifies fine control points, there is no notion of what affect this has on broader control points. In contrast, wavelets form a linearly independent basis, and so each curve or surface has a unique representation.

6.2 Control Handles

The first approach one might attempt is to have the user directly manipulate the wavelet coefficients. This is not likely to produce an intuitive interface. Moving such a control point, and thus changing the amount of some wavelet basis function used, changes the solution in a "wave" like fashion. In contrast, it is more intuitive to move a B-spline control point which changes the solution in a "hump" like fashion (see Figure 29).

To allow for multiresolution editing, a hybrid method is employed. Instead of using the complete wavelet basis

$$\{\phi_{0,j}, \psi_{i,l}\} \; 0 \leq i \leq L-1 \tag{133}$$

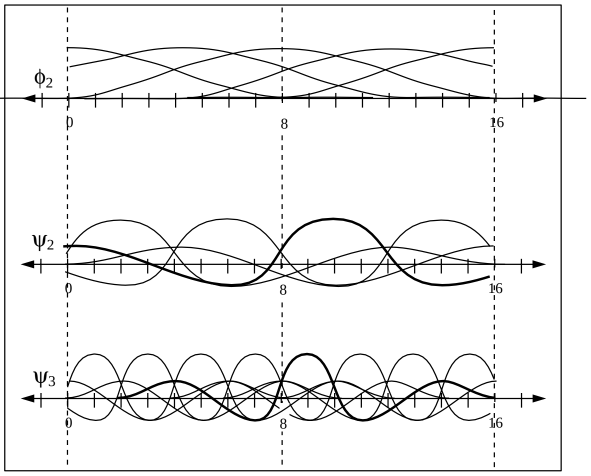


Figure 30: The hybrid basis, with $L = 4$ and $r = 2$. This basis consists of B-spline basis functions at level 2 and wavelet functions at the levels 2 and 3.

0 $\boxed{\texttt{BB}}$
$\boxed{\texttt{W}}$

1 `BBB`
$\boxed{\texttt{WW}}$

2 `BBBBB`
$\boxed{\texttt{WWWW}}$

3 `BBBBBBBBB`
$\boxed{\texttt{WWWWWWWW}}$

4 `BBBBBBBBBBBBBBBBB`

Figure 31: At each level a vector of B-spline ("B") and wavelet ("W") coefficients is maintained. Not all of these coefficients are necessary for all of the hybrid bases. The necessary coefficients for the hybrid basis with $r = 0$ are in boxes.

or the B-spline basis

$$\{\phi_{L,j}\} \tag{134}$$

a variety of *hybrid* bases for different fixed resolutions r are employed

$$\{\phi_{r,j}, \psi_{i,l}\}\ r \leq i \leq L-1 \tag{135}$$

$0\ \texttt{BB}$
$\ \texttt{W}$

$1\ \boxed{\texttt{BBB}}$
$\ \boxed{\texttt{WW}}$

$2\ \texttt{BBBBB}$
$\ \boxed{\texttt{WWWW}}$

$3\ \texttt{BBBBBBBBB}$
$\ \boxed{\texttt{WWWWWWWW}}$

$4\ \texttt{BBBBBBBBBBBBBBBBB}$

Figure 32: The necessary coefficients for the hybrid basis with $r = 1$ are in boxes.

$0\ \texttt{BB}$
$\ \texttt{W}$

$1\ \texttt{BBB}$
$\ \texttt{WW}$

$2\ \boxed{\texttt{BBBBB}}$
$\ \boxed{\texttt{WWWW}}$

$3\ \texttt{BBBBBBBBB}$
$\ \boxed{\texttt{WWWWWWWW}}$

$4\ \texttt{BBBBBBBBBBBBBBBBB}$

Figure 33: The necessary coefficients for the hybrid basis with $r = 2$ are in boxes.

0 BB
  W

1 BBB
  WW

2 BBBBB
  WWWW

3 $\boxed{\texttt{BBBBBBBBB}}$
  $\boxed{\texttt{WWWWWWWW}}$

4 BBBBBBBBBBBBBBBBB

Figure 34: The necessary coefficients for the hybrid basis with $r = 3$ are in boxes.

0 BB
  W

1 BBB
  WW

2 BBBBB
  WWWW

3 BBBBBBBBB
  WWWWWWWW

4 $\boxed{\texttt{BBBBBBBBBBBBBBBBB}}$

Figure 35: The necessary coefficients for the hybrid basis with $r = 4$ are in boxes.

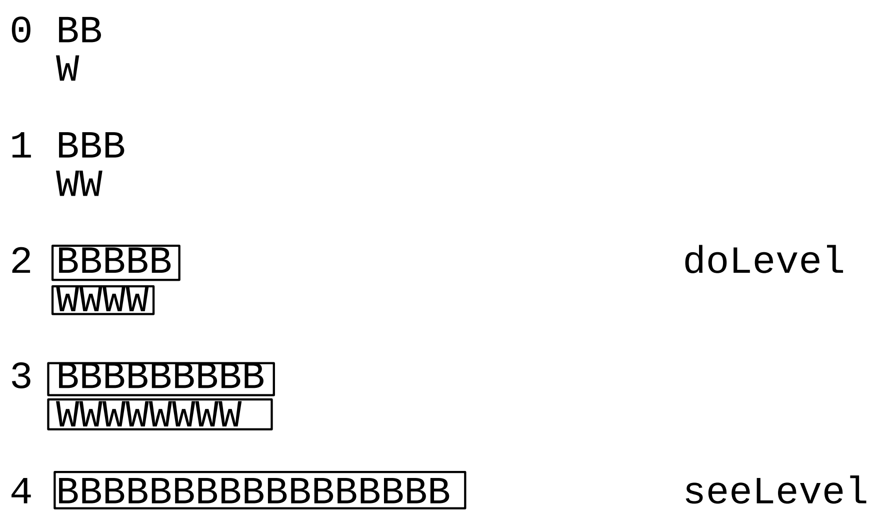


Figure 36: A `partial_pyrm_down` has been performed between levels 2 and 4. All B-spline coefficients between these two levels become valid.

Figure 3 (of Chapter 2) shows the B-spline basis (with $L = 4$) this can be considered a hybrid basis with $r = 4$. Figure 11 shows the complete wavelet basis, this can be considered a hybrid basis with $r = 0$. Figure 30 shows the hybrid basis with $r = 2$. Each of these bases has B-spline basis functions at its "top" resolution level r, and wavelet basis functions at that resolution, and below. Each of these bases is comprised of 17 basis functions.

In multiresolution editing, the parameter r is chosen dynamically by the user. It specifies the resolution level at which the basis includes B-spline basis functions. The user then manipulates the corresponding B-spline control points at the desired level r. If the user sets $r = L$ then the chosen basis is the original B-spline basis. If the user sets $r = 0$, then the chosen basis is the complete wavelet basis. In between 0 and L are a sequence of hybrid bases. Each of these hybrid bases is made up of translations and scales of the wavelet shape, ψ, and translations of the B-spline shape, ϕ, at resolution r. Choosing a basis with small r allows the user to manipulate rather coarse B-spline basis functions, while choosing a basis with large r allows the user to manipulate finer B-spline basis functions.

Transformations between the hybrid bases at two chosen levels r_{in} and r_{out}, are done using modified `coef_pyrm_down` and `coef_pyrm_up` procedures. Instead of always transforming between levels 0 and L, like the complete pyramid procedures, these procedures only apply the requested number of `xform` transformations.

For these procedures, it will be assumed that at each level i with $0 \leq i \leq L$, two vectors of coefficients b and w are maintained. In order to represent a curve in one of the hybrid bases, not all of the coefficients in these vectors need to be used, just the ones required by the particular hybrid basis. Figures 31-35 show which coefficients need to be valid in the hybrid bases with $L = 4$.

Upon input to the modified pyramid procedures, it is assumed that the there are valid coefficients for the hybrid basis at level r_{in}, (i.e., the ψ coefficients are all valid from level $L-1$ up through r_{in} and the ϕ coefficients are valid at level r_{in}). Upon output, there must be valid coefficients for the hybrid basis at level r_{out} (i.e., the ψ coefficients must be valid from level $L-1$ up through r_{out} and the ϕ coefficients must be valid at level r_{out}).

As a side effect of these procedures, the ϕ coefficients will also become valid at all levels between r_{in} and r_{out}. Figure 36 shows which coefficients will be valid when this transformation is done between resolution levels 2 and 4.

Here is the procedure `partial_coef_pyrm_up` which assumes that $r_{\text{in}} > r_{\text{out}}$, and the procedure `partial_coef_pyrm_down` which assumes that $r_{\text{in}} < r_{\text{out}}$: The structures b and w are passed by reference, and are used for both input and output.

```
partial_coef_pyrm_up( $b$[][], $w$[][], $r_{\text{in}}$, $r_{\text{out}}$ )
    for( $i = r_{\text{in}}$; $i > r_{\text{out}}$; $i--$ )
        coef_xform_up( $b[i][]$, $b[i-1][]$, $w[i-1][]$, $i$ ) ;
```

```
partial_coef_pyrm_down( $b$[][], $w$[][] , $r_{\text{in}}$, $r_{\text{out}}$ )
    for( $i = r_{\text{in}}+1$; $i \leq r_{\text{out}}$; $i$++ )
        coef_xform_down( $b[i-1][]$, $w[i-1][]$, $b[i][]$, $i$ ) ;
```

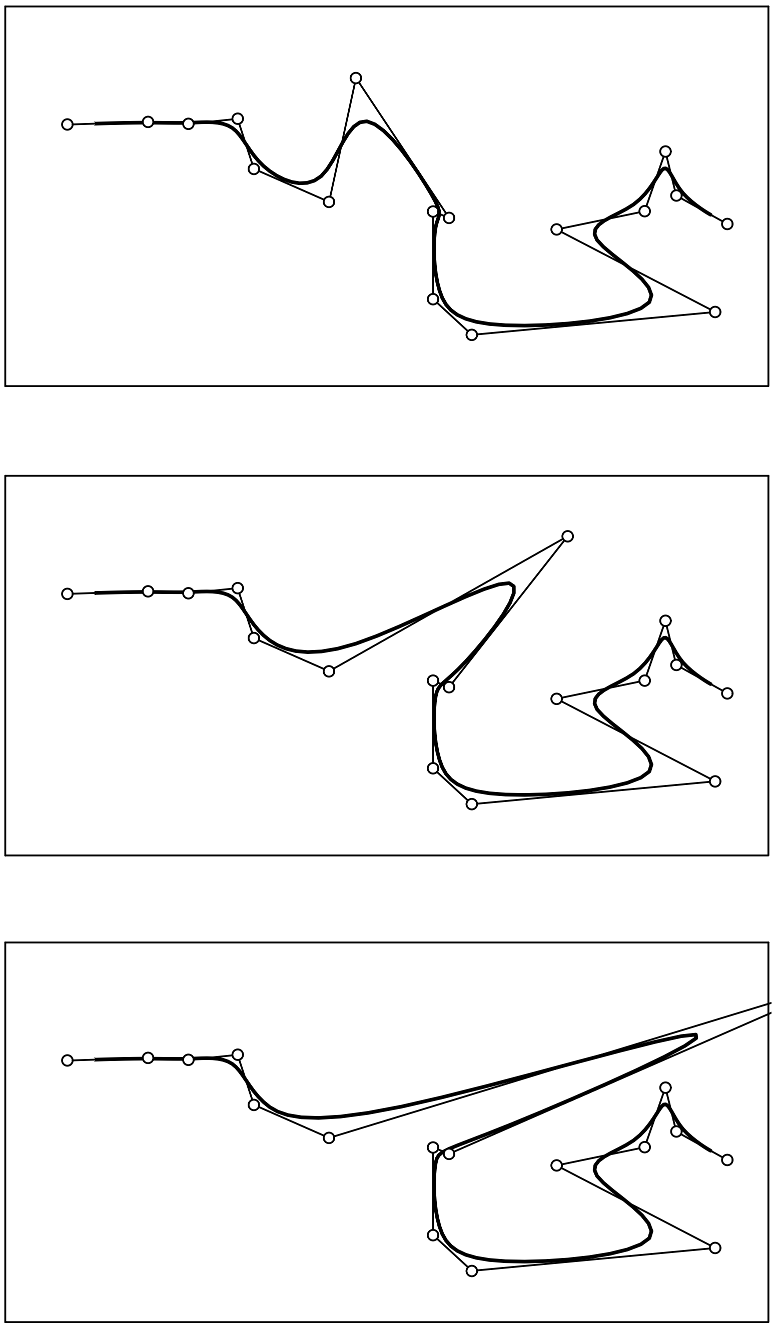


Figure 37: The user edits the fine detail of the curve. The `doLevel` is set to 4.

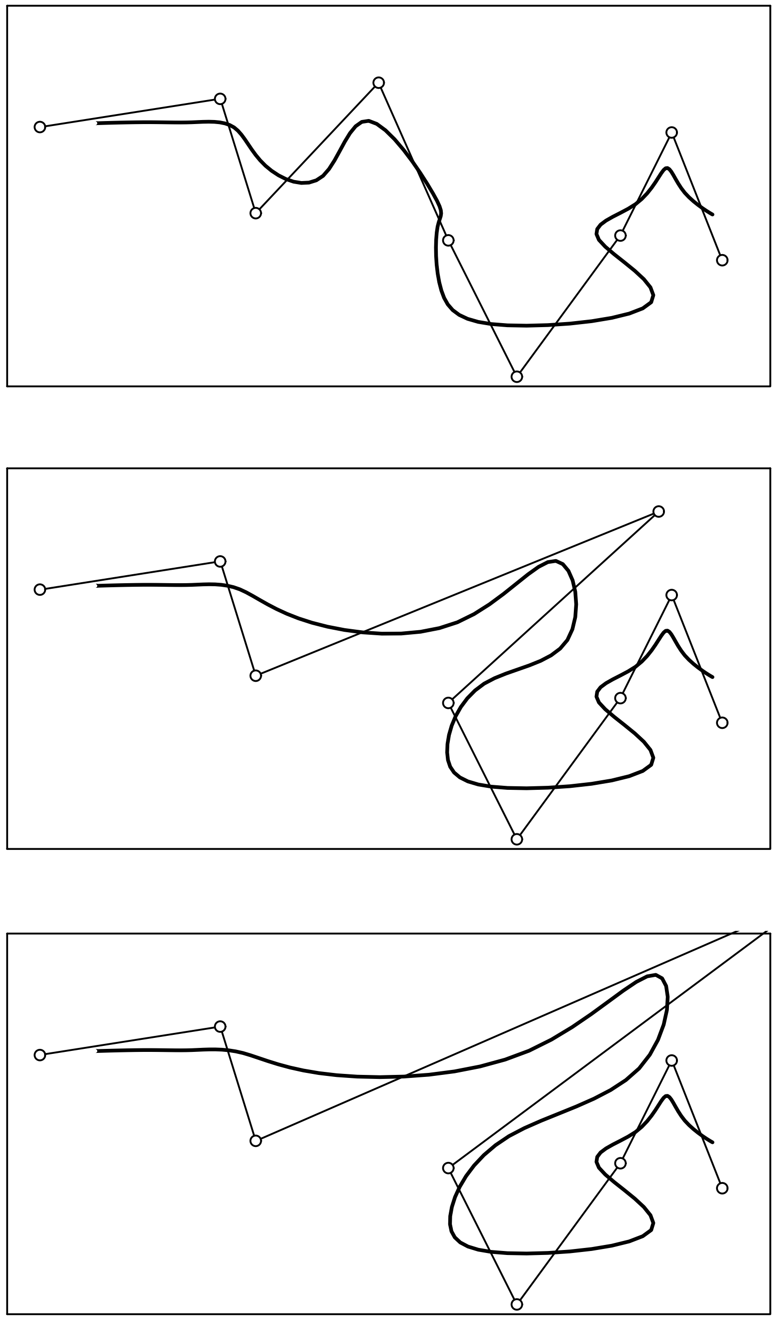


Figure 38: The user edits the curve at a medium resolution. The `doLevel` is set to 3.

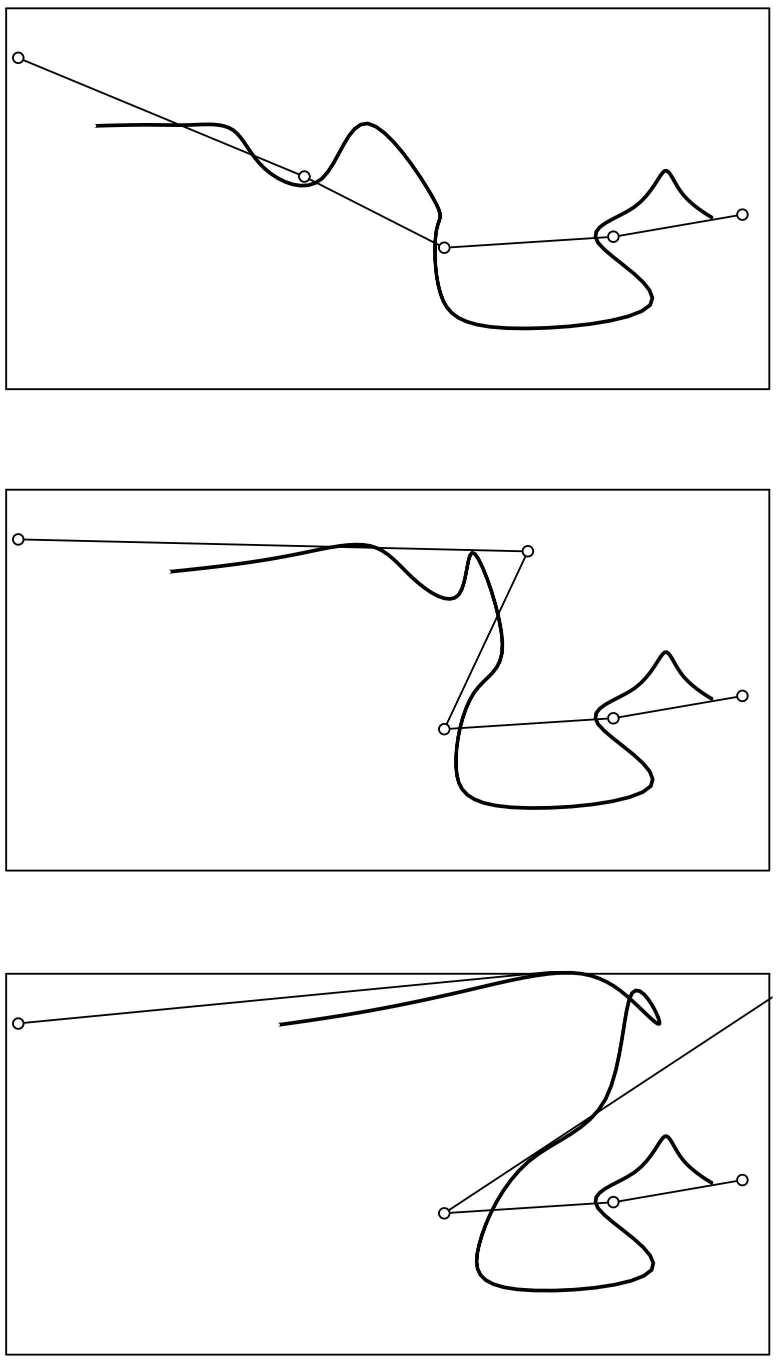


Figure 39: The user edits the overall sweep of the curve. The `doLevel` is set to 2.

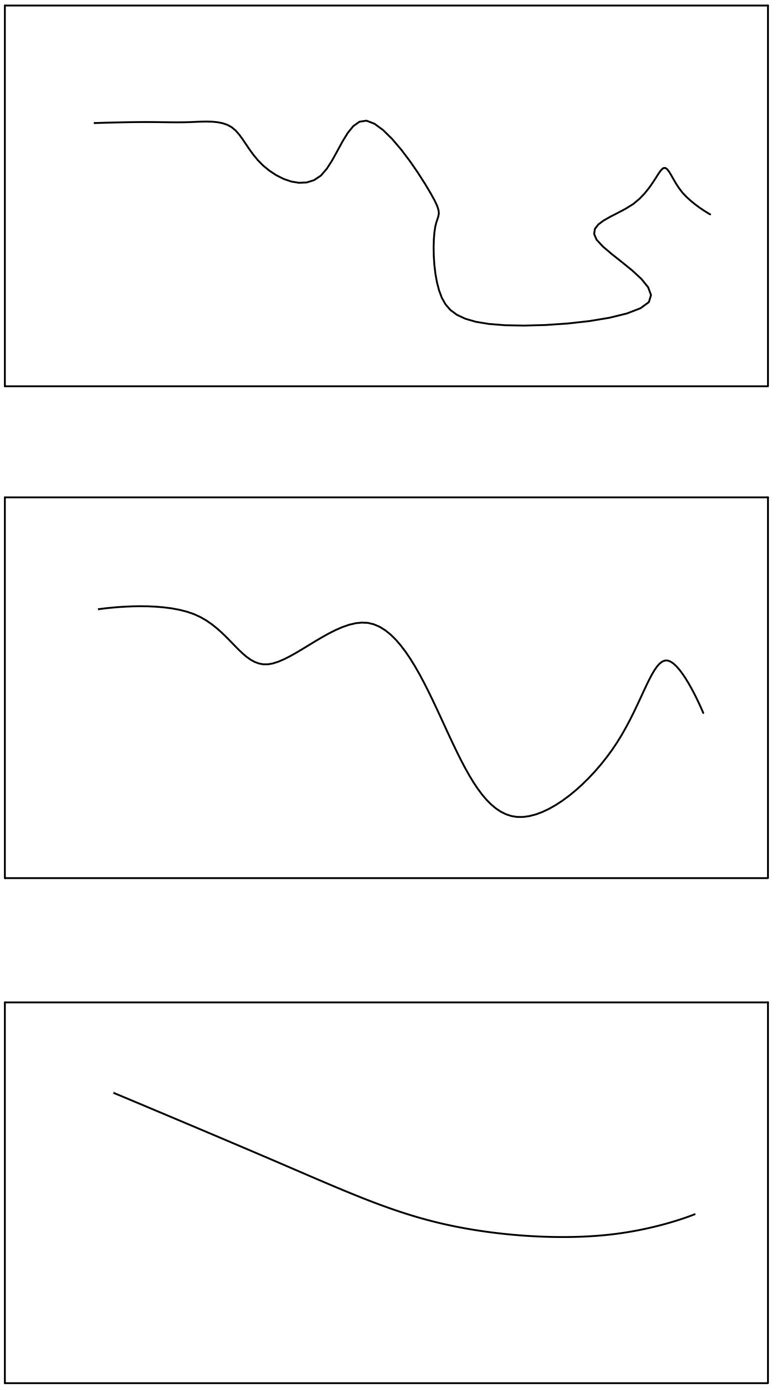


Figure 40: The user views the curve at three different resolutions by setting the `seeLevel` to 4, 3 and 2 (top to bottom).

When performing multiresolution editing, there are two resolution levels that are significant. The resolution at which the user views the curve, called r_s or the `seeLevel`, and the resolution at which the user manipulates the curve, called r_d or the `doLevel`. These levels are chosen dynamically by the user, and can be set independently. Therefore two valid representations of the same curve must be maintained.

The B-spline coefficients at the `seeLevel` are used to draw the curve using a GL `nurbscurve` call [41]. If the `seeLevel` is set to L the user views the complete resolution curve. If it is set to a smaller number, then the user views a smoother projected version of the curve. Figure 40 displays the same curve with the `seeLevel` set to 4, 3, and 2.

For manipulate purposes, the B-spline coefficients at the `doLevel` are displayed as a control polygon. The user can then manipulate these B-spline coefficients by clicking and dragging on them with a mouse. This allows the user to alter the curve in a "hump" like fashion at the `doLevel` resolution.

In general the `doLevel` and `seeLevel` can be set to any desired levels. One useful combination of these levels is for the user to fix the `seeLevel` at L, so he always views the complete curve. The `doLevel` can then be adjusted to various levels allowing manipulation of the curve at different resolutions. Figure 37 shows a user manipulating a curve with the `doLevel` set to 4 (the `seeLevel` is also set to 4). Figure 38 shows the user manipulating the same curve with the `doLevel` set to 3 and the `seeLevel` set to 4. The user is shown a coarser control polygon, and the manipulation is at a broader scale. Figure 39 shows the user manipulating the same curve with the `doLevel` set to 2 and the `seeLevel` set to 4. This allows for manipulation at even a broader scale.

Another useful combination is for the user to always set the `doLevel` and `seeLevel` be the same, viewing and manipulating smoother versions of the curve. Figure 40 shows a curve displayed at three different `seeLevels`.

In order to maintain both a `doLevel` and a `seeLevel`, hybrid bases for both r_d and r_s must be maintained. This can be done using the the b and w vectors described above and shown in Figures 31-35. In order to maintain both hybrid bases, there must be valid coefficients for both bases. In addition, because the partial pyramid procedures are used between these levels, the B-spline coefficients between the `doLevel` and the `seeLevel` will also be valid. Figure 36 shows the valid coefficients with $r_d = 2$ and $r_s = 4$.

Using the mouse, the user can `manipulate` a B-spline coefficient at the `doLevel`. After the user manipulates a B-spline coefficient, the change must be propagated to the `seeLevel` representation so that the curve can be displayed at the proper resolution. The proper pyramid (up or down) is chosen depending on the order of the `doLevel` and `seeLevel`.

```
manipulate( $b$[][], $w$[][] , $r_d$, $r_s$ )
    change_coef($b[r_d][]$) ;
    if ( $r_d < r_s$)
        partial_pyrm_down( $b[][]$, $w[][]$, $r_d$, $r_s$ ) ;
    if ( $r_d > r_s$)
        partial_pyrm_up( $b[][]$, $w[][]$, $r_d$, $r_s$ ) ;
    display_curve($b[r_s][]$);
```

The user can also interactively change either the `seeLevel` or `doLevel` up or down by one level. This is implemented in the following procedures.

```
upSeeLevel( $r_d$, $r_s$ )
    if ( $r_s > r_d$)
        $r_s$-- ;
    else
        partial_pyrm_up( $b[][]$, $w[][]$, $r_s$, $r_s$-1 ) ;
        $r_s$-- ;
```

```
downSeeLevel( $r_d$, $r_s$ )
    if ( $r_s < r_d$)
        $r_s$++ ;
    else
        partial_pyrm_down( $b[][]$, $w[][]$, $r_s$, $r_s$+1 ) ;
        $r_s$++ ;
```

```
upDoLevel( $r_d$, $r_s$ )
    if ( $r_s < r_d$)
        $r_d$-- ;
    else
        partial_pyrm_up( $b[][]$, $w[][]$, $r_d$, $r_d$-1 ) ;
        $r_d$-- ;
```

```
downDoLevel( $r_d$, $r_s$ )
    if ( $r_s > r_d$)
        $r_d$++ ;
    else
        partial_pyrm_down( $b[][]$, $w[][]$, $r_d$, $r_d$+1 ) ;
        $r_d$++ ;
```

In the above procedures, because the B-spline coefficients on the levels between the `doLevel` and the `seeLevel` are valid, no pyramid transformations are necessary when the levels are moving "towards" each other. When the levels are moving "away from" each other, then a modified pyramid up or down is performed.

This modeling method is quite similar to the Hierarchical B-spline editing method described in [30] where the resulting curve (or surface) is defined as the superposition of the hierarchical B-spline functions

$$\{\phi_{i,j}\}\ 0 \leq i \leq L \tag{136}$$

and the user directly manipulates the hierarchical B-spline control points. In that method there are many combinations of B-spline control points that give rise to the same curve. The actual representation that gets used is determined by which control points the user dragged by some amount to build this curve. When a user adds some detail at some fine level, there is no notion of this having any affect at any coarser level description. In the wavelet method the user also manipulates hierarchical B-spline control points, but each curve has a unique wavelet representation, and a unique projected representation $\phi_{i,j}$ at every level i (see Figure 40).

6.2.1 Tangent, Normal, Binormal Frame

In the parametric representation, the curve or surface is represented by three functions X, Y, Z. In the the multi-resolution paradigm, when a user adds in some fine directional detail, say a fine hump in the X direction, this detail is locked in the originally chosen direction. If the user later manipulates the broad sweep of the curve, the detail will still maintain its direction (see Figure 41). This is not fully desirable, since the user may want the detail's orientation to follow the broad sweep.

An "orientation" approach used with hierarchical B-splines [30] may be applied to the wavelet multiresolution editing scheme, as suggested by Adam Finkelstein [28]. In a multiresolution representation, each level of the hierarchy represents a further level of detail. Usually this detail is expressed as three independent functions of x, y, z. The basic idea of the orientation approach is to instead represent this detail with respect to the geometric shape of the lower resolution version of the curve. In particular this is done by computing the tangent and normal directions of the lower resolution curve at a series of sample points. The detail is then expressed with respect to these tangent and normal directions instead of the x, y, z directions. If the smooth component of the curve is altered, the detail's orientation will change appropriately.

The technical details of this "orientation" approach are as follows: Focusing on just one level of the multiresolution representation, the `xform` procedure is used to transform the B-spline coefficients

$$(x_{\phi_{L,j}}, y_{\phi_{L,j}}, z_{\phi_{L,j}}) \tag{137}$$

into the two-part coefficients

$$\begin{aligned}&(x_{\phi_{L-1,j}}, y_{\phi_{L-1,j}}, z_{\phi_{L-1,j}})\\ &(x_{\psi_{L-1,j}}, y_{\psi_{L-1,j}}, z_{\psi_{L-1,j}})\end{aligned} \tag{138}$$

Then each particular triplet of wavelet coefficients (for each particular j)

$$(x_{\psi_{L-1,j}}, y_{\psi_{L-1,j}}, z_{\psi_{L-1,j}}) \tag{139}$$

is rewritten in the tangent, normal, binormal frame:

$$(t_{\psi_{L-1,j}}, n_{\psi_{L-1,j}}, b_{\psi_{L-1,j}}) \tag{140}$$

.

The tangent, normal, and binormal directions provide an orthogonal set of directions based on the geometry of the curve. The *tangent* direction of the curve at parameter value t_j is in the direction of the first derivative (with respect to t) of the curve.

$$(\dot{x}(t_j), \dot{y}(t_j), \dot{z}(t_j)) \tag{141}$$

The *binormal* direction is perpendicular to the directions of the first and second derivatives of the curve. This can be found using the cross product.

$$(\dot{x}(t_j), \dot{y}(t_j), \dot{z}(t_j)) \times (\ddot{x}(t_j), \ddot{y}(t_j), \ddot{z}(t_j)) \tag{142}$$

The *normal* direction is perpendicular to the tangent and the binormal directions.

$$tangent \times binormal \tag{143}$$

If the curve lies in a plane, then the binormal vector is perpendicular to that plane.

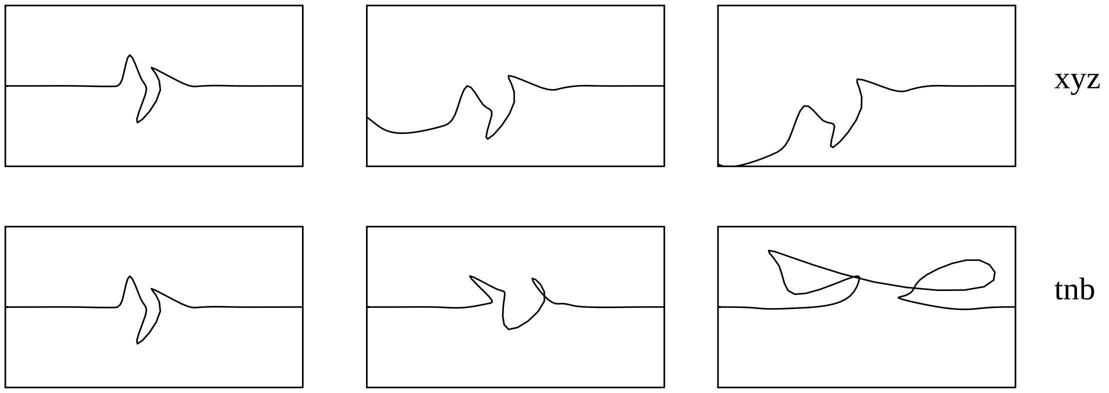


Figure 41: When the xyz frame is used for wavelet multiresolution editing, detail maintains its orientation as the sweep is changed. When the (t, n, b) frame is used, the detail does not maintain its structure as the sweep is changed.

To perform this transformation, the first and second derivatives must be taken of the smooth curve defined by the $\phi_{L-1,j}$ functions. This can be done symbolically using the piecewise cubic definition of the B-spline basis functions. The derivatives are evaluated at the each parameter location t_j where the wavelet basis function $\psi_{L-1,j}$ has its greatest influence (at its center). For example, the zeroth wavelet basis function, $\psi_{L-1,0}$, is centered at parameter value $t = 7$, see Figure 7. Given the tangent, normal, binormal frame, for each j, the triplet of wavelet coefficients

$$(x_{\psi_{L-1,j}}, y_{\psi_{L-1,j}}, z_{\psi_{L-1,j}}) \tag{144}$$

which defines a vector in (x, y, z) coordinates, is rewritten as the same vector in (t, n, b) coordinates

$$(t_{\psi_{L-1,j}}, n_{\psi_{L-1,j}}, b_{\psi_{L-1,j}}) \tag{145}$$

This triplet of coordinates is then stored, instead of the (x, y, z) triplet, to represent the wavelet component of the curve. Later if the broad sweep of the curve is changed, this defines new tangent, normal, binormal frames at the t_j. The stored (t, n, b) triplets are used with these new frames. Using this method, the wavelet description of detail follows the broad orientation of the curve.

Even though this idea works well in the context of hierarchical B-splines [30], in the wavelet context, this is not as successful. When the user changes the broad sweep of the curve, changing the tangent, normal, and binormal frame at t_j, this remixes the wavelet functions. Since these wavelet functions are "wave" shaped, the resulting curve includes non-intuitive wiggly changes, (see Figure 41). This method appears to work better with hierarchical B-splines, where changing the (t, n, b) frame remixes a set of x, y, z finer B-spline functions which are "hump" shaped with very small support.

6.3 Least Squares

A multiresolution editing approach can also be applied to the least squares editing paradigm. The user interactively specifies the manipulation resolution using the `doLevel` r_d parameter. When r_d is small, the user wishes the constraints to have a broad effect, and when r_d is large, the user wishes the constraints to have a narrow effect.

Given the specified r_d, a modified `pyrm` procedure is used to put the curve in the proper basis (Equation (135)). The ψ basis functions are "locked" and the least squared solution is found over the $\phi_{r_d,j}$ coefficients. The least squared solution may be found using the pseudoinverse [32].

The least squared problem can also be posed as a minimization problem [76], whose solution can be found by solving following linear system that is related to Equation (131):

$$\begin{vmatrix} \mathbf{I} & \mathbf{A^T} \\ \mathbf{A} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \mathbf{x} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{x_0} \\ \mathbf{b} \end{vmatrix} \tag{146}$$

where $\mathbf{x_0}$ are the control points of the current curve or surface. Informal experiments have shown that this system can be solved using just a few (< 10) conjugate gradient iterations, and thus leads to a useful interactive modeling tool.

It should be noted that the hierarchical least squares method described in this section can be implemented with a bit of extra work using a "delta" method, without using the wavelets. Given a present curve c_i and a new set set of constraints, the new solution c_{i+1} can be expressed as the old curve plus some "delta" curve that must be solved for

$$c_{i+1}(t) = c_i(t) + \Delta_i(t) \tag{147}$$

The constraints for c_{i+1} can be reexpressed as constraints on the Δ_i curve. For example if the new curve at parameter value t_0 is constrained to interpolate the geometric point b_0, this can be achieved by constraining the delta curve to interpolate the point

$$b_0 - c_i(t_0) \tag{148}$$

Finally a "minimal" least squares solution for the delta curve is found that meets these reexpressed constraints, and minimizes the sum of the squares of the $\Delta_i(t)$ coefficients. One is free to choose any set of basis functions to represent the new delta curve independent of the basis used to represent the resulting curve c_{i+1}. In particular, one can choose B-spline basis functions of various resolutions to represent the delta curve. This will allow multi-resolution least squares editing.

It should also be noted here that when the (t, n, b) frame is being used, simple constraints such as interpolation are no longer linear, and so a more complicated solution process is required to solve such a case.

Pentland describes a related method where complicated variational problems are solved quickly using wavelets [58]. The system of Equation (131) is set up using a wavelet basis instead of a finite-element basis, and then all of the off diagonal terms of **H** are discarded. This new linear system can then be solved very rapidly. The justification given in [58] for this method cites Beylkin [9] and claims that due to the vanishing moments of the wavelet basis, most of the off diagonal terms are negligible and so can be ignored.

Although the arguments of Beylkin are valid in the context of integral equations (like radiosity) where the matrix coefficients are defined by an expression like

$$\int ds \int dt \ K(s,t)\psi_{i,j}(s)\psi_{k,l}(t) \tag{149}$$

and the vanishing moments cancel against a smooth K, in a variational context, the matrix terms are defined by some expression like

$$\int dt \ \ddot{\psi}_{i,j}(t)\ddot{\psi}_{k,l}(t) \tag{150}$$

where the vanishing moments are irrelevant. In fact, as will be seen in Section 7.4.1, the wavelet matrix in a variational problem is usually more dense than the finite element matrix. By discarding the off-diagonal terms of the wavelet matrix, Pentland is solving a system like Equation (146) and so is implicitly solving a (weighted) least squared problem in the wavelet basis. This method does produce smoother solutions than are obtained by a least squared solution using a finite element basis, because when a wavelet basis is used the least squares solution affects all resolutions. The properties of the wavelet least squares interpolant deserve further study.

Jawerth et al. [48] discuss a methodology, where given a differential equation (as opposed to a variational problem), a special wavelet basis can be constructed on the fly so that the finite-element matrix is indeed diagonal. In essence this representation constructs a special basis, whose least squared solution is actually the minimum energy solution. It would be interested to investigate how this method can be used in a modeling context.

6.4 Discussion

This chapter has discussed how wavelets can be used in the context of multiresolution editing. A hierarchical control handles method has been discussed as well as a hierarchical least squares method. Both of these methods could be implemented using hierarchical B-splines, without wavelets (although the use of Hierarchical B-splines in the least squared context has not appeared in the literature). Wavelet analysis elegantly extends the Hierarchical B-spline framework by including the notion that any curve or surface has a unique projected smoother version at every resolution.

A brief analysis of a wavelet least squares method is also given. This area could certainly use more investigation.

Chapter 7

Variational Modeling with Wavelets

7.1 Contribution

In the *variational modeling* paradigm the user directly manipulates the object, creating constraints, and the modeling tool returns the "best" object that meets the constraints (where the notion of best is typically measured the amount of curvature in the object). Solving for such an optimal surface is a time-consuming numerical task. This chapter will discuss how the wavelet representation allows for fast numeric solutions of the associated systems of equations. This allows the tool to respond more quickly. In contrast to the radiosity equation where the wavelets are used to obtain a sparse system, in variational modeling wavelets are used to obtain a well conditioned system that can be solved using fewer iterations than required with a more standard representation such as the B-spline basis.

In the *variational modeling* paradigm, the wavelet basis allows an optimizing procedure the ability to alter the curve or surface at a variety of resolutions. This allows for a much more efficient solution process than could be obtained by using a single resolution basis, such as uniform cubic B-splines. The hierarchical nature of the wavelet basis also allows the optimizing procedure to work top-down; beginning with a small coarse basis and locally adding in finer and finer wavelet basis functions where needed as the process continues.

The optimization work described in this chapter is related to the work of a variety of researchers. Qian et al. and Jaffard et al. [60, 47] use wavelets to quickly solve simple differential equations. In work more related to the topic at hand, Szeliski [71] uses a hierarchical pyramid basis change to perform efficient surface reconstruction from sparse data, to be used for machine vision problems. The pyramid basis change described there, works bottom-up starting with a uniform basis at some fixed resolution. The method described in this chapter uses a wavelet basis, where the basis functions are all translations and dilations of a single function. Thus, the optimizer can work *adaptively* in a top-down fashion, performing more "refinement" in some regions and less in others, and not stopping at any fixed resolution level. Also, by using a basis derived from uniform cubic B-splines, this method is *compatible* with other modeling software tools.

In other related work, Pentland [58] also describes a method similar to Szeliski. Independent of this work, Yaou et al. [80] discuss a wavelet optimization approach similar to the one described here. These approaches are not based on B-splines, and no adaptive refinement method is investigated. Also these researchers do not explore the *implicit* matrix representation, which is essential for efficiency.

After describing the relevant theory this chapter discusses a variety of implementation issues and reports some experimental results.

7.2 Minimization

In contrast to the control handles and least squares modeling paradigms, where the basis chosen defines the semantics of the user's manipulation, in variational modeling the minimum energy curve or surface is sought. The energy of a curve or surface is independent of the basis used [1], however in variational modeling the choice of basis can have a strong impact on the efficiency of the solution method.

[1] Although energy measurements such as thin plate are sensitive to the particular parameterization of the curve or surface.

Iterative methods are usually employed to solve minimization problems for a number of reasons. For most non-quadratic minimization problems, there is no direct way to find the minimum. Instead, one must progress downhill from some initial guess to a local minimum. Even for linear systems/quadratic minimization problems (where there are many direct solution methods) descent methods are also used if the system is sparse because each iteration can be done in linear time. Iterative methods are also useful because as one performs more iterations, a progressively better solution is obtained. A user can therefore quickly see a rough solution. Iterative methods are also good for updating solutions when a new constraint is added, since the previous solution often provides an excellent starting guess for the new solution [2]. This section will briefly review various iterative methods, and their behavior under basis change.

7.2.1 Quadratic Functions

Suppose one wishes to solve the linear system

$$\mathbf{M}\mathbf{x} = \mathbf{b} \tag{151}$$

If $\mathbf{M}$ is a symmetric positive definite matrix, then the solution $\mathbf{x}$ is also the minimum of the quadratic function

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{t}}\mathbf{M}\mathbf{x} - \mathbf{x}^{\mathbf{t}}\mathbf{b} \tag{152}$$

And if $\mathbf{M}$ is not positive definite (but is non-singular), then one can instead solve the equivalent positive definite system

$$\mathbf{M}^{\mathbf{T}}\mathbf{M}\mathbf{x} = \mathbf{M}^{\mathbf{T}}\mathbf{b} \tag{153}$$

which minimizes the ℓ^2 norm of the residual [3].

There are a variety of related descent methods used to minimize the quadratic function (Equation (152)). Given a current guess, $\mathbf{x}^{(\mathbf{k})}$, these methods pick a descent direction, and then travel along that direction by some amount. Often the distance

[2] QR factorization updating is another good method for quickly updating a solution as new constraints are added [37].

[3] This squaring of the matrix has the unfortunate consequence of squaring the condition number.

is chosen that minimizes the quadratic function along that direction [4]. Choosing this distance in general is called *line search*, but for quadratic functions the desired distance can be computed directly without any iterative searching.

Steepest descent descends in the steepest downhill direction, which is the direction of the residual vector

$$\mathbf{r}^{(\mathrm{k})} = \mathbf{M}\mathbf{x}^{(\mathrm{k})} - \mathbf{b} \tag{154}$$

If there are $O(n)$ non-zero terms in the matrix, then this direction can be computed in linear time.

Gauss-Seidel relaxation iteratively descends in each of the axis directions. At the minimum point along each axis, the corresponding entry of the gradient vector $\mathbf{r}^{(\mathbf{k})}$ is zero. Thus a descent step for variable i can be calculated as

$$\Delta\mathbf{x}_i = \mathbf{b}_i - \sum_j \mathbf{M}_{i,j}\mathbf{x}_j \tag{155}$$

If there are $O(n)$ non-zero terms in the matrix, then successively descending down the n axis directions, called one complete Gauss-Seidel sweep, can be done in linear time. A complete Gauss-Seidel sweep can be expressed as

$$\mathbf{D}\mathbf{x}^{(\mathrm{k+1})} = -\mathbf{L}\mathbf{x}^{(\mathrm{k+1})} - \mathbf{U}\mathbf{x}^{(\mathrm{k})} + \mathbf{b} \tag{156}$$

where $\mathbf{D}$, $\mathbf{L}$, and $\mathbf{U}$ are the the diagonal, lower and upper components of the matrix $\mathbf{M}$.

Conjugate Gradient iteration descends down a set of mutually *conjugate* directions. Two directions $\mathbf{a}$ and $\mathbf{b}$ are called conjugate if

$$0 = \mathbf{a}^{\mathbf{T}}\mathbf{M}\mathbf{b} \tag{157}$$

A new conjugate direction can be found by doing a small number of matrix vector multiplies in linear time given a sparse matrix [59]. In exact arithmetic, the precise minimum should be found after considering n conjugate directions, but in floating point computation this does not occur, and further iterations are often required. Generally, conjugate gradient requires the fewest iterations for convergence. This is followed by Gauss-Seidel, which is followed by steepest descent [59, 71].

[4] Over-relaxation schemes travel a distance along the ray further than the minimizer.

7.3 Fast Convergence With Wavelets

The number of iterations required to obtain a satisfactory solution is sensitive to the basis chosen for representing the problem. In particular, when solving for a minimal curvature curve or surface using a fixed resolution basis, such as cubic B-splines, the optimizing procedure must make all of its changes at that fixed fine resolution, even if the necessary change is at some coarser resolution. This gives rise to very slow convergence. For example, in Gauss-Seidel iteration, the optimizer cycles through the basis functions in order, and for each one, adds in the "currently optimal" amount of that basis function. If one is using a fixed resolution basis, and the necessary change is at a coarser resolution, the Gauss-Seidel iteration cannot perform large changes for any particular basis function; that would lead to a high energy curve or surface. Instead, only small changes are made, until the optimal curve is finally approached. In contrast to the B-spline basis, the wavelet basis expresses the solution at a combination of resolutions, and so the optimizing program is able to make direct changes at a wide variety of resolutions, thus converging quickly.

Most of the theoretical results known in this area are for simple boundary value problem differential equations such as

$$\ddot{X}(t) + B(t)X(t) = G(t) \tag{158}$$

for some given functions B and G and some boundary conditions on X. For these problems, when a fixed resolution basis, such a B-spline basis, is used in a finite element approximation of X, the condition number κ of the resulting matrix grows as n^2, where n is the number of basis functions [47, 11]. The number of iterations required by the conjugate gradiant method to achieve convergence to within some epsilon grows proportionally to $\sqrt{\kappa}$ [71, 47]. As a result, obtaining accurate solutions to these problems using a finite element basis is very costly. When a multiresolution basis, such as a wavelet basis is employed to solve these problems, it can be shown that the number of iterations remains constant, independent of n [81, 47, 25]. Thus, wavelets provide a practical way of quickly obtaining solutions to many differential equations. For a more detailed discussion see Appendix C.

The constrained optimization problems that arise in geometric modeling are more complicated than the simple differential equations studied in the theoretical literature. In particular, in a modeling problem, the constraints may be placed anywhere in the domain of the curve or surface, and not just on the boundary. Therefore the proofs for the theoretical bounds for simple differential equations can not be directly applied to geometric modeling problems. But as will be shown in the experimental results section, in practice, the wavelet basis is a powerful method to reduce the number of iterations.

7.3.1 Scaling

The scaling of the basis functions is very significant for the behavior of the optimizing procedures. Traditionally [54, 58] the wavelet functions are defined with the following scaling:

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(i-L)/2}\ \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{(i-L)/2}\ \psi(2^{(i-L)}t - j) \end{aligned} \tag{159}$$

This means that at each level moving up, the basis functions become twice as wide, and are scaled $\frac{1}{\sqrt{2}}$ times as tall. While in many contexts this normalizing may be desirable, for optimization purposes it is counter productive. When the basis functions are normalized, the basis transformation to the wavelet basis can become close to an orthogonal transformation. An orthogonal basis transformation merely "rotates" the finite dimensional optimization problem, and cannot change its conditioning properties. For the optimization procedure to be well conditioned [47, 25] it is essential to emphasize the coarser levels. The correct theoretical scaling depends on both the energy function used, and the dimension of problem. For a fuller discussion, see Appendix C. In the experiments described in this thesis the following scaling was used

$$\begin{aligned} \phi_{i,j}(t) &= 2^{-(i-L)}\ \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{-(i-L)}\ \psi(2^{(i-L)}t - j) \end{aligned} \tag{160}$$

This means that as one goes from level i to level $i-1$ the basis functions become twice as wide, and 1/2 as tall. In the pyramid code, this is achieved by multiplying all of the h and g entries by 2, and all of the $\tilde{h}$ and $\tilde{g}$ by 1/2 [5].

7.4 The Wavelet Linear System

Transforming a problem from the cubic B-spline basis to the wavelet basis can be expressed using the basis transform matrix $\mathbf{W}$. The goal of this transformation is to produce a quickly converging system.

In the B-spline basis, the optimization procedure resulted in the linear system given by Lagrangian Equation (131) [6]. In the wavelet basis, a different Lagrangian linear system results which is given by

$$\begin{vmatrix} \bar{\mathbf{H}} & \bar{\mathbf{A}}^{\mathbf{T}} \\ \bar{\mathbf{A}} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \bar{\mathrm{x}} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{161}$$

where the bars signify that the variables are wavelet coefficients, $\bar{\mathrm{x}} = \mathbf{W}\mathbf{x}$, and the Hessian and constraint matrix are expressed with respect to the wavelet basis. To see the relationship with the B-spline system, the new system can also be written as

$$\begin{vmatrix} \mathbf{W}^{-\mathbf{T}}\mathbf{H}\mathbf{W}^{-1} & \mathbf{W}^{-\mathbf{T}}\mathbf{A}^{\mathbf{T}} \\ \mathbf{A}\mathbf{W}^{-1} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \bar{\mathrm{x}} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{162}$$

Although Equation (131) and Equation (161/162) imply each other, they are two distinct linear systems of equations. Because the wavelet system (161/162) is hierarchical it will not suffer from the poor conditioning of the B-spline system of Equation (131) (see Figure (53)). As explained above, the intuition behind the better behavior of the wavelet basis is due the ability of the relaxation method to make changes at any appropriate resolution.

[5]The proper scaling is essential to obtain the quick convergence of the wavelet method when steepest descent or conjugate gradient iteration is used. Scaling is not important with Gauss-Seidel iteration, which will perform the same sequence of iterations regardless of scale.

[6]This system is not positive definite so the squared system (Equation (153)) must be used with descent methods.

7.4.1 Implicit versus Explicit Matrix Representation

By using a wavelet basis instead of a finite element basis, fewer iterations are required to converge to a solution. It is important that extra computational costs are not accrued by going to the wavelet basis slowing down each iteration. This leads to the following choice. In an iterative conjugate gradient solver, the common operation is multiplication of a vector times the wavelet matrix given in Equations (161/162). There are two ways to implement this.

One approach, the *explicit* approach, is to compute and store the wavelet Hessian matrix $\bar{\mathbf{H}}$ and the wavelet constraint matrix $\bar{\mathbf{A}}$ (Equation (161)). These can be computed directly from a closed form (piecewise polynomial) representation of the wavelet functions $\psi_{i,j}$. For example, for wavelet curves, the Hessian terms of the thin plate energy are $\int dt\ \ddot{\psi}_{i,j}(t)\ddot{\psi}_{k,l}(t)$. Unfortunately, these matrices are not as sparse as the B-spline Hessian and constraint matrices for two reasons. Firstly, the wavelets are wider than B-spline functions. On a single level, i, there are 13 wavelet curve functions that overlap with a chosen wavelet, (instead of the 7 B-splines that overlap a single B-spline). For tensor product surfaces, there are over 300 wavelet functions that overlap with a chosen wavelet (instead of the 49 B-splines). Secondly, since wavelets are hierarchical, a single wavelet will overlap with wavelets on all L levels. As a result there are $O(n \lg n)$ overlapping wavelet basis functions, and thus the wavelet Hessian matrix is only $O(n \lg n)$ sparse.

Alternatively, there is the *implicit* approach [81, 71] which only computes and stores the entries of the B-spline matrices $\mathbf{H}$ and $\mathbf{A}$ (Equation (162)). In this approach, the multiplication of a vector $\mathbf{v}$ by a wavelet matrix $\bar{\mathbf{H}}$ or $\bar{\mathbf{A}}$ is computed as $\mathbf{W}^{-\mathbf{T}}\mathbf{H}\mathbf{W}^{-\mathbf{1}}\mathbf{v}$ or $\mathbf{A}\mathbf{W}^{-\mathbf{1}}\mathbf{v}$, where the matrix-vector multiplies involving $\mathbf{W}^{-\mathbf{1}}$ are implemented using the `coef_pyrm_down` (tensor version for surfaces), and the multiplies involving $\mathbf{W}^{-\mathbf{T}}$ are implemented using the `basis_pyrm_up` (tensor). These procedures are given in the Appendix. The advantage of this approach is that the whole multiply remains $O(n)$ in both time and space, since the `pyrm` procedures run in linear time, and the matrices $\mathbf{H}$ and $\mathbf{A}$ are $O(n)$ sparse. There is, of course, some constant factor of overhead introduced by executing the `pyrm` procedures.

Even though one of the methods explicitly uses wavelet terms while the other uses B-spline terms, these two methods are mathematically equivalent, and so both will have the same condition properties. On the other hand, they may have different space requirements, and require different amounts of time to compute a matrix-vector multiply. Both approaches have been implemented the results are discussed in Section 7.9.

The implicit approach can only be used when the iterative algorithm can be implemented with simple matrix vector multiplies. In particular, the conjugate gradient and steepest descent methods can be expressed this way. A complete Gauss-Seidel sweep, which is expressed in Equation (156), cannot be expressed as simple multiplications with the matrix $\mathbf{M}$, and so the implicit method may not be used for it.

7.5 Adaptive Oracle

By limiting the possible surfaces to only those that can be expressed as a linear combination of a fixed set of basis functions (perhaps uniform B-splines of a given size), one obtains an approximation of the true optimal surface. As more basis functions are added, perhaps by using smaller B-splines, the space of possible solutions becomes richer and a closer approximation to the true optimal surface can be made. Unfortunately, as the space becomes richer, the number of unknown coefficients increases, and thus the amount of computation required per iteration grows. A priori it may be difficult to know how many basis functions are needed to allow an approximation with a sufficiently small error. It is therefore desirable to have a solution method that adaptively chooses the appropriate basis functions given some user defined error bound. This approach was applied using hierarchical B-splines in [76]. When refinement was necessary, "thinner" B-splines basis functions were added, and the redundant original "wider" B-splines were removed. With wavelets, all that must be done is to add in new "thinner" wavelets wherever refinement is deemed necessary. Since the wavelets coefficients correspond directly to local detail, all previously computed coefficients are still valid.

The decision process of what particular wavelets to add and remove is governed by an `oracle` procedure which is called after every fixed number of iterations. The oracle must decide what level of detail is required in each region of the curve or surface. In [76] two criteria for refinement were suggested. Refinement to better satisfy a constraint, and refinement to obtain any solution with lower energy. Using B-splines, the first criterion is easy to implement because it is simple to detect that a constraint is not being satisfied, but there is no easy way to implement the second criterion because there is no direct way of knowing how much local detail is being used in the solution. In a wavelet basis the second criterion is also easy because the magnitude of a wavelet coefficient itself specifies the degree of detail in the solution thus far.

When some region of the solution does not need fine detail, the corresponding wavelet coefficients are near zero, and so the first thing the `oracle` does is to deactivate the wavelet basis functions whose corresponding coefficients are below some small threshold. The `oracle` then activates new wavelet basis functions where it feels more detail may be needed. There are two criteria used. If a constraint is not being met, then the oracle adds in finer wavelet functions in the region that is closest in parameter space to the unmet constraint. Even if all the constraints are being met, it is possible that more basis functions would allow the freedom to find a solution with lower energy. This is accomplished by looking for the current basis functions with coefficients above some maximum threshold, and activating finer wavelet functions in that parameter region. This is based on the reasoning that if the current solution requires detail in some region, it may be improved by adding even finer detail in that region. This can be summarized as follows:

```
oracle($w[][]$,constraints[])
   forall (i,j)
      if ($w[i][j]$ < minSize)
         deactivateWavelet($w[i][j]$);
      if ($w[i][j]$ > maxSize)
         activateWavelet(children($w[i][j]$));
   forall (k)
      if (error(constraint[k]) > errTol)
         activateWavelet(nearby(constraint[k]))
```

The function `children` returns the wavelets on the next finer level $i+1$ that are closest to the parent wavelet. For curves it returns two wavelet functions. For surfaces it returns the four wavelet functions. These four are of the same type ($\psi\psi$, $\psi\phi$, or $\phi\psi$) as the parent.

The function `nearby` returns the unactive wavelets nearest to the violated constraint.

There can be some tendency for the `oracle` to get into cycles; at one iteration it may activate a new basis function, and at some later iteration it may notice that the basis function is not being used and deactivate it. To avoid the immediate reactivation of basis functions, a basis function is marked as being `dormant` when it is removed from consideration. Of course, it is possible that later on the solution may really need this basis function, and so periodically there is a `revival` phase, where the `dormant` marks are removed.

When the oracle is used, the computation proceeds in a top down fashion, beginning with a coarse basis, and including more basis functions as the computation proceeds. This has the added advantage of starting with small simple systems where the iterations may be performed very quickly, and rough approximate solutions are obtained quickly. It is possible for these small systems to be better behaved than the larger systems representing a problem with more dimensions.

7.6 Replacing Wavelets with B-splines

When the implicit matrix representation is being used, the constraint and Hessian matrices, with respect to the wavelet basis functions, are computed by explicitly storing B-spline constraint and Hessian matrix entries, and using the pyramid transformation $\mathbf{W}$. But when an adaptive `oracle` method is being used, only a subset of the wavelet basis functions, $\psi_{i,j}$, may be active. And so it may be unnecessary to compute the constraint and Hessian matrices with respect to the *complete* uniform B-spline basis, $\phi_{L,j}$.

Instead, it is only necessary to compute matrix terms with respect to some minimal set of B-spline basis functions, $\phi_{i,j}$. This set, which may include B-spline basis functions from any resolution i, must span a space that *includes* the space spanned by the active wavelets. This set should also be as small as possible. This set can be computed using the following algorithm:

```
ComputeActiveBSplines(activeWavelets)
   activeBSplines = emptySet;
   forall(activeWavelets)
      (i,j) = index of active wavelet;
      activeBSplines += BSplines on level i+1 that span wavelet (i,j) ;
   forall (activeBsplines)
      (i,j) = index of active BSpline;
      if ( Bspline (i,j) is represented by finer active BSplines)
      activeBSplines -= BSpline (i,j);
```

In the above procedure, a *sufficient* B-spline set is found by replacing each wavelet function, $\psi_{i,j}$, with enough B-spline functions from one level lower $\phi_{i+1,j}$, that can construct it (Equation (16)). The resulting set of B-spline basis functions can represent any function that was represented by the active wavelets. Since this set contains B-spline basis functions from arbitrary levels i, it may be redundant. Therefore, in a second pass, this sufficient set is made smaller by removing the B-splines that can be constructed using other B-splines from the sufficient set that are on lower levels.

The `pyrm` procedures are also altered so that they perform transformations between the active wavelet set, and the minimal B-spline set. (The `for` loops in the `pyrm` procedures of the appendix only iterate over the necessary `j` and `k`). The running time of these altered `pyrm` procedures is linear in the number of active wavelets.

7.7 Comparison With Previous Approaches

There are a number of researchers who have discussed wavelet/hierarchical conditioning schemes. Terzopoulos discusses a multigridding approach for surface reconstruction problems that arise in computer vision [72]. In these problems, a sensor or camera has returned a sparse set of data points on some actual surface. The goal of the reconstruction process is to obtain a reasonable representation of the complete surface, using only the sparse data. In multigrid, a series of different problems at different resolutions are posed and solved. In the simplest multigrid scheme, one begins by solving a simplified problem, using a small basis of coarse functions. The result of this approximation is used as the starting guess for a refined problem defined with a finer grid of basis functions. This process is repeated on down to finer and finer levels until the result with desired resolution is obtained. In other multigrid schemes, the algorithm uses a more complicated pattern of subproblems at various resolutions. This is done so to avoid slow convergence. Multigrid is a successful and popular methodology, with entire conferences devoted to its details [12].

In contrast to the multigrid methods, the wavelet method used here attempts to solve a single problem, but the good conditioning is obtained by using a multiresolution basis. Various researchers have discussed using a wavelet basis to quickly solve differential equations [60, 47]. In the machine vision community, Szeliski has used a hierarchical basis to quickly solve surface reconstruction problems [71]. The basis he uses is contructed by starting with the $\phi_{L,j}$ functions defined as Terzopoulos' quadratic elements[74]. Then the linear pyramid of Yserentant [81] is used to define the $\phi_{i,j}$ and $\psi_{i,j}$ basis functions. In this construction, the functions on each level i are not scales of each other, and they have no simple closed form. As a result, one must a priori choose the level of resolution L, and no adaptivity is possible. Szeliski also explores the implicit matrix representation [81] to counteract the fact that the wavelets produce denser matrices. He experiments with both conjugate gradient and Gauss-Seidel iteration, and find conjugate gradient to require fewer iterations. This corresponds to what is generally known for iterative methods.

In [58] Pentland discusses using a wavelet basis for quickly solving variational problems [58]. (His method of using the diagonal terms from the wavelet matrix is discussed in Section 5.2). He uses an orthonormal wavelet transformation, without discussing scaling and also *assumes* the wavelet matrix to be sparse, and so does not pursue the implicit matrix representation. Pentland does not explore any adaptive oracle method.

In research independent of this author's, Yaou et al. use a semi-orthogonal linear B-spline wavelet for surface reconstruction [80]. This is an unusual choice for thin plate problems, since the second derivative measure is not very meaningful with bi-linear basis functions. Once again, the implicit method is not pursued, and thus each of the wavelet iterations are considerably more expensive than the with finite element basis. In their implementation a full wavelet basis is used, and no adaptive oracle is investigated. They also experiment with both Gauss-Seidel and conjugate gradient and surprisingly finds Gauss-Seidel to work better.

The system described in this thesis is closely related to these works. It applies conjugate gradient iterations to a properly scaled cubic B-spline wavelet system to obtain a method that uses few iterations, and can be adapted on the fly to the proper level of detail. The implicit method is used to exploit the sparseness of the B-spline finite element matrix. An oracle is used to adapt the size of the basis to the complexity of the particular solution. By using a basis derived from cubic B-splines, this method becomes compatible with other modeling tools based on cubic B-splines.

7.8 Implementation

The ideas outlined in the previous sections have been used to build an interactive variational modeling system (see Figures (42-46)). Some implementation details are described here.

7.8.1 Choice of Wavelet

Section 2.3 describes a number of wavelet constructions based on cubic B-splines. This implementation uses B-spline wavelets since the the resulting solutions are C^2. Moreover the result can be displayed with GL `nurbscurve` and `nurbssurface` calls [41], and interfaced with other computer graphics software that knows how to handle B-splines.

In Section 2.3, two wavelet constructions are discussed, a bi-orthogonal and a semi-orthogonal construction. The projections P_i arising from the semi-orthogonal construction are orthogonal, which is a natural way to create smoother versions of some curve. In the semi-orthogonal construction, the sequences $\tilde{h}$ and $\tilde{g}$ are infinite in length, and so the procedure `coef_xform_up` can only be made to run in linear time by solving a band-diagonal system. This has a high constant factor. In the bi-orthogonal construction, the sequences $\tilde{h}$ and $\tilde{g}$ are finite and so the procedure `coef_xform_up` can be simply implemented using convolutions (see Appendix A.1). For this reason, the implementation uses the bi-orthogonal construction.

In Section 2.3.2, two methods are described to construct a wavelet basis over an interval: mirror reflection, and using special boundary basis functions. This implementation uses mirror reflection because it is the simplest construction, but this does have its draw backs. Mirror reflection implies that the B-spline control points double back on each other (see Figure 9). This implies that the functions $X(t), Y(t), Z(t)$ must have zero derivative at the boundaries, and the geometric curve has zero curvature κ at the boundaries. The use of special boundary functions is worthy of further exploration.

For surfaces the non-standard bivariate basis is used.

These images show a user sculpting a surface interactively using constraints. The user begins with a surface pinned down at four corners.

Figure 42: The user then pulls down one of the corners.

The user adds a new interpolation constraint. The constraint is colored pink because the delta information is valid for that constraint.

Figure 43: Using the valid delta information, that constraint can be manipulated without resolving the system of equations.

The user adds a new tangent and interpolation constraint.

Figure 44: The user changes the tangent orientation slightly.

Later the user has added more interpolation and tangent constraints.

Figure 45: The user manipulates one of the interpolation/tangent constraints.

The pink constraints can be manipulated quickly using the delta information.

Figure 46: Here the user twists one of the tangent constraints.

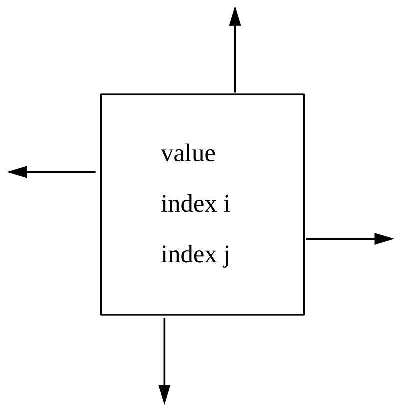


Figure 47: A matrix entry

7.8.2 Sparse Matrix

The speed of the algorithm depends on being able to quickly manipulate a sparse matrix. The required operations are

- Multiply a vector with the matrix.
- Multiply a vector with the transpose of the matrix.
- Add an arbitrary entry to the matrix (when the oracle activates new basis functions).
- Delete a row or column of the matrix (when the oracle deactivates a basis function).

These matrices do not have a simple structure (such as band diagonal), and so a general sparse representation is used. In the implementation the matrix is represented using one *node* to represent each non-zero entry. Each node contains a value as well as an i, j index (see Figure 47). Each row of the matrix is represented as a doubly linked list of nodes. Each column is also represented as a doubly linked list of nodes (see Figure 48). Vectors are stored densely in an array, so they can be accessed randomly in constant time.

A matrix vector multiply is implemented by walking down each of the rows i and multiplying the node value by the value in the proper vector element j. Transpose

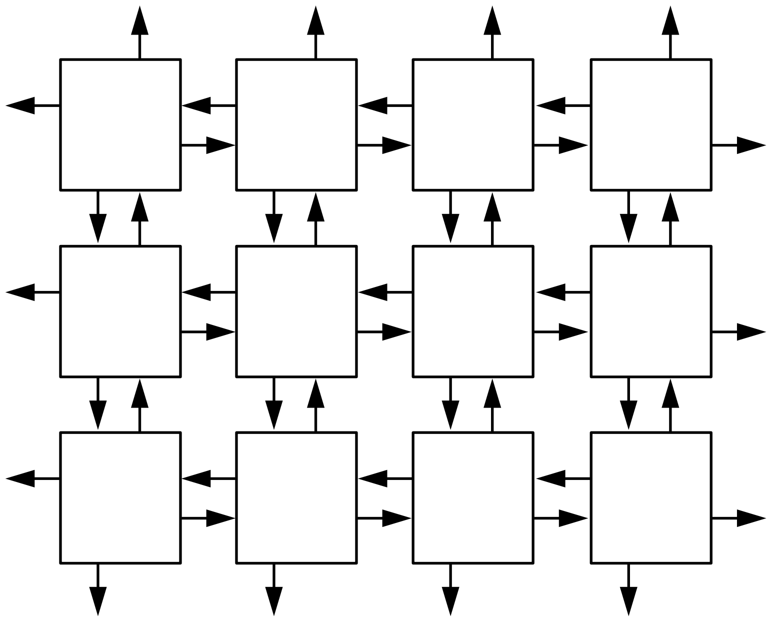


Figure 48: The rows and columns are made up of doubly linked lists.

matrix multiplies are handled similarly. Deleting a row is done by walking down it, and deallocating each node. When a node of some row is deallocated the two nodes in the same column as that node, above and below it, must have their pointers fixed up so that they point to each other and not the deleted node. Deleting a column is done similarly. Adding an arbitrary i, j entry is simply done by placing it at the head of the i row linked list, and the head of the j column linked list [7].

For efficient implementations, Unix `malloc` and `free` system calls are not used for individual matrix nodes. Instead large chunks of memory are taken when necessary from the system, and doled out internally.

In this implementation vectors are stored densely in an array, but many of the entries may represent deactivated basis functions are need not be used (they are zero by definition). This knowledge can be used to add two vectors, or perform the dot product of two vectors more quickly. To achieve this, the number `n` of active entries

[7]For the operations required here, there is no need for the linked lists to have any internal sorted order since an individual entry of a row or column is never sought. Matrix multiplies, and row, or column deletions traverse entire linked lists. And so new entries can always be put at the head of the lists.

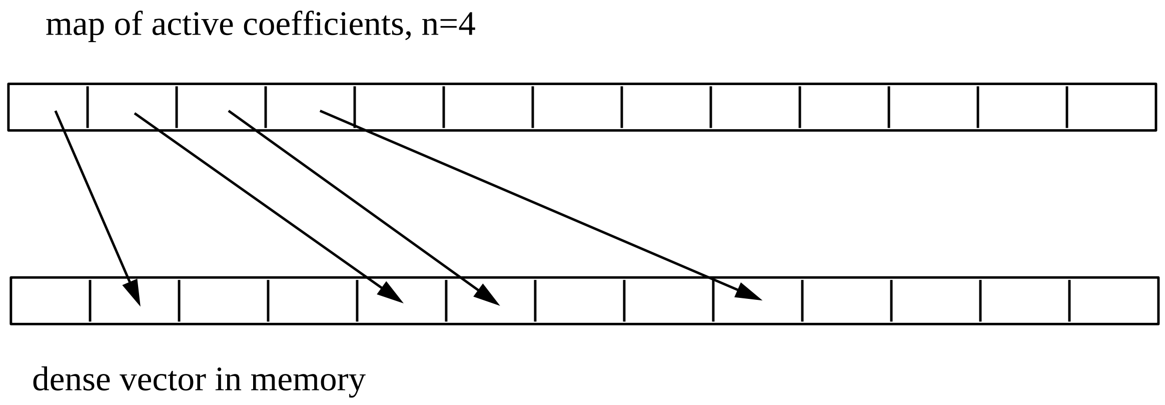


Figure 49: Vectors are stored densely in memory. A separate map structure points to the active entries.

is maintained along with a `map` array associated with the vector. The first `n` entries of the `map` hold the indices of the active entries in the vector. To perform a vector operation (such as an add or dot product) the `map` is traversed instead of the entire vector, and thus no computation is performed for non active basis functions (see Figure 49).

7.8.3 Computing Matrix Terms

The Hessian matrix contains second derivative of the energy function with respect to the basis functions. For thin plate curves, the Hessian matrix has terms

$$\int dt\ \ddot{\psi}_{i,j}(t)\ddot{\psi}_{k,l}(t) \tag{163}$$

or if the implicit method is being used

$$\int dt\ \ddot{\phi}_{i,j}(t)\ddot{\phi}_{k,l}(t) \tag{164}$$

For thin plate surfaces, the Hessian matrix term relating the two non-standard basis functions $\psi_{i,j}(s)\psi_{i,l}(t)$ and $\psi_{m,n}(s)\psi_{m,p}(t)$ is

$$\begin{aligned}\int ds\int dt\ & D_{ss}(\psi_{i,j}(s)\psi_{i,l}(t)) * D_{ss}(\psi_{m,n}(s)\psi_{m,p}(t)) \\ + \quad 2\int ds\int dt\ & D_{st}(\psi_{i,j}(s)\psi_{i,l}(t)) * D_{st}(\psi_{m,n}(s)\psi_{m,p}(t)) \\ + \quad \int ds\int dt\ & D_{tt}(\psi_{i,j}(s)\psi_{i,l}(t)) * D_{tt}(\psi_{m,n}(s)\psi_{m,p}(t))\end{aligned} \tag{165}$$

which can be rearranged and expressed with respect to univariate basis functions

$$\begin{aligned}
&\int ds\; \ddot{\psi}_{i,j}(s)\ddot{\psi}_{m,n}(s) * \int dt\; \psi_{i,l}(t)\psi_{m,p}(t) \\
&+\; 2\int ds\; \dot{\psi}_{i,j}(s)\dot{\psi}_{m,n}(s) * \int dt\; \dot{\psi}_{i,l}(t)\dot{\psi}_{m,p}(t) \\
&+\; \int ds\; \psi_{i,j}(s)\psi_{m,n}(s) * \int dt\; \ddot{\psi}_{i,l}(t)\ddot{\psi}_{m,p}(t)
\end{aligned} \tag{166}$$

If the implicit method is used, then all of the basis functions in the above expression are B-spline basis functions $\phi_{i,j}$.

Each of the univariate basis functions ψ is piecewise cubic (with 14 pieces) and the univariate basis functions ϕ are also piecewise cubic (with 4 pieces). The functions $\dot{\psi}$ and $\dot{\phi}$ are piecewise quadratic, and $\ddot{\psi}$ and $\ddot{\phi}$ are piecewise linear. Computing the integrals thus requires integrating each of the pieces separately using symbolic anti-differentiation, and summing up the total. At the boundary of the domain, one must use the polynomial form of the special boundary basis functions, or mirror reflected basis functions.

7.8.4 The Main Loop

A user of the system is first presented with a default curve or surface. Constraints can then be introduced by clicking on the curve or surface with the mouse. The location of the mouse click defines a parametric position t (and s) on the curve (or surface). The user can then drag this point to a new location to define an interpolation constraint. Tangent constraints at a point can also be defined by orienting "arrow" icons at the point. Once the constraint is set, the solver is called to compute the minimum energy solution that satisfies the constraints placed so far. The system uses thin plate energy to measure the smoothness of the solution. Resulting curves and surfaces are displayed by transforming the wavelet representation back to a uniform cubic B-spline representation at the finest level of detail, and then making a SGI GL `nurbscurve` or `nurbssurface` call [41] [8].

The solver sets up the appropriate linear system (Equation (161/162)). Initially the solver assumes that very few degrees of freedom are needed, thus only a few

[8]One GL call to `nurbssurface` can be more expensive than a complete iteration.

variables $\bar{\mathbf{x}}$ are active, corresponding to the basis functions on wavelet level 0 (the coarsest level). The matrices are represented sparsely, so the solver only needs storage for the non-zero entries, and the cost of a matrix-vector multiply scales with the number of non-zero entries. This linear system is then solved by using conjugate gradient iterations [59]. After every k iterations (25 is the default), the `oracle` is called. As explained in Section 7.5, the oracle decides which new wavelet basis functions and their corresponding variables $\bar{x}$ should be activated, and which ones can be deactivated.

At this point the matrices must be updated. If the explicit approach is being used (Equation (161)), then the bookkeeping is very simple. When activating a new wavelet basis function all that must be done is to add in a row and column to $\bar{\mathbf{H}}$, and a column to $\bar{\mathbf{A}}$. These matrix terms are computed on the fly. Because the basis functions are piecewise polynomial, multiplication, derivation with respect to s, t, and integration can all be performed symbolically. This is summarized in the following pseudocode [9]

```
activateWavelet(i,j)
   forall( active wavelets with support overlapping (i,j) )
      (k,l) = index of an overlapping wavelet ;
      val = computeHessianTerm( (i,j) , (k,l) );
      sparseMatrixInsert(val, matrix= Hess, row=(i,j) , col=(k,l) );
      sparseMatrixInsert(val, matrix= Hess, row=(k,l) , col=(i,j) );
   forall (constraints under the support of (i,j) )
      (k) = index of an overlapped constraint ;
      val = computeConstraintTerm( (i,j) ,k);
      sparseMatrixInsert(val, matrix= Constraint, row=k, col=(i,j) );
```

When deactivating a wavelet basis function, all that is done is to delete a column

[9]For simplicity, the basis functions will be indexed with two indices. For surfaces, the bivariate non-standard wavelet basis is used, and so the indexing is more complicated.

(and row) from the matrices. If there is sufficient memory on the computer, then the discarded values are cached; if the `oracle` later reactivates those basis function, then the solver does not need to symbolically recompute those matrix entries.

```
deactivateWavelet(i,j)
   sparseMatrixDeleteRow(matrix= Hess, row=(i,j) );
   sparseMatrixDeleteCol(matrix= Hess, col=(i,j) );
   sparseMatrixDeleteCol(matrix= Constraint, col=(i,j) );
```

When the implicit approach is used (Equation (162)), the bookkeeping becomes a bit more complicated. When very few wavelet functions are active, it is not necessary or efficient to compute the entire B-spline matrices **H** and **A**, nor is it necessary to pay the cost of the complete `pyrm` procedures. Instead the following is done. When the `oracle` activates or deactivates a wavelet, this just updates a list of active wavelets.

```
activateWavelet(i,j)
   add (i,j) to list of active wavelets;
```

```
deactivateWavelet(i,j)
   remove (i,j) from list of active wavelets;
```

After the `oracle` is finished, a *minimal* B-spline basis that spans the space of the active wavelets is computed using the procedure `computeActiveBSplines`. Then a B-spline constraint and hessian matrix are updated. This can be described as the following:

```
activateAndDeactivate(ActiveWavelets, oldActiveBSplines)
   newActiveBsplines = computeActiveBSplines(ActiveWavelets);
   forall(newActiveBSplines - oldActiveBSplines)
      (i,j) = index of a new BSpline;
      activateBSpline(i,j);
   forall(oldActiveBSplines - newActiveBSplines)
      (i,j) = index of a old BSpline;
      deactivateBSpline(i,j);
```

The procedures `activateBSpline` and `deactivateBspline` update the B-spline constraint and Hessian matrix.

```
activateBSpline(i,j)
   forall( active BSplines with support overlapping (i,j) )
      (k,l) = index of an overlapping BSpline ;
      val = computeHessianTerm( (i,j) , (k,l) );
      sparseMatrixInsert(val, matrix= Hess row=(i,j) , col=(k,l) );
      sparseMatrixInsert(val, matrix= Hess row=(k,l) , col=(i,j) );
   forall (constraints under the support of (i,j) )
      (k) = index of an overlapped constraint ;
      val = computeConstraintTerm( (i,j) ,k);
      sparseMatrixInsert(val, matrix= Constraint, row=k, col=(i,j) );
```

```
deactivateBSpline(i,j)
   sparseMatrixDeleteRow(matrix= Hess, row=(i,j) );
   sparseMatrixDeleteCol(matrix= Hess, col=(i,j) );
   sparseMatrixDeleteCol(matrix= Constraint, col=(i,j) );
```

This adaptive approach is possible with the wavelets described in this thesis, but was not possible with the hierarchical construction in [71]. This is because an adaptive method must be able to compute new Hessian and constraint matrix terms on the fly, and therefore one needs an explicit expression for the hierarchical basis functions. When wavelets or hierarchical basis functions are only defined by some complete pyramid transformation, without a closed symbolic form for the individual basis functions, then one must always use the complete hierarchical basis. The $\phi_{i,j}$ and $\psi_{i,j}$ functions described in this thesis have a piecewise polynomial form, making the adaptive method possible.

The conjugate gradient iterations are continued until the residual is smaller than some tolerance.

7.8.5 Delta Constraints

When the solution is completed, the result provides information for not only the curve or surface satisfying the specific value of the new constraint, but for curves or surfaces with respect to any value of this constraint. Once the linear system (Equation (161/162)) with the newest constraint has been solved, the solver stores the delta vector

$$\frac{\mathbf{\Delta \bar{x}}}{\Delta b_m} \tag{167}$$

where m is the index of the newest constraint, and b_m is the constraint value (i.e., the position or tangent specified by the user). This vector stores the change of the coefficient vector due to a unit change in the new constraint Δb_m, essentially a column of the inverse matrix. The user is now free to interactively move the target location of the constraint without having to resolve the system since, as long as the parameters s, and t of the constraints do not change, the matrix of the system, and thus its inverse, do not change. However, as soon as a new constraint is added (or a change to the parameters s and t is made) there is fresh linear system that must be solved, and all of the delta vectors are invalidated. The ability to interactively change the value of a constraint is indicated to the user by coloring the constraint icon.

The application of this process to an interpolation constraint for the univariate function, X, is shown in the following pseudocode. `m` is the constraint identifier. t_m is the parameter value for the constraint (a surface would have two parameters (s_m, t_m)). `newB` is the new requested interpolation point.

```
manipulate(m, $t_m$, newB)
   if (oldConstraint (m) and validDelta(m) )
      oldB = $X(t_m)$ ; /* evaluate the function at $t_m$ */
      changeB = newB - oldB;
      $\bar{x}$ += changeB * deltaVec[m];
      return;
   if (new(m))
      invalidate all deltas;
   /* solve, and set new delta */
   oldXvec = $\bar{x}$;
   oldB = $X(t_m)$ ;
   $\bar{x}$ = conjugateGradiantSolver();
   $\mathbf{\Delta}\bar{x}$ = $\bar{x}$ - oldXvec;
   $\Delta b_m$ = newB - oldB;
   deltaVec[m] = $\frac{\mathbf{\Delta}\bar{\mathbf{x}}}{\mathbf{\Delta b_m}}$ ;
   validate(deltaVec[m]);
```

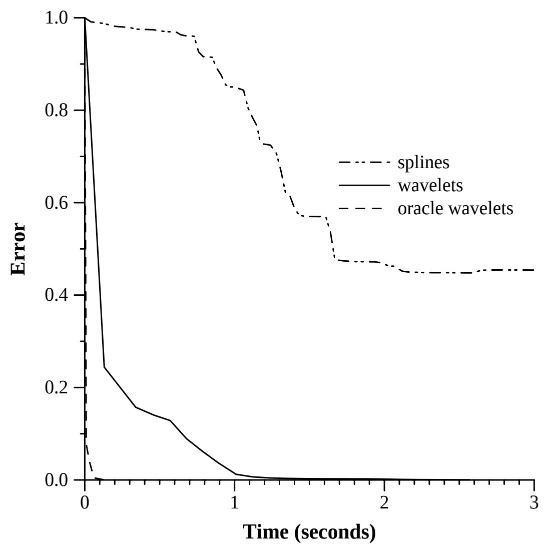


Figure 50: Error per time. Curve with 65 control points, 3 constraints.

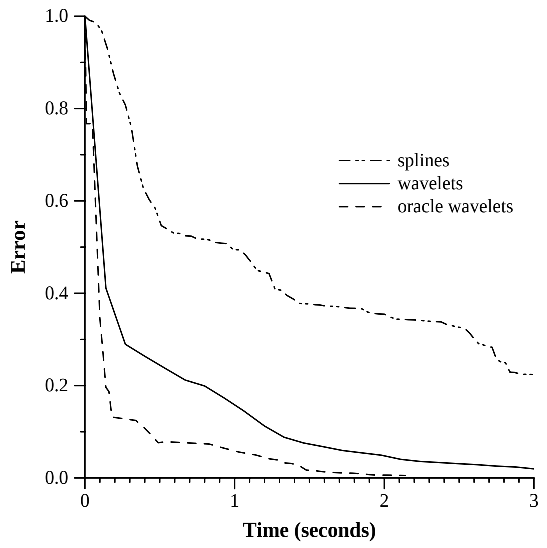


Figure 51: Error per time. Curve with 65 control points, 7 constraints.

7.9 Results

A series of experiments were conducted to examine the performance of the wavelet based system compared to a B-spline basis. In the 2D experiments, the number of levels of the hierarchy, L, was fixed to 6, and thus there were 65 B-spline, or wavelet functions in the complete basis for $x(t)$ (and another 65 basis functions for y as well

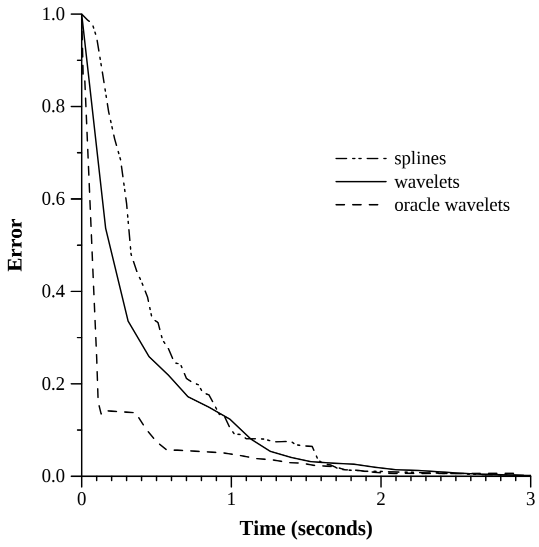


Figure 52: Error per time. Curve with 65 control points, 13 constraints.

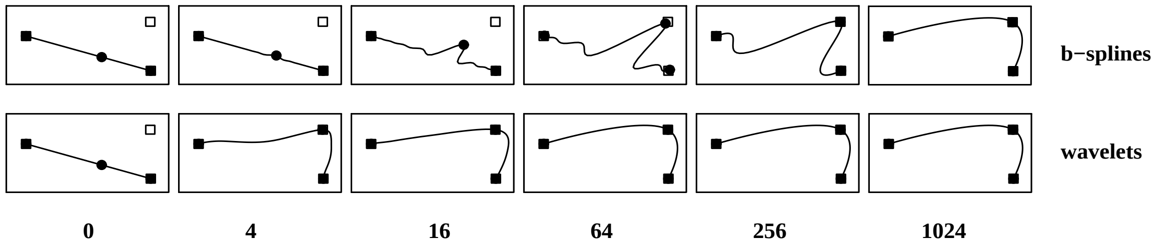


Figure 53: Solutions found by B-spline and wavelet methods after various numbers (0-1024) of iterations. There are 65 variables. This sequence shows the iterations after the third constraint (open square) has been added. This illustrates the ill conditioning of the B-spline optimization problem.

as z). The optimization process was then run on problems with various amounts of constraints. The results of these tests are shown in Figures (50-52). These graphs show the convergence behavior of three different methods, solving with the complete B-spline basis, solving with the complete wavelet basis, and solving with an adaptive wavelet basis that uses an oracle. (The wavelet results shown here use the *implicit* implementation). If $x^{(m)}$ is the computed solution expressed as B-spline coefficients at time m, and x^* is the correct solution of the complete linear system [10] (i.e., the complete system with $2^L + 1$ variables, and no adaptive oracle being used) then the error at time m is defined as

$$\frac{\sum_j \mid x_j^* - x_j^{(m)} \mid}{\sum_j \mid x_j^* - x_j^{(0)} \mid} \tag{168}$$

To obtain the starting condition $x^{(0)}$, two constraints were initialized at the ends of the curve, and the minimal thin plate solution (which in this case is a straight line) was computed. (For surfaces, the four corners were constrained.) All times were taken from runs on an Silicon Graphics R4000 Reality Engine running at 50 megahertz.

When there are a small number of constraints, and the solution is thus very under constrained, the B-spline method is very poorly conditioned, and converges quite slowly while the wavelet method converges dramatically faster (this is illustrated in Figure (53)). Since the optimization problem is very underconstrained, the oracle decides that it needs only a very small active set of wavelets and so the adaptive method converges even faster. As the number of constraints is increased, the solution becomes more tightly constrained, and the condition of the B-spline system improves. (Just by satisfying the constraints, the B-spline solution is very close to minimal energy). Meanwhile the oracle requires a larger active set of wavelets. Eventually, when enough constraints are present, the wavelet methods no longer offer an advantage over B-splines. While it still requires fewer iterations to converge, it must pay more per iteration.

How much is this extra cost? Experiments with both the explicit and implicit wavelet methods showed that for curves with a complete basis (no oracle), and $L = 6$,

[10]computed numerically to high accuracy

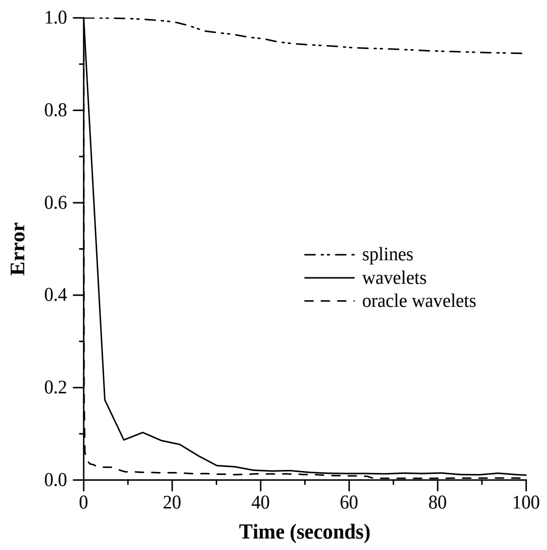


Figure 54: Error per time. Surface with 1089 control points, 5 constraints.

one iteration of the explicit or the implicit wavelet method was slower than one B-spline iteration by about a factor of 3 [11]. Thus, for problems of this size, the explicit and the implicit methods cost about the same. For the implicit method, this factor of 3 is the extra constant factor incurred by using the `pyrm` procedures and is independent of L. For the explicit method, this factor of 5 was a result of the denser wavelet matrices $\bar{H}$ and $\bar{A}$, which are only $O(n \lg n)$ sparse, and so this factor will increase as L increases.

As expected, for surfaces everything becomes more expensive (because of the denser surface matrices), except for the `pyrm` procedures. And so for surfaces, a single iteration of the implicit wavelet method is only slower than a single iteration of the B-spline method by about a factor of 1.3, regardless of L [12]. Meanwhile one iteration of the explicit wavelet method for $L = 5$, where the complete basis has 1089 functions (33 in s, times 33 in t), is slower than one B-spline iteration by about a factor of 10.

In the surface experiments, L was set to 5. (There are 1089 basis functions for the

[11]Each B-spline iteration took 0.0035 seconds while each iteration using a complete wavelet basis took 0.011 seconds.

[12]For $L = 5$ each iteration using B-splines took 0.68 seconds while each iteration using the complete wavelet basis took 0.85 seconds.

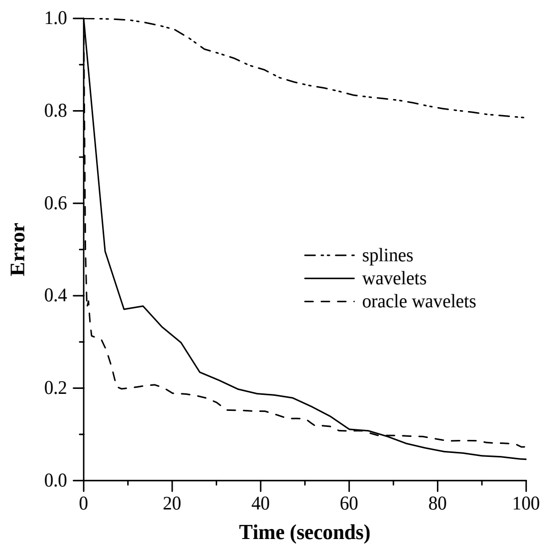


Figure 55: Error per time. Surface with 1089 control points, 11 constraints.

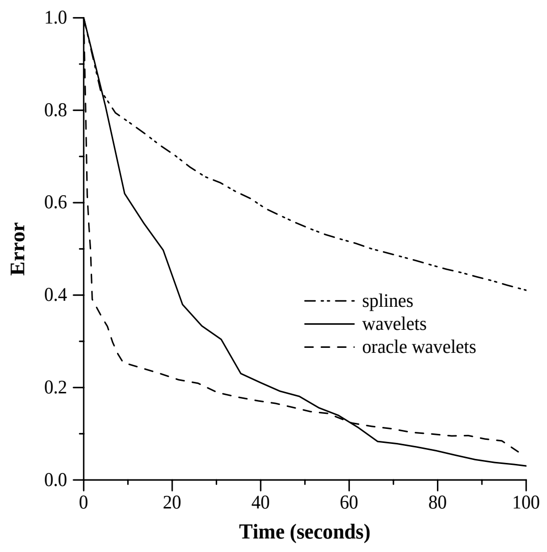


Figure 56: Error per time. Surface with 1089 control points, 23 constraints.

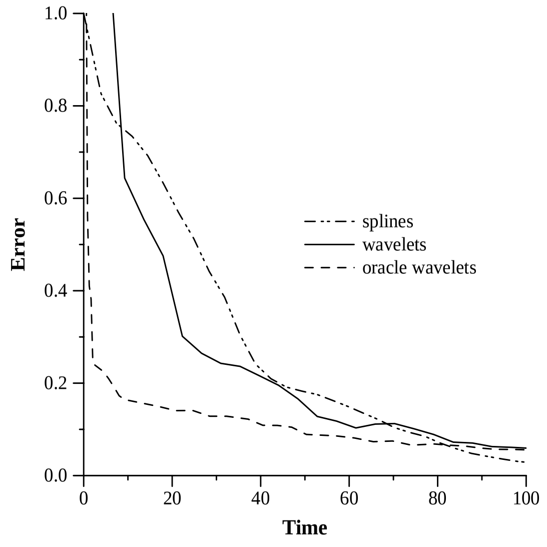


Figure 57: Error per time. Surface with 1089 control points, 36 constraints.

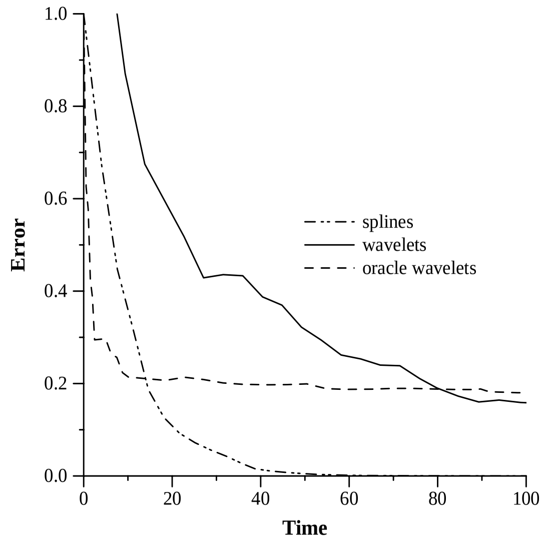


Figure 58: Error per time. Surface with 1089 control points, 64 constraints.

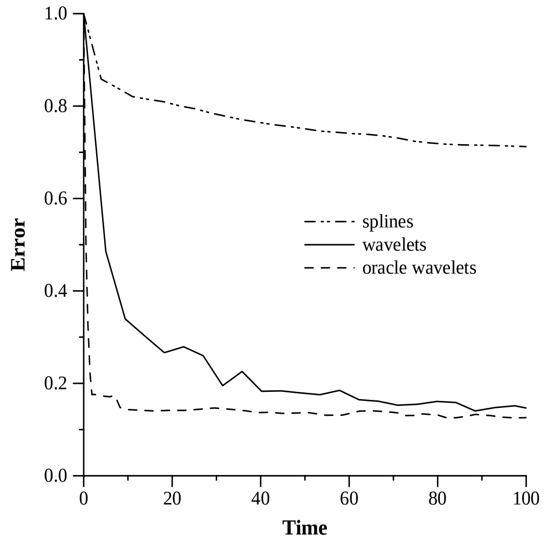


Figure 59: Error per time. Surface with 1089 control points, 62 constraints. along the boundary.

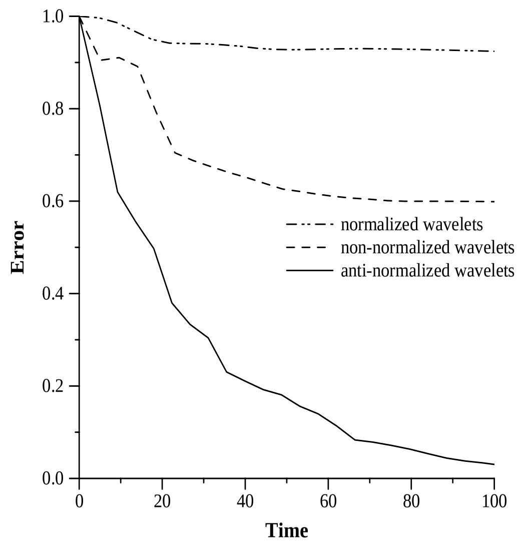


Figure 60: Error per time. Surface with 1089 control points, 23 constraints. Traditionally normalized wavelets perform much worse than non-normalized. The anti-normalized wavelets perform the best.

x coordinates, and another 1089 for y as well as z). A set of experiments with evenly spaced constraints was run. The experimental results are shown in Figures (54-58). With few constraints, the wavelet methods greatly outperform the B-spline method. As more constraints are added, the B-spline method becomes better behaved, and eventually converges faster than the wavelets.

Experiments were also run where all the constraints were along the boundary of the surface. In these experiments there are many constraints, but the since the constraints are along the boundary, much of the surface is "distant" from any constraint. In these problems, the wavelets also performed much better than the B-spline method, (see Figure 59).

These experiments imply that wavelets are best suited for problems where there are large gaps between the constraints.

To demonstrate the importance of the scaling used for the wavelet basis functions, the experiment using 23 constraints was rerun using wavelets with different scaling factors between the levels: Traditionally normalized wavelets

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(i-L)/2}\ \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{(i-L)/2}\ \psi(2^{(i-L)}t - j) \end{aligned} \tag{169}$$

un-normalized wavelets

$$\begin{aligned} \phi_{i,j}(t) &= \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= \psi(2^{(i-L)}t - j) \end{aligned} \tag{170}$$

and the "anti-normalized" wavelets that have been used in the rest of the experiments thus far

$$\begin{aligned} \phi_{i,j}(t) &= 2^{-(i-L)}\ \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{-(i-L)}\ \psi(2^{(i-L)}t - j) \end{aligned} \tag{171}$$

The results are shown in Figure 60. Clearly anti-normalization is essential for the wavelet optimization method studied in this chapter.

7.10 Discussion

The ability to directly manipulate curves and surfaces provides an intuitive interface to the user for design purposes. Direct manipulation of a surface, however, leads to an underconstrained problem since there are, in general, many possible surfaces through a given point or set of points. Finding the "best" solution requires solving a variational problem.

This chapter has presented a methodology to efficiently solve such problems based on a hierarchical representation of the curves and surfaces. In particular, a wavelet basis is used to accelerate the computation of optimal curves and surfaces based on a thin plate functional. Implementation and a system developed with these ideas are presented and experimental results are discussed that demonstrate the advantages over a pure B-spline based approach.

Another application for the adaptive oracle is data reduction. The size of the active set of wavelet coefficients is often much smaller than the number of B-spline control points in the computed solution. For example, in the curve experiments, the size of the final active sets for 3, 7, and 13 constraints respectively were 8, 21, and 29, as compared to the 65 B-spline control points. Similarly, in the surface experiments, the size of the final active sets for 5, 11, 23, 36,and 64 constraints respectively were 82, 150, 350, 218 and 239 as compared to the 1089 B-spline control points.

Future work will be required to explore the use of higher order functionals like those given in [57, 62]. Because the optimization problems resulting from those functionals are non-linear, they are much more computationally expensive, and it is even more important to find efficient methods. It is also important to study optimization modeling methods where constraint changes only have local effects. Finally, it would be desirable to extend the multiresolution framework beyond tensor product bases in order to model surfaces with arbitrary topology [43]. Presently, the only way to model an arbitrary structure is to piece together patches, and constrain the boundaries to have the necessary continuity. In addition, improvements to the *oracle* function that determines the appropriate local resolution may lead to more efficient results.

Chapter 8

Conclusion

This thesis has discussed the use of a wavelet basis for two computer graphics applications radiosity and geometric modeling.

In radiosity, the compression properties of the wavelet basis is exploited, to obtain a sparse matrix. The radiosity matrix contains coefficients of the kernel function, a function that represents the energy interaction between pairs of surface points in the environment. Because many regions of this kernel function are smooth, the corresponding coefficients in the wavelet basis are negligible. In order to efficiently enumerate the significant coefficients, a top down recursive procedure `ProjectKernel` is presented. This procedure makes use of a `oracle` function that measures the smoothness of the kernel. By using different wavelets, with better compression properties, more efficient radiosity solutions can be obtained. This wavelet methodology also allows for a more formal understanding of the hierarchical radiosity method of [44].

Some implementation issues have been discussed and experimental results have been reported. In particular, experiments with two simple 2D environments have shown that wavelets with 2 vanishing moments lead to sparser radiosity matrices than the Haar basis. A variety of "tree" wavelets have been implemented in a working 3D radiosity system. Numerical experiments have shown how raising the number of vanishing moments leads to a sparser matrix. User experience has been best with the $\mathcal{M}_2$ basis (i.e., 2 vanishing moments).

This implementation only used tree wavelets, leading to possible discontinuities in the reconstructed solution. These discontinuities perhaps offer too much freedom in the representation. This freedom causes the number of basis functions to grow quickly as the number of vanishing moments is raised. Also the discontinuities can cause artifacts in the resulting images. It is important to study families of smooth wavelets, and understand how they can be used in a radiosity implementation.

In radiosity algorithms, much of the subdivision is used to resolve discontinuities in the illumination function, such as shadow boundaries. Recently, Lischinski et al. [51] have discussed combining hierarchical radiosity with discontinuity meshing. In this algorithm, the mesh is not simply refined by quad-tree subdivision, but rather along the discontinuity lines which are computed a-priori by analyzing the geometry of the environment. This method allows the radiosity algorithm to capture the shadow patterns with fewer mesh elements. It would be interesting to study how the general wavelet method could be combined with such a discontinuity meshing strategy.

In hierarchical radiosity algorithms, the algorithm begins with the initial input geometry (typically polygons), and then subdivides them as necessary. The hierarchical nature of the wavelet method performs this subdivision in an efficient manner, avoiding the computation of unnecessary detail. But, if there are k initial geometric objects input to the program, the interactions between all pairs of these objects must be computed. This is a cost of $O(k^2)$. While this cost is negligible for simple environments, it becomes overwhelming for complicated environments described by many small geometric objects (such a football stadium with $100,000$ seats). A method of *clustering* is needed that avoids computing the interactions between each pair of tiny input objects. A number of solutions have recently been discussed [68, 70], but more research is necessary.

The second computer graphics application this thesis has discussed is geometric modeling (in particular, curve and surface modeling). In this context a wavelet basis based on uniform cubic B-splines has been used. The wavelet representation allows the user to interact with the curve or surface at any resolution level desired. This provides the useful ability to manipulate the object in both broad and fine ways. The wavelet transformation also allows an optimization program the ability to alter the solution at any necessary resolution. This results in a efficient variational modeling tool that can find the "smoothest" curve or surface meeting some set of constraints efficiently with few iterations. In this context, an `oracle` function is used to determine where the fine detail is needed in the solution, and where it is unnecessary.

This implemented variational modeling tool only used the thin plate measure of smoothness. There are other measures that may be employed, which may yield more desirable surfaces. Many of these measures are highly non-linear, requiring very time consuming optimization methods. Thus, in these cases, it is even more crucial to explore methods to speed up this process.

The wavelet representation used in this thesis for geometric modeling has a number of limitations. In the cubic B-spline wavelet basis, the resulting curves and surfaces are always C^2. While this is desirable in general, at times one would like to introduce discontinuities at specific places, in order to model sharp edges. It is important to learn how this can be done within the context of a hierarchical representation. Secondly, the tensor product B-spline used in this thesis can only model surfaces that have a square parametric domain. This is certainly a limitation when trying to model arbitrary objects. Recently researchers have been exploring very flexible representation methods for objects with arbitrary topology [43]. A hierarchical representation bases on such methods has also been studied [53]. This hierarchical representation offers a flexible way to extend the methods described in this thesis to more general surface topologies.

Numerical optimization is a general paradigm that can be applied to many problem domains. One such example is the control problems that arise in computer animation [19]. In such a problem the animator wishes to create an animation sequence for some creature without specifying the exact trajectory that the creature will take. In this context, a costly non-linear minimization problem must be solved. Research in this area has shown that wavelet methods, like those described in this thesis, can help to efficiently produce solutions [52]. It is important to see how these wavelet optimization methods can be applied to other areas as well.

Appendix A

A.1 Bi-orthogonal B-spline Wavelet Transform

This appendix provides pseudo code for the bi-orthogonal wavelet transform. The convolution sequences are given in [17].

The `bs_index` procedure returns an index into the array of B-spline coefficients $bs[-2 \ldots 2^L 2]$ using mirror reflection The procedure first reflects around 0, and then periodizes.

```
int bs_index($L$, $j$)
   $range$ = $2^L$ ;
   $j$ = $j$ + 2 ;
   $j$ = absolute_value($j$) ;
   $j$ = $j$ % ($2 * range$) ;  /* "%" == "mod" */
   if ($j \leq range$)
      return($j$ - 2) ;
   else
      return($2 * range$ - $j$ - 2);
```

This `wave_index` procedure indexes into the array of mirrored wavelet coefficients $wave[-3 \ldots 2^L - 4]$.

```
int wave_index($L$, $j$)
    $range$ = $2^L$ ;
    $j$ = $j$ + 3 ;
    $j$ = $j$ + .5 ;
    $j$ = absolute_value($j$) ;
    $j$ = $j$ - .5 ;
    $j$ = ( $j$ % $(2 * range)$) ;
    if ($j \le range - 1$)
        return($j$ -3) ;
    else
        return($2 * range$ - 1 - $j$ - 3) ;
```

Coefficient transformations are implemented with the procedures `coef_xform_up` and `coef_xform_down`. The first procedure implements Equation (27). (The sequences h,g,$\tilde{h}$, and $\tilde{g}$ are zero outside of their defined ranges).
$h[0..4] = \{\frac{1}{8}, \frac{4}{8}, \frac{6}{8}, \frac{4}{8}, \frac{1}{8}\}$;
$g[0..10] = \{\frac{5}{256}, \frac{20}{256}, \frac{1}{256}, \frac{-96}{256}, \frac{-70}{256}, \frac{280}{256}, \frac{-70}{256}, \frac{-96}{256} \frac{1}{256}, \frac{20}{256}, \frac{5}{256}\}$;
$\tilde{h}[-3..7] = \{\frac{-5}{256}, \frac{20}{256}, \frac{-1}{256}, \frac{-96}{256}, \frac{70}{256}, \frac{280}{256}, \frac{70}{256}, \frac{-96}{256} \frac{-1}{256}, \frac{20}{256}, \frac{-5}{256}\}$;
$\tilde{g}[3..7] = \{\frac{1}{8}, \frac{-4}{8}, \frac{6}{8}, \frac{-4}{8}, \frac{1}{8}\}$;

```
coef_xform_up( $bs_{\text{in}}$[], $bs_{\text{out}}$[], $w_{\text{out}}$[], $L$)
    $range$ = $2^{L-1}$ ;
    $bs_{\text{out}} = w_{\text{out}}$ = 0 ; /* zero vectors */
    for ($j = -2$ ; $j \leq range2$ ; $j{+}{+}$)
        $wdx$ = wave_index($L-1$, $j$) ;
        $bdx$ = bs_index($L-1$, $j$) ;
        for ($k = (2j-3)$; $k \leq (2j+7)$ ; $k{+}{+}$)
            $idx$ = bs_index($L$, $k$) ;
            $bs_{\text{out}}$[$bdx$]    += $\tilde{h}[k-2j] * bs_{\text{in}}[idx]$ ;
            $w_{\text{out}}$[$wdx$]    += $\tilde{g}[k-2j] * bs_{\text{in}}[idx]$ ;
```

The inverse transformation which uses Equation (29) is implemented in the following procedure

```
coef_xform_down( $bs_{\text{in}}$[], $w_{\text{in}}$[], $bs_{\text{out}}$[], $L$ )
    $range$ = $2^{L}$ ;
    $bs_{\text{out}}$ = 0 ; /* zero vector */
    for($j = -2$; $j \leq range - 2$; $j{+}{+}$)
        $odx$ = bs_index($L$, $j$) ;
        for($k = \frac{j-10}{2}$; $k \leq \frac{j}{2}$; $k{+}{+}$)
            $bdx$ = bs_index($L-1$, $k$) ;
            $wdx$ = wave_index($L-1$, $k$) ;
            $bs_{\text{out}}[odx]$ += $h[j-2k] * bs_{\text{in}}[bdx]$
                      + $g[j-2k] * w_{\text{in}}[wdx]$ ;
```

The following procedure transforms the basis functions themselves (see Equations 15 and 16). Because the basis functions centered exactly on the boundaries do not have a nearby mirrored copy, they must be scaled slightly differently.

```
basis_xform_up( $bs_{\text{in}}$[] , $bs_{\text{out}}[]$ , $w_{\text{out}}[]$ , $L$ )
    $range$ = $2^{L-1}$ ;
    $downRange$ = $2^{L}$ ;
    $bs_{\text{in}}[0] *= 2.0$;     $bs_{\text{in}}[downRange] *= 2.0$ ;
    $bs_{\text{out}} = w_{\text{out}}$ = 0 ; /* zero vectors */
    for ($j = -2$; $j \le range - 2$; $j{+}{+}$)
        $wdx$ = wave_index($L-1$, $j$) ;
        $bdx$ = bs_index($L-1$, $j$) ;
        for ($k = (2j)$; $k \le (2j+10)$; $k{+}{+}$)
            $idx$ = bs_index($L$, $k$) ;
            $bs_{\text{out}}[bdx]$       += $h[k-2j] * bs_{\text{in}}[idx]$ ;
            $wave_{\text{out}}[wdx]$ += $g[k-2j] * bs_{\text{in}}[idx]$ ;
    $bs_{\text{out}}[0] *= 0.5$ ;     $bs_{\text{out}}[range] *= 0.5$ ;
```

And the inverse transformation which implements Equation 19

```
basis_xform_down( $bs_{\text{in}}[]$, $wave_{\text{in}}$[], $bs_{\text{out}}$[], $L$ )
    $range$ = $2^L$ ;
    $upRange$ = $2^{L-1}$ ;
    $bs_{\text{in}}[0] *= 2.0$;    $bs_{\text{in}}[upRange] *= 2.0$ ;
    $bs_{\text{out}}$ = 0 ; /* zero vector */
    for($j = -2$; $j \leq range - 2$; $j{+}{+}$)
        $odx$ = bs_index($L$, $j$) ;
        for($k = \frac{j-7}{2}$; $k \leq \frac{j+3}{2}$; $k{+}{+}$)
            $bdx$ = bs_index($L-1$, $k$) ;
            $wdx$ = wave_index($L-1$, $k$) ;
            $bs_{\text{out}}[odx]$ += $\tilde{h}[j-2k] * bs_{\text{in}}[bdx]$
                        + $\tilde{g}[j-2k] * w_{\text{in}}[wdx]$ ;
    $bs_{\text{out}}[0] *= 0.5$ ;    $bs_{\text{out}}[range] *= 0.5$ ;
```

A.2 Semi-orthogonal B-spline Wavelet Transform

This appendix provides pseudo code for semi-orthogonal B-spline wavelet transform. The sequence h and the sequence g are the convolution sequences used to construct the inner basis functions [16], while the vectors $\underline{h}_j$ and the vectors $\underline{g}_j$ are used to construct the boundary basis functions [15, 61]. Only the vectors for the left boundary are given, the vectors for the right boundaries are the mirror images of these vectors. It is assumed that the boundary B-spline basis functions are those that arise by placing quadruple knots at the boundaries.

$$
\begin{aligned}
h[0..4] &= \tfrac{1}{8} * \{1, 4, 6, 4, 1\} \\
g[0..10] &= \tfrac{1}{8!} * \{-1, 124, -1677, 7904, -18482, 24264, -18482, \\
&\qquad\qquad 7904, -1677, 124, -1\} \\
\underline{h}_{-3} &= \tfrac{1}{16} * \{16, 8, 0, 0, 0, \ldots\} \\
\underline{h}_{-2} &= \tfrac{1}{16} * \{0, 8, 12, 3, 0, 0, \ldots\} \\
\underline{h}_{-1} &= \tfrac{1}{16} * \{0, 0, 4, 11, 8, 2, 0, \ldots\}
\end{aligned}
$$

$$
\frac{1}{8!} * \left(\begin{array}{ccc}
\underline{g}_{-3} & \underline{g}_{-2} & \underline{g}_{-1} \\
\\
\frac{1136914560}{27877} & -\frac{2387315040}{195139} & \frac{123066720}{1365937} \\
-\frac{1655323200}{27877} & \frac{2141121840}{195139} & -\frac{2226000}{1365937} \\
\frac{1321223960}{27877} & \frac{878161880}{195139} & \frac{188417600}{1365937} \\
-\frac{633094403}{27877} & -\frac{498772701}{27877} & -\frac{2293862247}{1365937} \\
\frac{229000092}{27877} & \frac{4726413628}{195139} & \frac{10796596516}{1365937} \\
-\frac{46819570}{27877} & -\frac{3606490941}{195139} & -\frac{25245248833}{1365937} \\
124 & 7904 & 24264 \\
-1 & -1677 & -18482 \\
0 & 124 & 7904 \\
. & -1 & -1677 \\
. & 0 & 124 \\
. & . & -1 \\
. & . & 0 \\
. & . & .
\end{array} \right)
$$

The following procedure describes how the two-part basis functions are constructed by linearly combining the B-spline basis functions. (see Equations 15 and 16).

```
basis_xform_up( $b_{\mathrm{in}}$[], $b_{\mathrm{out}}$[], $w_{\mathrm{out}}$[], $L$ )
    $b_{\mathrm{out}} = w_{\mathrm{out}}$ = 0 ; /* zero vectors */
    for ($j = 0$; $j \leq 2^{L-1} - 4$; $j{+}{+}$)
        for ($k = 2j$; $k \leq (2j + 4)$; $k{+}{+}$)
            $b_{\mathrm{out}}[j]$ += $h[k - 2j] * b_{\mathrm{in}}[k]$ ;
    for ($j = 0$; $j \leq 2^{L-1} - 7$; $j{+}{+}$)
        for ($k = 2j$; $k \leq (2j + 10)$; $k{+}{+}$)
            $w_{\mathrm{out}}[j]$ += $g[k - 2j] * b_{\mathrm{in}}[k]$ ;
    for ($j$ in [$-3$,$-2$,$-1$,$2^{L-1} - 3$,$2^{L-1} - 2$,$2^{L-1} - 1$] )
            $b_{\mathrm{out}}[j]$ = $\underline{h}_j \cdot b_{\mathrm{in}}$ ; /*dot product*/
    for ($j$ in [$-3$,$-2$,$-1$,$2^{L-1} - 4$,$2^{L-1} - 5$,$2^{L-1} - 6$] )
            $w_{\mathrm{out}}[j]$ = $\underline{g}_j \cdot b_{\mathrm{in}}$ ;
```

This procedure can be expressed as multiplication by a banded matrix, and so the inverse procedure
`basis_xform_down(` b_{in}`[],` w_{in}`[],` b_{out}`[],` L `)`
can be obtained by solving this banded linear system.

The following procedure can be used to obtain B-spline coefficients given two-part coefficients.

```
coef_xform_down( $b_{\text{in}}$[], $w_{\text{in}}$[], $b_{\text{out}}$[], $L$ )
    $b_{\text{out}}$ = $\mathbf{0}$ ; /* zero vector */
    for($k = 0$; $k \leq 2^{L-1} - 4$; $k{+}{+}$)
        for($j = 2k$; $j \leq (2k+4)$; $j{+}{+}$)
            $b_{\text{out}}[j]$ += $h[j-2k] * b_{\text{in}}[k]$ ;
    for($k = 0$; $k \leq 2^{L-1} - 7$; $k{+}{+}$)
        for($j = 2k$; $j \leq (2k+10)$; $j{+}{+}$)
            $b_{\text{out}}[j]$ += $g[j-2k] * w_{\text{in}}[k]$ ;
    for($k$ in [$-3$,$-2$,$-1$,$2^{L-1} - 3$,$2^{L-1} - 2$,$2^{L-1} - 1$] )
            $b_{\text{out}}$    += $\underline{h}_k * b_{\text{in}}[k]$ ; /*vector addition*/
    for($k$ in [$-3$,$-2$,$-1$,$2^{L-1} - 4$,$2^{L-1} - 5$,$2^{L-1} - 6$] )
            $b_{\text{out}}$    += $\underline{g}_k * w_{\text{in}}[k]$ ;
```

This procedure can be expressed as multiplication by a banded matrix, and so the inverse procedure
`coef_xform_up(` b_{in}`[],` b_{out}`[],` w_{out}`[],` L `)`
can be obtained by solving this banded linear system.

Appendix B

Bounding the Magnitude of Coefficients

Beylkin et al. [8] prove that Calderon-Zygmund operators can be represented in the non-standard wavelet basis with only $O(n)$ significant entries. This appendix summarizes their arguments.

Given a Calderon-Zygmund kernel, the magnitude of a wavelet coefficient can be bounded as follows. In order to compute some coefficient

$$\int ds \int dt \; K(s,t)\, \psi_{i,j}(s)\psi_{k,l}(t) \tag{172}$$

where ψ has M vanishing moments, $K(s,t)$ is approximated by a degree M multivariate taylor series about a point (s_0, t_0) that is under the support of $\psi_{i,j}(s)\psi_{k,l}(t)$:

$$K(s,t) = P(s,t) + R(s,t) \tag{173}$$

$P(s,t)$ is a degree M polynomial. $R(s,t)$ is a residual function which, under the support of $\psi_{i,j}(s)\psi_{k,l}(t)$, is can be bounded by

$$\begin{aligned} \mid R(s,t) \mid \;&\leq\; D^M K(s_m, t_m) * dist((s,t),(s_0,t_0))^M && (174) \\ &\leq\; \frac{C_M}{\text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} * \max(I_s, I_t)^M && (175) \end{aligned}$$

D^M is the total M^{th} derivative. (s_m, t_m) is the parametric position where the derivative D^M has its maximum value. $dist$ measures the distance of the point (s,t) from

(s_0, t_0). Line (175) uses the Calderon-Zygmund smoothness property given in Equation (94). singDist measures the minimum distance between the singularity in (s, t) and the support of the basis function $\psi_{i,j}(s)\psi_{k,l}(t)$. I_s is the support length of the basis function $\psi_{i,j}(s)$. And I_t is the support length of the basis function $\psi_{k,l}(t)$. More generally, if the dimension d of the parameter space s is greater than 1, and the one dimensional support length is I_s, then the d dimensional volume supported by the basis function is I_s^d.

The magnitude of the coefficient can now be bounded

$$
\begin{aligned}
& \mid \int ds \int dt\ K(s,t)\psi_{i,j}(s)\psi_{k,l}(t) \mid && (176)\\
= & \mid \int ds \int dt\ (P(s,t) + R(s,t))\, \psi_{i,j}(s)\psi_{k,l}(t) \mid && (177)\\
= & \mid \int ds \int dt\ R(s,t)\, \psi_{i,j}(s)\psi_{k,l}(t) \mid && (178)\\
\leq & \int_{supp(\psi_{i,j})} ds \int_{supp(\psi_{k,l})} dt\ \frac{C_M\ \max(I_s, I_t)^M}{I_s^{d/2} I_t^{d/2} \text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} && (179)\\
= & \frac{C_M\ \max(I_s, I_t)^M}{I_s^{d/2} I_t^{d/2} \text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} \int_{supp(\psi_{i,j})} ds \int_{supp(\psi_{k,l})} dt\ 1 && (180)\\
= & \frac{C_M\ \max(I_s, I_t)^M}{I_s^{d/2} I_t^{d/2} \text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} I_s^d I_t^d && (181)\\
= & \frac{C_M\ I_s^{d/2} I_t^{d/2} \max(I_s, I_t)^M}{\text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} && (182)\\
\leq & \frac{C_M\ \max(I_s, I_t)^d \max(I_s, I_t)^M}{\text{singDist}(\psi_{i,j}, \psi_{k,l})^{d+M}} && (183)\\
= & C_M \left(\frac{\max(I_s, I_t)}{\text{singDist}(\psi_{i,j}, \psi_{k,l})} \right)^{d+M} && (184)
\end{aligned}
$$

In line (178), the M vanishing moments are used to make P vanish. Line (179), uses a bound on R. The terms $I_s^{d/2}$ and $I_s^{d/2}$ appear in the denominator since the basis function $\psi_{i,j}(s)\psi_{k,l}(t)$ is L^2 normalized. Line (181) uses the volume of the dt and ds integrals.

If the non-standard bivariate wavelet basis functions (e.g., $\psi_{i,j}(s)\psi_{i,l}(s)$) are used, then the ratio of line (184) can simply be expressed as

$$\frac{C_M}{1 + \mid j - l \mid^{d+M}} \tag{185}$$

independent of the level i. Thus for any user supplied epsilon, there is a width w (independent of the level i) such that the coefficients of the basis functions that are "far enough" away from the diagonal, with indices $\mid j - l \mid > w$, have magnitudes less than epsilon. Only the basis functions within w of the diagonal $j = l$ will be significant. Therefore the number of significant coefficients on a single level i is equal to $w * length(diagonal(i))$. Since there are only $\log n$ levels, the number of significant entries is $O(n)$:

$$\begin{aligned} \sum_{i=0}^{\log n} w * length(diagonal(i)) &= \sum_i w * 2^i \\ &= 2wn \\ &= O(n) \end{aligned} \tag{186}$$

Appendix C

Bounding the Condition Number

A number of researchers [81, 47, 25] have studied the condition number of the matrices that arise from finite-element solutions to differential equations. Their main result is that when a wavelet basis is used, the condition number of the matrix remains constant, independent of n, the number of functions in the basis. This is an important improvement over fixed resolution bases, where the condition number grows as a polynomial of n. This appendix summarizes some of the theory.

Given some symmetric positive definite matrix $\mathbf{M}$, the *Raleigh quotient* for some vector $\mathbf{x}$ is defined as

$$R(\mathbf{x}) = \frac{\mathbf{x^T M x}}{\mathbf{x^T x}} \tag{187}$$

The numerator of the Raleigh quotient is the "quadratic energy" of the vector $\mathbf{x}$, and the denominator is the square of the euclidean length of $\mathbf{x}$. The condition number κ of $\mathbf{M}$ is defined as the ratio of the largest and smallest value that R takes, over all non-zero vectors $\mathbf{x}$. In some sense, the condition number measures how close the euclidean length of $\mathbf{x}$ is to the energy of $\mathbf{x}$.

When one is solving a differential equation or an optimization problem using a finite element solution method, the resulting matrix has terms of the form

$$\mathbf{M}_{i,j} = \int dt \; D^a(\phi_i(t)) * D^a(\phi_j(t)) \tag{188}$$

for univariate problems and

$$\mathbf{M}_{i,j} = \int ds \int dt \; D^a(\phi_i(s,t)) * D^a(\phi_j(s,t)) \tag{189}$$

for bivariate problems, where a is some fixed constant, and D^a is the ath derivative (or sum of partial derivatives) of ϕ. Thus, the energy term $\mathbf{x^T M x}$ induces the norm

$$E(X) = \left(\int dt\ (D^a(X))^2\right)^{1/2} \tag{190}$$

on the function $X(t)$ that corresponds to the vector of coefficients $\mathbf{x}$. This norm is called the *Sobolov norm of degree a*.

When one uses a fixed resolution basis, then the euclidean length of the vector $\mathbf{x}$ is a poor estimate of the Sobolov energy of the function X for the following reason. As one uses a fixed resolution basis with more and more basis functions, the energy of any single basis function becomes higher and higher; for $a \geq 1$, thin "humps" have more Sobolov energy than wide "humps". Even though each individual basis function has more energy, the euclidean length of the vector associated with a single basis function remains 1. Thus, the maximum value of R increases with n. On the other hand, by using a higher resolution basis, one can still obtain a function X with at least as low as energy as could be obtained with the lower resolution basis, and so the minimum value of R does not increase. Thus the condition number, κ, increases with n. The condition number can not be improved by scaling the fixed resolution basis functions. If one scales down all of the basis functions by a factor of s, then the maximum value of R decreases by a factor of s^2, but so does the smallest value of R.

When using a wavelet basis, one is able to scale down the thinner basis functions, while maintaining the scale of the wider basis functions. In particular, it can be shown that if all of the wavelet basis functions $\psi_{i,j}$ are properly scaled on each level so that the Sobolov energy of each individual basis function is the same, then the euclidean length of the vector of wavelet coefficients, $\mathbf{x}$, is a good estimate of the Sobolov energy of the function X regardless of the number of wavelet levels i, and the number of basis functions n employed. Thus, κ remains bounded [81, 47, 25].

To obtain the correct scaling, the diagonal terms of the matrix, which for the univariate problem are:

$$\int dt\ D^a(\psi_{i,j}(t)) D^a(\psi_{i,j}(t)) \tag{191}$$

must all have the same value irrespective of i (by doing this, all basis functions, on all levels i, have the same Sobolov energy). The exact scaling needed to obtain this property depends on the number, d, of parameters (i.e., s and t) in the problem, and the degree, a, of the derivative in the energy term. It can easily be verified, that the necessary scaling relationship is

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(1/2-a/d)(i-L)}\ \phi(2^{(i-L)}t-j) \\ \psi_{i,j}(t) &= 2^{(1/2-a/d)(i-L)}\ \psi(2^{(i-L)}t-j) \end{aligned} \tag{192}$$

For thin plate curves, $a=2$ and $d=1$, thus the proper scaling is

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(-3/2)(i-L)}\ \phi(2^{(i-L)}t-j) \\ \psi_{i,j}(t) &= 2^{(-3/2)(i-L)}\ \psi(2^{(i-L)}t-j) \end{aligned} \tag{193}$$

This means that as one goes from level i to level $i-1$ the basis functions become twice as wide, and $2^{3/2}$ as tall. In the pyramid code, this is achieved by multiplying all of the h and g entries by $2^{3/2}$, and all of the $\tilde{h}$ and $\tilde{g}$ by $2^{-3/2}$

Likewise, for thin plate surfaces, $a=2$ and $d=2$, and the proper scaling for the univariate basis functions making up the bivariate non-standard basis functions is

$$\begin{aligned} \phi_{i,j}(t) &= 2^{(-1/2)(i-L)}\ \phi(2^{(i-L)}t-j) \\ \psi_{i,j}(t) &= 2^{(-1/2)(i-L)}\ \psi(2^{(i-L)}t-j) \end{aligned} \tag{194}$$

This means that as one goes from level i to level $i-1$ the basis functions become twice as wide, and $2^{1/2}$ as tall. In the pyramid code, this is achieved by multiplying all of the h and g entries by $2^{1/2}$, and all of the $\tilde{h}$ and $\tilde{g}$ by $2^{-1/2}$

The constrained optimization problems that arise in geometric modeling are more complicated than the simple differential equations studied in the theoretical literature. In particular, in a modeling problem, the constraints may be placed anywhere in the domain of the curve or surface, and not just on the boundary. And so the matrix **M** measures not just Sobolov (thin plate) energy but also constraint information. Adding constraints can both raise and lower the minimum value of R. Therefore the proofs for the theoretical bounds for simple differential equations can not be directly applied to geometric modeling problems, and the "proper" theoretical scaling may not be optimal.

For constrained thin plate curves and surfaces both the scalings of Equations (193) and (194) and the scaling

$$\begin{aligned}\phi_{i,j}(t) &= 2^{-(i-L)}\ \phi(2^{(i-L)}t - j)\\ \psi_{i,j}(t) &= 2^{-(i-L)}\ \psi(2^{(i-L)}t - j)\end{aligned} \tag{195}$$

were experimented with. The results using these different scalings did not differ greatly, but the best results were obtainted using the scaling of Equation (195), and so those results are reported in the experimental section.

Bibliography

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM Journal on Scientific Computing*, 14(1), Jan 1993.

[2] Bradley Alpert. Construction of simple multiscale bases for fast matrix operations. In Gregory Beylkin, Ronald Coifman, Ingrid Daubechies, Stephane Mallat, Yves Meyer, Louise Raphael, and Mary Beth Ruskai, editors, *Wavelets and Their Applications*, pages 211–226. Jones and Bartlett, Cambridge, 1992.

[3] Bradley Alpert. A class of bases in L^2 for the sparse representation of integral operators. *SIAM Journal on Mathematical Analysis*, 24(1), Jan 1993.

[4] Peter R. Atherton. A scan-line hidden surface removal procedure for constructive solid geometry. *Computer Graphics*, 17(3):73–82, 1983.

[5] Richard Bartels and John Beatty. A technique for the direct manipulation of spline curves. In *Graphics Interface 1989*, pages 33–39, 1989.

[6] Richard Bartels, John Beatty, and Brian Barsky. *An Introduction to Splines for Use in Computer Graphics and Modeling.* Morgan Kaufmann, 1987.

[7] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics*, 23(3):325–334, July 1989.

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

[9] G. Beylkin, R. Coifman, and V. Rokhlin. Wavelets in numerical analysis. In Gregory Beylkin, Ronald Coifman, Ingrid Daubechies, Stephane Mallat, Yves Meyer, Louise Raphael, and Mary Beth Ruskai, editors, *Wavelets and Their Applications*, pages 181–210. Jones and Bartlett, Cambridge, 1992.

[10] James Blinn. A generalization of algebraic surface drawing. *ACM TOG*, 1(3):235–256, July 1982.

[11] Terrence Boult. *Information-Based Complexity in Nonlinear Equations and Computer Vision.* PhD thesis, Columbia University, 1986.

[12] William L. Briggs. *A Multigrid Tutorial.* SIAM, 1987.

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

[14] Per Christensen, Eric Stollnitz, David Salesin, and Tone DeRose. Importance-driven wavelet radiance. Technical Report TR 94-01-05, Department of Computer Science, University of Washington, January 1994.

[15] Charles Chui and Ewald Quak. Wavelets on a bounded interval. *Numerical Methods of Approximation Theory*, 9:53–75, 1992.

[16] Charles K. Chui. *An Introduction to Wavelets*, volume 1 of *Wavelet Analysis and its Applications.* Academic Press Inc., 1992.

[17] A. Cohen, Ingrid Daubechies, and J. C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communication on Pure and Applied Mathematics*, 45:485–560, 1992.

[18] Elane Cohen, Tom Lyche, and Richard Riesenfeld. Discrete b-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, October 1980.

[19] Michael F. Cohen. Spacetime constraints. *Computer Graphics*, 26(2):293–302, July 1992.

[20] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988.

[21] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985.

[22] Michael F. Cohen, Donald P. Greenberg, David Immel, and Phillip Brock. An efficient radiosity approach for realistic image synthesis. *IEEE CG&A*, 6(3):26–35, 1986.

[23] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, 1993.

[24] Robert Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, July 1984.

[25] Wolfgang Dahmen and Angel Kunoth. Multilevel preconditioning. *Numerische Mathematik*, 63:315–344, 1992.

[26] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, 1992.

[27] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 1990.

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

[29] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics, Principles and Practice.* Addison-Wesley Publishing, 1990.

[30] David Forsey and Richard Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4):205–212, August 1988.

[31] David Forsey and Lifeng Weng. Multi-resolution surface approximation for animation. In *Graphics Interface*, 1993.

[32] Barry Fowler. Geometric manipulation of tensor product surfaces. In *Proceedings, Symposium on Interactive 3D Graphics*, pages 101–108, 1992.

[33] Reid Gershbein. Personal communication.

[34] Reid Gershbein, Peter Schröder, and Pat Hanrahan. Textures and radiosity: Controlling emission and reflection with texture maps. In *Computer Graphics, Annual Conference Series, 1994*, pages 51–58. Siggraph, 1994.

[35] Philip Gill, Walter Murray, and Margaret Wright. *Practical Optimization*. Academic Press, 1981.

[36] Robert A. Goldstein and Roger Nagel. 3-D visual simulation. *Simulation*, 16(1):25–31, January 1971.

[37] Gene Golub and Charles Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[38] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):212–222, July 1984.

[39] Steven J. Gortler, Michael F. Cohen, and Philipp Slusallek. Radiosity and relaxation methods. *IEEE Computer Graphics and Applications*, 14, November 1994.

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

[41] Silicon Graphics. *Graphics Library Programming Guide*. Silicon Graphics Documents, 1991.

[42] A. Haar. Zur theorie der orthogonalen funktionen-systeme. *Math. Ann.*, 69:331–371, 1910.

[43] Mark Halstead, Michael Kass, and Tone DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Computer Graphics, Annual Conference Series, 1993*, pages 35–43. Siggraph, 1993.

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

[45] Paul S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing.* PhD thesis, University of California at Berkeley, January 1991.

[46] Tim Van Hook. Real-time shaded NC milling display. *Computer Graphics*, 20(4):15–20, August 1986.

[47] S. Jaffard and Ph. Laurençot. Orthonormal wavelets, analysis of operators, and applications to numerical analysis. In Charles K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications*, pages 543–602. Academic Press, 1992.

[48] Bjorn Jawerth and Wim Sweldens. Wavelet multiresolution analysis adapted for the fast solution of boundary value ordinary differential equations. In *Sixth Copper Mountain Multigrid Conference*, April 1993.

[49] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, 1986.

[50] Robert Lewis. Making shaders more physically plausible. In *Fourth Eurographics Workshop on Rendering*, pages 47–62, June 1993.

[51] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics, Annual Conference Series, 1993*, pages 199–208. Siggraph, August 1993.

[52] Zicheng Liu, Steven Gortler, and Michael F. Cohen. Hierarchical spacetime control. In *Computer Graphics, Annual Conference Series, 1994*, pages 35–42, August 1994.

[53] Michael Lounsbery, Tony DeRose, and Joe Warren. Multiresolution surfaces of arbitrary topological type. Technical Report 93-10-05b, Department of Computer Science and Engineering, University of Washington, January 1994.

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

[55] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19:129–147, June 1982.

[56] Jean Meinguet. Multivariate interpolation at arbitrary points made simple. *Journal of Applied Mathematics and Physics (ZAMP)*, 30:292–304, 1979.

[57] Henry Moreton and Carlo Sequin. Functional optimization for fair surface design. *Computer Graphics*, 26(4):167–176, July 1992.

[58] Alex Pentland. Fast solutions to physical equilibrium and interpolation problems. *The Visual Computer*, 8(5):303–314, 1992.

[59] Willaim Press, Saul Teukolski, William Vetterling, and Brian Flannery. *Numerical Recipes in C, The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1992.

[60] Sam Qian and John Weiss. Wavelets and the numerical solution of partial differential equations. *Journal of Computational Physics*, 106(1):155–175, May 1993.

[61] Ewald Quak and Norman Weyrich. Decomposition and reconstruction algorithms for spline wavelets on a bounded interval. Technical Report 294, Center for Approximation Theory, Texas A&M, 1993.

[62] T. Rando and J Roulier. Designing faired parametric surfaces. *Computer Aided Design*, 23(7):492–497, September 1991.

[63] David Salesin, Dani Lischinski, and Tony DeRose. Reconstructing illumination functions with selected discontinuities. *Third Eurographics Workshop on Rendering*, pages 99–112, 1992.

[64] Christophe Schlick. A customizable reflectance model for everyday rendering. In *Fourth Eurographics Workshop on Rendering*, pages 73–84, June 1993.

[65] Peter Schröder, Steven J. Gortler, Michael F. Cohen, and Pat Hanrahan. Wavelet projections for radiosity. In *Fourth Eurographics Workshop on Rendering*, pages 105–114, June 1993. Journal version appears in Computer Graphics Forum 13(2):141-152, June 1994.

[66] Peter Schröder and Pat Hanrahan. On the form factor between two polygons. In *Computer Graphics, Annual Conference Series, 1993*, pages 163–164. Siggraph, August 1993.

[67] Peter Schröder and Pat Hanrahan. Wavelet methods for radiance computation. In *Fifth Eurographics Workshop on Rendering*, pages 303–312, June 1994.

[68] Francois Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–118, June 1994.

[69] Francois Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics*, 23(3):335–344, July 1989.

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

[71] Richard Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–439, June 1990.

[72] Demetri Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):129–139, March 1986.

[73] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, July 1986.

[74] Demetri Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, July 1988.

[75] Roy Troutman and Nelson Max. Radiosity algorithms using higher-order finite elements. In *Computer Graphics, Annual Conference Series, 1993*, pages 209–212. Siggraph, August 1993.

[76] William Welch and Andrew Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, July 1992.

[77] Turner Whitted. An improved illumination model for shaded display. *CACM*, 23(6):343–349, June 1980.

[78] Lance Williams. Pyramidal parametrics. *Computer Graphics*, 17(4):1–11, 1983.

[79] Goeff Wyvill, Craig McPheeters, and Brian Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, April 1986.

[80] Ming-Haw Yaou and Wen-Thong Chang. Fast surface interpolation using multiresolution wavelet transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994. to appear.

[81] Harry Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.

[82] Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Computer Graphics, Annual Conference Series, 1993*, pages 213–220. Siggraph, August 1993.