Wavelet Radiosity

Steven J. Gortler Peter Schröder Michael F. Cohen Pat Hanrahan

Department of Computer Science
Princeton University

Abstract

Radiosity methods have been shown to be an effective means to solve the global illumination problem in Lambertian diffuse environments. These methods approximate the radiosity integral equation by projecting the unknown radiosity function into a set of basis functions with limited support resulting in a set of n linear equations where n is the number of discrete elements in the scene. Classical radiosity methods required the evaluation of n^2 interaction coefficients. Efforts to reduce the number of required coefficients without compromising error bounds have focused on raising the order of the basis functions, meshing, accounting for discontinuities, and on developing hierarchical approaches, which have been shown to reduce the required interactions to $O(n)$.

In this paper we show that the hierarchical radiosity formulation is an instance of a more general set of methods based on *wavelet* theory. This general framework offers a unified view of both higher order element approaches to radiosity and the hierarchical radiosity methods. After a discussion of the relevant theory, we discuss a new set of linear time hierarchical algorithms based on wavelets such as the multiwavelet family and a flatlet basis which we introduce. Initial results of experimentation with these basis sets are demonstrated and discussed.

CR Categories and Subject Descriptors: I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; G.1.9 [Numerical Analysis]: *Integral Equations – Fredholm equations*.

Additional Key Words and Phrases: global illumination, wavelets, hierarchical radiosity.

1 Introduction

In computer graphics, radiosity methods have been used to solve the global illumination problem in environments consisting entirely of Lambertian (diffuse) reflectors and emitters. The solution is a radiosity function over the domain of the surfaces in the scene. Classical radiosity [9, 6] (CR), derived from the radiative heat transfer literature, approximates the radiosity function as piecewise constant. An energy balance argument gives rise to a linear system. This system has n^2 coefficients called *form factors*. Here n is the number of discrete areas, or *elements*, over which the radiosity function has been assumed to be constant. The form factor describes the fraction of the energy leaving one element and arriving at another. Typically, an iterative algorithm such as Gauss-Seidel iteration [22] or progressive radiosity [5, 10] is used to solve the system of linear equations for the radiosities.

An integral equation called the rendering equation was proposed by Kajiya to model the global illumination problem [14]. He showed that CR is a particular approximation to this equation. By casting the problem in this form, techniques developed for the solution of integral equations [8] can be exploited to solve the radiosity equation.

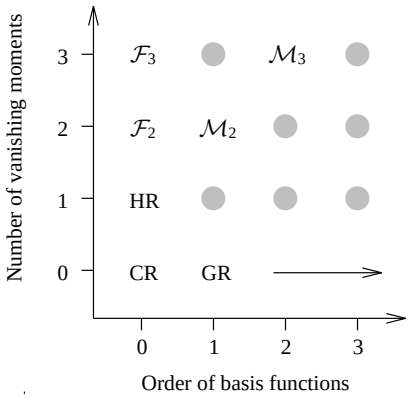

Figure 1: The space of projection methods for radiosity.

In particular, Heckbert [12, 13] has demonstrated that the linear system in radiosity can be derived by *projecting* the radiosity integral into a finite dimensional function space. The CR algorithm results from using the *space of piecewise constant functions*, (i.e., projecting the function into a set of constant (or "box") *basis* functions). In general, a function can be projected into any finite dimensional function space. A desirable finite dimensional space is one that can represent the function accurately with as few terms as possible. In his studies, Heckbert considered radiosity functions that are piecewise linear. Zatz [25] has used Legendre polynomials to arrive at solutions that are piecewise polynomial of higher order. Other researchers have explored the use of higher order bases in the mesh construction and reconstruction phases of the algorithm [18] as well as discontinuity meshing [15, 13]. The use of higher order bases, which we will refer to as galerkin radiosity (GR), has been shown to lower the number of basis functions needed to obtain a particular level of accuracy, albeit at a higher cost per basis.

A second avenue of research has attempted to lower the computational complexity of solving the linear system which arises in CR. Hanrahan et al. [11] presented a hierarchical radiosity method (HR) modeled after recent advances in n-body algorithms. HR exploits the fact that neighboring patches in the environment often have similar form factors to distant patches. This reasoning is extended to form a hierarchy of patches, (i.e., a hierarchy of basis functions) in a straightforward manner.

While the methods using higher order bases try to exploit *coherence* in the illumination function, HR tries to exploit the coherence in the form factor itself, more precisely, in the kernel of the radiosity integral. In particular, HR is based on approximating the kernel as a constant function over intervals of varying sizes. In places that the kernel varies slowly, large intervals are used. Where the kernel varies quickly, smaller intervals are needed.

Recently Beylkin et al. [3] made the observation that integral operators satisfying very general smoothness conditions can be approximated to any finite precision with only $O(n)$ coefficients when projected into a wavelet basis instead of the usual $O(n^2)$. This remarkable result means that, in practice, integral equations governed by smooth kernels lead to sparse matrices that can be solved in linear time. Since the radiosity kernel is, in general, a smooth function of the type required by this theorem, wavelet methods can be used to obtain $O(n)$ complexity radiosity algorithms. We call this *wavelet radiosity*.

Hierarchical basis functions have been used before with finite-element methods [24] and applied to problems such as surface interpolation [23]. In those instances, hierarchical basis functions were used to improve the condition number of the matrix. In our context, the hierarchical basis functions (wavelets) are used because many of the resulting matrix coefficients are small enough to be ignored while still allowing for an accurate answer. In some sense we are regarding the matrix as an image on which we are able to perform lossy compression. Coefficients are negligible because over many regions the kernel can be well approximated by a low order polynomial.

The mathematical tools of wavelet analysis provide a general framework offering a unified view of both higher order element approaches to radiosity, and the hierarchical radiosity methods. Figure 1 places earlier algorithms plus the new methods we investigate here into a matrix relating hierarchy versus the order of the underlying basis. CR uses zero order polynomials, while 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. HR exploits "constant" smoothness in the kernel. Within this context, we recognize HR 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 [1] ($\mathcal{M}_{2,3}$ in Figure 1). We will also introduce a new family of wavelets, which we have dubbed *flatlets* ($\mathcal{F}_{2,3}$ in Figure 1) that require only low order quadrature methods while maintaining most of the benefits of other wavelet sets.

This paper proceeds with a review of projection methods for solving integral equations followed by a discussion of recent advances concerning the solution of integral equations using wavelets. Finally we discuss our implementation and report experimental findings. Some of the more technical details of wavelet projections, as well as a detailed analysis of the underlying mathematical framework, are described in [20].

2 The Radiosity Integral Equation

If all surfaces and emitters are Lambertian diffuse, the rendering equation can be written as,

$$B(s_1, s_2) = E(s_1, s_2) + \rho(s_1, s_2) \int\int dt_1 dt_2 \, \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} V_{st} B(t_1, t_2) \tag{1}$$

where $B(s_1, s_2)$ gives the radiosity at a point specified by the surface parameters s_1, s_2, E the emission, and ρ the reflectivity[1]. The *kernel* of the integral,

$$k(s_1, s_2, t_1, t_2) = \rho(s_1, s_2) \frac{\cos\theta_s \cos\theta_t}{\pi r_{st}^2} V_{st}$$

is a function describing the geometric and visibility relationship between two points in the domain; θ_s and θ_t are the angles between the surface normals and the line between **s** and **t**; r_{st} is the distance between the two points; V_{st} is 1 if point **s** is visible to point **t** and 0 otherwise.

Over many large intervals, where r is large relative to the size of the patches, the kernel is well represented by a low order polynomial. Notable exceptions include the corners of the environment where r^2 goes to 0 and the kernel is singular, and shadow discontinuities where the visibility switches abruptly from 0 to 1.

3 Projections

After a short review of function projections we will show how projections can be used to find approximate solutions to integral equations such as the radiosity equation. The ideas presented here can be found in greater detail in [12, 25].

We begin by writing the approximation of a function $B(s)$ in a finite dimensional function space where all functions $\hat{B}(s)$ can be expressed as a linear combination of n *basis functions* $N_i(s)$

$$B(s) \approx \hat{B}(s) = \sum_{i=1}^{n} B_i N_i(s)$$

where the B_i are scalar coefficients with respect to the chosen bases. For example, the space of piecewise constant functions is spanned by a basis of translated "box" functions, and the space of piecewise linear functions is spanned by a basis of translated "hat" functions.

To complete the approximation, we must find a way to derive the coefficients. For this, we define an inner product of two functions $f(s)$ and $g(s)$ as $\langle f, g \rangle = \int ds \, f(s) g(s)$. Two functions are orthogonal iff $\langle f, g \rangle = 0$. We then say that a function $\hat{B}(s)$ is the orthogonal projection of $B(s)$ into the finite dimensional function space if $\langle B - \hat{B}, N_i \rangle = 0$ for all basis functions $N_i(s)$.

If the original basis functions are orthonormal we can find the coefficients of a function $B(s)$ with respect to the basis $\{N_i\}$ by performing inner products

$$\hat{B}(s) = \sum_i B_i N_i(s) = \sum_i \langle B, N_i \rangle N_i(s)$$

In the case of bases which are not orthonormal we must use inner products with the *dual* basis functions (see [20]) to find the coefficients.

Using projection methods, instead of solving the integral equation (1), we solve the related integral equation[2]

$$\hat{B}(s) = \hat{E}(s) + \sum_i \left\langle \int dt \, k(s,t) \hat{B}(t), \, N_i(s) \right\rangle N_i(s) \tag{2}$$

In words, we *operate* on (integrate against the kernel) the projected 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 reprojected against the $N_i(s)$. $\hat{B}$ can be obtained by solving the linear system

$$\begin{aligned} B_i &= E_i + \sum_j B_j K_{ij} \\ K_{ij} &= \int ds \int dt \, k(s,t) N_j(t) N_i(s) \end{aligned} \tag{3}$$

To compute the integrals K_{ij} some form of numerical quadrature or closed form solution [21] must be employed. If the basis functions are piecewise constant, these integrals are related to the well known form factors.

[1] The reflectivity, ρ, is actually a function of wavelength. Without loss of generality, we will consider only a monochromatic world for the remainder of this paper.

[2] In order to simplify the presentation we will write the radiosity function as having one variable, and the kernel function as having two variables. In the text we will explain what needs to be done for a 3D radiosity implementation.

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 resolution of the grid, and p the degree of the polynomial used which favors higher order basis functions. Higher order basis functions 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.

One set of choices for basis functions is given by the family of functions called wavelets.

4 Wavelets

Wavelet theory is a rapidly developing field that has its roots in pure mathematics [7] and signal processing [16]. Good introductions to the topic can be found in [17, 4]. In this section we review some wavelet theory focusing on the relevant issues for radiosity.

Wavelets form hierarchical bases which can offer alternative bases for familiar finite dimensional function spaces. The simplest wavelet construction is the Haar construction shown in Figure 2. In the upper left is a set of basis functions which span all piecewise constant functions at resolution 8 on the interval. Using the operators g (pairwise differencing) and h (pairwise averaging) we can construct another basis for the same space (upper right). Four of these functions are just like the original basis, only wider, thus we can repeat the construction (middle right). Repeating once more we finally have a basis for the original space of functions consisting of the overall average ϕ_0 and the difference functions $\psi_{i,j}$ from all the lower levels. The last set of functions is known as the Haar *wavelet basis*. This construction is very similar to an image pyramid that one might use for texture mapping. In such a pyramid the image (function in our case) is represented at different levels of resolution by successive averaging steps. In the Haar pyramid we only remember the overall average and all the *differences* between successive levels of the pyramid.

The Haar basis is only the simplest example of an infinite family of such constructions, however the basic principles are the same for all wavelet bases. More formally we start with two functions $\psi(s)$ (sometimes called the *detail* function) and $\phi(s)$ (the *smooth* function) defined on the unit interval $s \in [0,1]$. Scales (or levels) i and translates j of $\phi(s)$ and $\psi(s)$ are expressed as

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

with $j = 0, \ldots, 2^i - 1$. According to this indexing, the function $\phi_{i,j}$ is just like the function $\phi_{i-1,j}$ except that $\phi_{i-1,j}$ is twice as wide, and $1/\sqrt{2}$ times as tall (the wider functions are shorter so that $\langle \phi_{i,j}, \phi_{i,k} \rangle$ remains constant independent of i). Similarly, $\phi_{i,j}$ is just like the function $\phi_{i,j+1}$ except it is translated. To create an $n = 2^L$ dimensional function space we construct an L level hierarchy of functions that are scales and translates of ϕ and ψ (Figure 2 illustrates $L = 3$). We obtain the wavelet basis for the hierarchy by choosing only the detail shapes on all levels plus the smooth shape on the top level, $\psi_{i,j}$, $i = 0, \ldots, L-1$ and ϕ_0. Between levels there is the so called *two-scale* relationship

$$\begin{aligned}\phi_{i-1,j} &= \sum_k h_{k-2j}\phi_{i,k}\\ \psi_{i-1,j} &= \sum_k g_{k-2j}\phi_{i,k}\end{aligned}$$

In words the ϕ functions at a given level can be linearly combined to yield ϕ and ψ functions at the next coarser level. This combination can be expressed as a convolution with some sequences h and g with the result subsampled by 2 (expressed by the factor 2 in the index "$k-2j$" of h and g). The sequences h and g can be thought of as a low pass filter and high pass filter respectively.

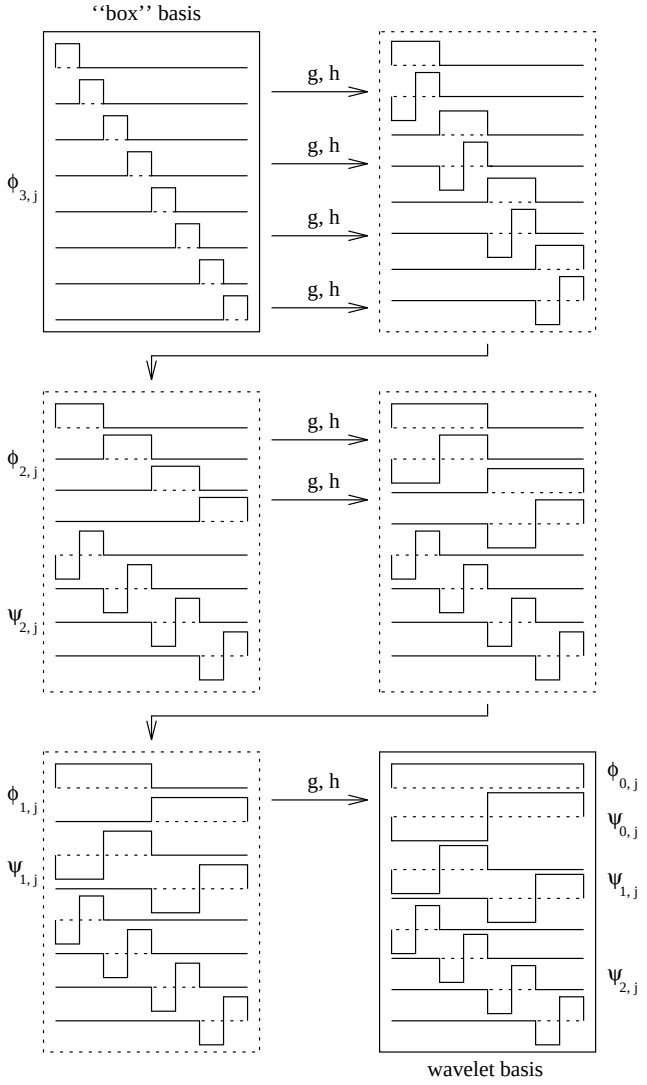

Figure 2: Transformation of a piecewise constant basis into the Haar wavelet basis.

The projection of an arbitrary function $B(s)$ into a wavelet[3] basis can be formally written as

$$\hat{B}(s) = \langle B, \phi_0 \rangle \phi_0(s) + \sum_{ij} \langle B, \psi_{i,j} \rangle \psi_{i,j}(s) \qquad (4)$$

Instead of computing all the above inner products, we can find the coefficients efficiently by exploiting the two-scale relationship. Given the projection of some arbitrary function $B(s)$ with respect to the lowest level basis $\phi_{L,j}$ the wavelet coefficients can be found using a pyramid algorithm [16]. Each stage of this algorithm takes a vector of coefficients and convolves it with the filters h and g, returning the smooth and detail coefficients one level up

[3] To simplify the discussion we are assuming that we have an orthonormal wavelet basis. We discuss the non orthonormal case in [20].

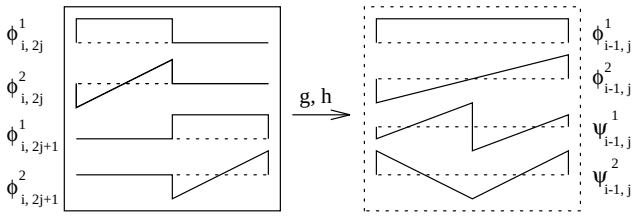

Figure 3: The $\mathcal{M}_2$ wavelet construction whose smooth shapes are the first two Legendre polynomials. Both of the detail shapes (lower right) have two vanishing moments.

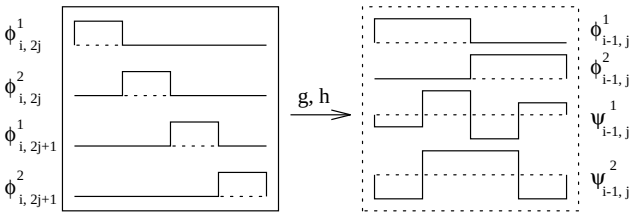

Figure 4: The $\mathcal{F}_2$ wavelet construction. $\mathcal{F}_2$ bases have two different detail shapes. Both of the detail shapes have two vanishing moments.

```
XformUp( vector B_φ, int i )
  for( j = 0; j < 2^i/2; j++)
    B_φ^up[j] = Σ_k h_{k-2j} B_φ[k];
    B_ψ^up[j] = Σ_k g_{k-2j} B_φ[k];
  return ( B_φ^up, B_ψ^up );
```

The entire one dimensional pyramid transform is then stated as

```
PyramidUp( vector B_{φ_{L,k}} )
  for( i = L; i > 0; i-- )
    (B_{φ_{i-1,k}}, B_{ψ_{i-1,k}}) = XformUp( B_{φ_{i,k}} , i );
  return ( B_{φ_0}, B_{ψ_{i,k}}, i = 0, ..., L-1);
```

If the h and g convolutions have constant width (with respect to i) then each call to **XformUp** has cost linear in the length of the array passed in. Since each successive call in **PyramidUp** works on only the smooth half left by the previous call the overall runtime to build the pyramid is $O(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1) = O(n)$.

A similar algorithm **PyramidDown** reverses this process using **XformDown** for successive calls

```
XformDown( vector B_φ, vector B_ψ, int i )
  for( j = 0; j < 2 * 2^i; j++ )
    B_φ^down[j] = Σ_k h_{j-2k} B_φ[k] + Σ_k g_{j-2k} B_ψ[k];
  return B_φ^down ;

PyramidDown( B_{φ_0}, B_{ψ_{i,k}}, i = 0, ..., L-1 )
  for( i = 0; i < L; i++ )
    B_{φ_{i+1,k}} = XformDown( B_{φ_{i,k}}, B_{ψ_{i,k}} , i );
  return B_{φ_{L,k}} ;
```

A key property of wavelets essential to this work is that a sufficiently smooth function $B(s)$, when expressed in a wavelet basis (Equation 4) will have many small coefficients. By ignoring these negligible coefficients we are left with a sparse, approximate representation. The negligible coefficients occur because wavelet functions have *vanishing moments*. We say that a function $\psi(s)$ has M vanishing moments if

$$\int ds\, \psi(s) s^i = 0, \quad i = 0, \ldots, M-1$$

The Haar wavelet (Figure 2) has one vanishing moment, thus the projection of a nearly constant function into the Haar basis will have wavelet coefficients near 0. Similarly, if a wavelet basis function has two vanishing moments, the projection of a linear function will vanish. Figures 3 and 4 show examples of wavelets, ψ, with two vanishing moments.

5 Wavelets In Higher Dimensions

Wavelet bases for functions of two or more variables are required for radiosity. Our goal is to project the kernel, which is a four dimensional function, into a basis set in which it has a sparse representation.

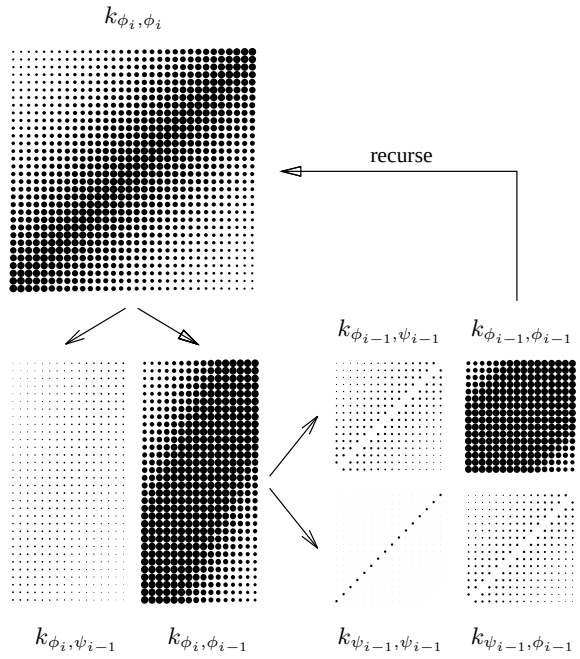

Figure 5: The 2D Pyramid Algorithm is applied to form factors taken from the flatland radiosity environment consisting of two parallel line segments. (Flatland [13] is radiosity in a plane). The dot size indicates the magnitude of a given entry in the matrix.

An arbitrary function $k(s,t)$ of two variables on a finite two dimensional interval can be approximated by some function $\hat{k}(s,t)$ that lies in a two variable finite dimensional function space. Given a particular one dimensional wavelet, a 2D wavelet basis[4] is made up of the functions

$$\begin{aligned} &\phi_0(s)\phi_0(t) \\ &\psi_{i,j}(s)\psi_{i,k}(t) \\ &\psi_{i,j}(s)\phi_{i,k}(t) \\ &\phi_{i,j}(s)\psi_{i,k}(t) \end{aligned}$$

where we only couple functions on the same scale i, where $i = 0, \ldots, L-1$ and $j, k = 0, \ldots, 2^i - 1$.

The 2D wavelet coefficients may be obtained from the finest resolution coefficients $B_{\phi_{L,j},\phi_{L,k}}$ using a 2D **PyramidUp** algorithm. This algorithm begins with the $B_{\phi_{L,j},\phi_{L,k}}$ written in a 2D matrix tableau. It then applies **XformUp** once to each row, followed by an application of **XformUp** to each resulting column. This procedure is applied recursively to the $B_{\phi_{L-1,j},\phi_{L-1,k}}$ quar-

[4] Another 2D wavelet basis could be constructed from the tensor product of a 1D wavelet basis. The different forms of multidimensional wavelet bases are discussed in [3, 20].

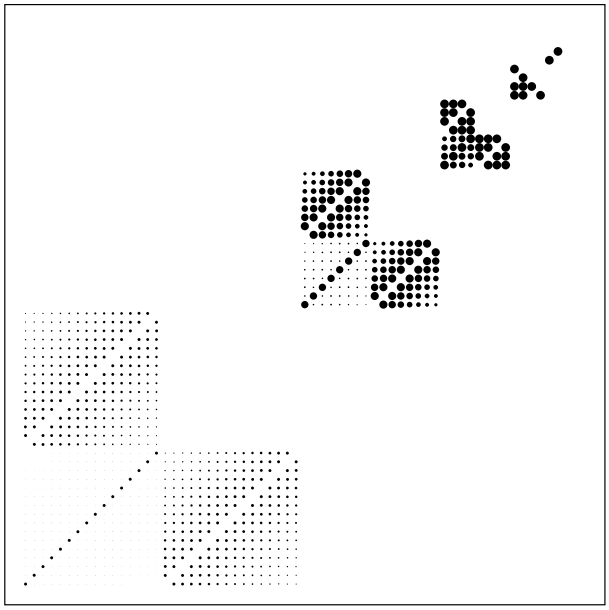

Figure 6: To illustrate the sparseness of the kernel matrix we transform the flatland radiosity matrix from Figure 5 into the 2D Haar basis. Many of the coefficients are small in magnitude (small dots).

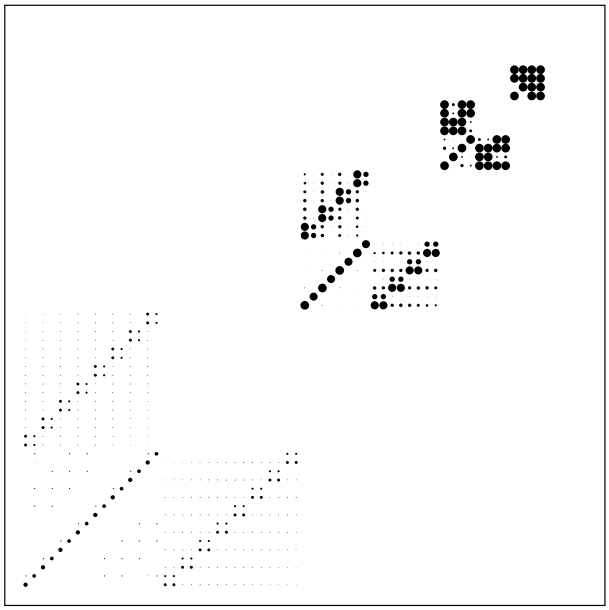

Figure 7: We transform the same matrix into the $\mathcal{F}_2$ basis. Notice that even more of the coefficients are negligible now.

ter (Figure 5). The construction of a 2D **`PyramidDown`** follows analogously from the one dimensional **`PyramidDown`**.

This construction can be extended to functions of four variables such as the kernel in 3D radiosity $k(s_1, t_1, s_2, t_2)$. For this case, there are sixteen combinations of ϕ and ψ functions in four variables. The basis is made up of all fifteen combinations on the same scale i which involve ψ functions. The corresponding pyramid transformation functions are constructed as in the two dimensional case by applying **`XformUp`** and **`XformDown`** respectively to each dimension in turn.

For this type of multidimensional wavelet basis Beylkin et al. [3] show that for a given error tolerance, only $O(n)$ coefficients need to be used to attain the prescribed error tolerance in the results of our computations. Figures 6 and 7 visualize the sparseness of a flatland radiosity kernel when written in two wavelet bases with one and two vanishing moments respectively.

6 Radiosity with Wavelets

To obtain an efficient radiosity algorithm, we project the kernel by taking inner products with the wavelet basis functions. The coefficients of the kernel with respect to the basis are given by

$$\begin{aligned} k^{\phi} = k_{\phi_0,\phi_0} &= \int dt \int ds\, k(s,t)\phi_0(s)\phi_0(t) \\ k^{\alpha}_{ijk} = k_{\psi_{i,j},\psi_{i,k}} &= \int dt \int ds\, k(s,t)\psi_{i,j}(s)\psi_{i,k}(t) \\ k^{\beta}_{ijk} = k_{\phi_{i,j},\psi_{i,k}} &= \int dt \int ds\, k(s,t)\phi_{i,j}(s)\psi_{i,k}(t) \\ k^{\gamma}_{ijk} = k_{\psi_{i,j},\phi_{i,k}} &= \int dt \int ds\, k(s,t)\psi_{i,j}(s)\phi_{i,k}(t) \end{aligned}$$

Because of the vanishing moment properties of the wavelets and the smoothness properties of the kernel, many of these terms are nearly zero.

A projected version of the integral operator can now be derived by projecting the kernel itself. This derivation which we only sketch here is described in greater detail in Beylkin et al. [3]. The k^{α}, k^{β} and k^{γ} coefficients are used to represent the kernel which has been approximated with respect to the wavelet basis. Given this projection, after performing the necessary algebra, the approximate operator can be written as

$$\begin{aligned} &\int dt\, \hat{k}(s,t)B(t) = \\ &\quad B^{\phi}k^{\phi}\phi_0(s) + \sum_{ij}\Big(\sum_{k} B^{\alpha}_{ik}k^{\alpha}_{ijk}\Big)\psi_{i,j}(s) \\ &\quad + \sum_{ij}\Big(\sum_{k} B^{\beta}_{ik}k^{\beta}_{ijk}\Big)\phi_{i,j}(s) + \sum_{ij}\Big(\sum_{k} B^{\gamma}_{ik}k^{\gamma}_{ijk}\Big)\psi_{i,j}(s) \end{aligned} \tag{5}$$

where

$$\begin{aligned} B^{\alpha}_{ik} = B^{\gamma}_{ik} = B_{\psi_{i,k}} &= \int dt\, \psi_{i,k}(t)B(t) \\ B^{\beta}_{ik} = B_{\phi_{i,k}} &= \int dt\, \phi_{i,k}(t)B(t) \\ B^{\phi} = B_{\phi_0} &= \int dt\, \phi_0(t)B(t) \end{aligned}$$

6.1 The Basic Algorithm

Equation 5 suggests the following three phase algorithm to approximate the kernel operating on a radiosity function.

Step 1 *Pull*: Obtain the n (n = number of bases of the radiosity function) coefficients B^{α} and the n coefficients B^{β} of the radiosity function. If we are initially given the coefficients $B_{\phi_{L,j}}$, the $2n$ needed coefficients can be obtained by calling a procedure **`Pull`** which is just like **`PyramidUp`** except it returns *both* the ϕ and ψ coefficients. This step transforms n coefficients into $2n$

coefficients. A 1D **Pull** would then be

```
Pull( vector B_{φ_{L,k}} )
  for( i = L; i > 0; i-- )
    (B_{φ_{i-1,k}}, B_{ψ_{i-1,k}}) = XformUp( B_{φ_{i,k}} , i );
  return ( B_{φ_{i,k}}, B_{ψ_{i,k}}, i = 0, ..., L-1 ) ;
```

Step 2 *Gather*: Let the projected kernel operate on the projected radiosity function. This means that we sum over the index k, and is equivalent to a matrix multiply. Because of the vanishing moments of the wavelet functions most of the n^2 kernel coefficients will be near zero and may be ignored if the action of the kernel is desired to finite precision. The procedure **Gather** results in $2n$ coefficients $G_{\phi_{i,j}}$ and $G_{\psi_{i,j}}$ that represent the resultant radiosity function as a combination of $\phi_{i,j}(s)$ and $\psi_{i,j}(s)$.

Step 3 *Push*: Reconstruction of the radiosity function using the $2n$ functions $\phi_{i,j}(s)$ and $\psi_{i,j}(s)$ is done with the procedure **Push** which is similar to **PyramidDown** but takes as arguments *both* the ϕ and ψ coefficients. A 1D **Push** would then be

```
Push( B_{φ_{i,k}}, B_{ψ_{i,k}}, i = 0, ..., L-1)
  for( i = 0; i < L; i++ )
    B_{φ_{i+1,k}} += XformDown( B_{φ_{i,k}}, B_{ψ_{i,k}} , i );
  return B_{φ_{L,k}} ;
```

Wrapping this projected operator within a Jacobi iteration loop results in the following algorithm

```
(k^α, k^β, k^γ) = ProjectKernel();
B_{φ_{L,k}} = E_{φ_{L,k}};
while( !converged )
  G = 0;
  (B_{φ_{i,k}}, B_{ψ_{i,k}}) = Pull( B_{φ_{L,k}} );
  (G_{φ_{i,j}}, G_{ψ_{i,j}}) = Gather( B_{φ_{i,k}}, B_{ψ_{i,k}}, k^α, k^β, k^γ );
  G_{φ_{L,k}} = Push( G_{φ_{i,j}}, G_{ψ_{i,j}} );
  B_{φ_{L,k}} = G_{φ_{L,k}} + E_{φ_{L,k}};
Display();
```

The push and pull can be done in $O(n)$ (linear in the number of elements) steps. The gather step (this is a complete gather sweep which updates all of the entries) can be done in $O(m)$ time where m is the number of terms in the kernel expansion (matrix) that are significant. We want m to be as small as possible. Wavelet bases will lead to $m = O(n)$ where the constant factor in $O(n)$ decreases with the number of vanishing moments.

What remains is to project the kernel into the wavelet basis, which may be done as follows

```
ProjectKernel()
  k_{φ_{L,j},φ_{L,k}} = Quadrature( k, φ_{L,j}, φ_{L,k} );
  (k^α, k^β, k^γ) = PyramidUp( k_{φ_{L,j},φ_{L,k}} );
  where( (k^α, k^β, k^γ) < ε )
    (k^α, k^β, k^γ) = 0 ;
```

6.2 The Top Down Approach

Unfortunately, this bottom up **ProjectKernel** is an expensive implementation requiring quadratic time and space. The costs can be dramatically cut by using an *oracle* which predicts which m of the n^2 coefficients of the projected kernel are significant. Then, these m values are computed directly by quadrature or symbolic integration.

Assuming that the oracle can estimate the smoothness of the kernel for a given region (vis-a-vis a given number of vanishing moments), an efficient top down recursive version of **ProjectKernel** can be written as follows

```
ProjectKernel( i, patch p, patch q )
  smooth = AskOracle( p, q );
  if( smooth ) return;
  else
    (k^α_{i,j(p),k(q)}, k^β_{i,j(p),k(q)}, k^γ_{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) );
```

If the oracle finds the region under consideration sufficiently smooth no more recursive calls need be executed, since the coefficients at lower levels will be insignificant by assumption. The function **Quadrature()** computes the projection of the kernel function onto the basis functions at the given level.

6.3 3D Radiosity

In 3D radiosity B is a function of two variables so in the main program we use a 2D **Pull** and a 2D **Push** respectively. k is a function of four variables so in the bottom up **ProjectKernel** we use a 4D **PyramidUp** function. In the top down approach to **ProjectKernel** there are fifteen not three quadratures and sixteen recursive calls for all combinations of four children of **p** and **q**.

7 Implementation

The top down algorithm described above has been implemented by extending the implementation of hierarchical radiosity described in Hanrahan et al. [11].

7.1 Choice of Basis

Two families of wavelets have been explored, multiwavelets [1] and a family of wavelets that we call flatlets. Each of these families have members with any number of vanishing moments.

The construction of $\mathcal{M}_M$ (multiwavelet with M vanishing moments) begins with M smooth functions which are the first M Legendre Polynomials, $\phi^m(s) = L_m(s)$, and M detail functions $\psi^m(s)$ that are piecewise polynomials of degree $M-1$, and have M vanishing moments. A hierarchy is then constructed from these shapes. $\mathcal{M}_1$ is the Haar basis, however, for M greater than 1, $\mathcal{M}_M$ is technically speaking not a true wavelet since it begins with a collection of ϕ and ψ functions instead of a single pair.

Multiwavelets form an orthonormal basis. Figure 3 shows the basis functions for the $\mathcal{M}_2$ hierarchy. The two-scale relationship for $\mathcal{M}_2$ is expressed concisely as

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

Using this relationship the push and pull operations can be computed using a binary tree, instead of as a subsampled vector convolution. A node stores the four coefficients of the functions $\phi^1_{i-1,j}$, $\phi^2_{i-1,j}$, $\psi^1_{i-1,j}$, $\psi^2_{i-1,j}$. During a pull, a node computes the values of its coefficients as a linear combination of the $\phi^1_{i,2j}$, $\phi^2_{i,2j}$ coefficients obtained from its left child, and the $\phi^1_{i,2j+1}$, $\phi^2_{i,2j+1}$ coefficients obtained from its right child. To represent the radiosity function over a patch we need a 2D $\mathcal{M}_2$ basis for which we use

a quad-tree where each node stores sixteen coefficients. During a pull, a node computes its coefficients as a linear combination of the sixteen $\phi\phi$ coefficients from its children (four from each child).

The flatlet basis $\mathcal{F}_M$ is made up entirely of piecewise constant functions. The ϕ^m are M adjacent box functions, and the ψ^m are M piecewise constant functions that have M vanishing moments. Figure 4 shows the $\mathcal{F}_2$ hierarchy.

For $\mathcal{F}_2$, the two-scale relationship is given by

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 3 & -3 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} \phi^1_{i,2j} \\ \phi^2_{i,2j} \\ \phi^1_{i,2j+1} \\ \phi^2_{i,2j+1} \end{bmatrix} = \begin{bmatrix} \phi^1_{i-1,j} \\ \phi^2_{i-1,j} \\ \psi^1_{i-1,j} \\ \psi^2_{i-1,j} \end{bmatrix} \quad (6)$$

The top two rows of the matrix in the above equation are chosen to give us box functions twice as wide. The bottom two rows are chosen to be orthogonal to constant and linear variation, (the vectors $[1,1,1,1]$, $[0,1,2,3]$)[5]. For a discussion of a similar construction see [2].

Both flatlets and multiwavelets can be constructed to have any number of vanishing moments to increase the sparseness of the integral operator representation. For both bases, the case $M = 1$ reduces to the Haar basis. For $M > 1$ multiwavelets offer the benefits of projecting into a higher order space, resulting in increased convergence rates and smoother basis functions to represent the answer. These benefits come at the expense of higher order quadratures necessary for the inner products. Flatlets for $M > 1$ also offer accelerated convergence while the quadratures remain equivalent to form factor computations for which there exits a large body of literature and code, and for which some closed form solutions are known. The final answer is still represented as a piecewise constant function, albeit at the finest resolution $\phi_{L,j}$. Since the degree of the basis functions does not go up in the flatlet case the width of support needs to be increased as M increases.

With multiwavelets and flatlets there is also a cost incurred by increasing the number of vanishing moments. Larger M will result in h and g filters with wider support. Thus any non-smoothness in $k(s,t)$, such as a shadow discontinuity, will fall under the support of more basis functions. This increases the number of significant terms in the integral operator.

[5] A technical detail concerns the fact that flatlets for $M > 1$ are not orthonormal and thus require the dual basis functions to compute `PyramidUp` (see [20]).

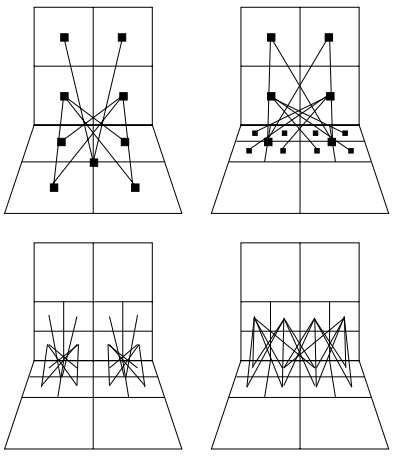

Figure 8: Two different oracles and the interaction patterns they generate.

7.2 Pull, Push and Gather

Both multiwavelets and flatlets are instances of *tree* wavelets. A tree wavelet has the property that the convolution sequences h and g for two neighboring elements do not overlap. This property allows us to organize all computations along a tree which does not need to have uniform depth. Tree wavelets also allow for another simplification. Since all necessary coefficients reside in the immediate children of a node we can use the two-scale relationship to store only the $\phi\phi$ coefficients and need not represent the $\phi\psi$, $\psi\phi$, and $\psi\psi$ coefficients explicitly. With this simplification `ProjectKernel` is implemented as follows

```
ProjectKernel( i, patch p, patch q )
  ParentLevelsmooth = AskOracle( p, q );
  if( ParentLevelsmooth || i == L )
    k_{φ,φ} = Quadrature( k, p, q );
    CreateLink( k_{φ,φ}, 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) );
```

In our implementation of radiosity using the $\mathcal{M}_M$ and $\mathcal{F}_M$ bases, the radiosity function over each polygon is represented by $B_{\phi\phi}$ coefficients that are stored in a quad-tree. Each node holds M^2 $B_{\phi\phi}$ coefficients. Pulling and pushing are done in the quad-tree as in [11] except that for different bases, we use different two-scale relationships. The kernel is represented by its $k_{\phi\phi\phi\phi}$ coefficients that are stored on links created between nodes of different polygons' quad-trees. Each such link caries M^4 interaction terms. For the $\mathcal{F}_M$ bases the interaction terms are still form factors, but for $\mathcal{M}_M$ the coefficients on the links represent higher order interactions which require quadrature computations of the appropriate order. Gathering is done by moving B values across the links, weighted by the k values on the link. In this context, HR can be viewed as wavelet radiosity using the Haar basis.

7.3 Oracle

The oracle must decide whether the kernel is sufficiently smooth over two patches in the environment i.e., resembles a polynomial of degree $M-1$ or less. If the kernel is smooth, all ψ terms will (sufficiently) vanish and thus any work to evaluate the lower interaction terms can be avoided.

The most accurate approach to measure the kernel smoothness is to directly evaluate the integrals of the kernel against the ψ on this and all lower levels and verify that they are below the required threshold. This is computationally too expensive and we approximate this computation in the following way. The kernel is sampled at the points required by a Gauss-Legendre quadrature rule of the appropriate order and an interpolating polynomial of degree $M-1$ is constructed using Neville's algorithm [22]. Given this interpolating polynomial k_P we compute the L_1 error $\int |k_P - k|$ with a quadrature rule which places sample points in-between the previously chosen points. If the value of this integral is small we conclude that our current level of (smooth) approximation matches the kernel function well and the `AskOracle` function returns `True`. Note that the sample points for the interpolating polynomial are chosen so that they can be used directly in the computation of the interaction link values. If the `AskOracle` function returns `False` these samples are discarded. A less costly approach could use geometric information, such as the size, orientation, and distance between two patches. In effect this was done in the original HR implementation. However for the $\mathcal{F}_M$ and $\mathcal{M}_M$, $M > 1$ bases it is not immediately clear what the corresponding geometric reasoning would be.

It is important to realize that any such implementation of an oracle will introduce errors due to its approximate nature. If the oracle is not stringent enough, and necessary terms are neglected, artifacts will appear in the image. Figure 8 shows two different oracles and the interactions they force. Two successive levels of interactions are shown (top to bottom). On the left is an oracle allowing patches close to the singularity (where the kernel varies rapidly) to be linked (meaning no further subdivision will be done). For this oracle the interaction patterns separate on the lower level. On the right is a more stringent oracle which does not allow singular interactions until patches have become very small. As a result we do not see the separation.

As in [11] we use *brightness refinement* which means that the stringency of the oracle is weighted by the brightness of the involved patches. Also as in [11] a fast partial visibility test is performed by using a constant number of jittered rays. If two patches are partially occluded and there is sufficient energy being transferred between the two patches the oracle returns **`False`**.

7.4 Quadrature

If the oracle returns **`True`**, numerical integrations must be performed to compute the $k_{\phi\phi\phi\phi}$ terms associated with the link to be created. Our implementation uses Gauss-Legendre quadrature [22] for this purpose. A Gauss-Legendre quadrature rule provides an accurate integration for polynomials up to order $2p-1$, where p is the number of sample points. The order of the quadrature and the related number of sample points required depends on the sum of the order of the wavelet bases, and the assumed order of the kernel itself.

For the projection of the kernel against a flatlet basis, a two point rule is used for each constant section of the basis function. In the case of multiwavelets $\mathcal{M}_M$, $M > 1$, M points are chosen along each coordinate axis since we need to have a high enough order of integration to account for the polynomial variance in the kernel and the polynomial basis functions themselves. For example, for $M = 3$ we compute coefficients when the kernel varies approximately up to 2^{nd} by projecting onto basis functions up to 2^{nd} order. Thus the integrand is approximately 4^{th} order, and we can use a three point Gauss rule.

The number of integrals which need to be computed for a link is M^4, however for all these integrals only a total of M^4 samples of the kernel function are required. Using precomputed weights, these samples are combined to give all the desired integrals.

We treat visibility following [11] by casting a constant number of jittered rays between two patches to estimate the fraction of visibility. This is then used to attenuate the quantity returned by the Gauss-Legendre quadrature. This technique relies on the fact that we always subdivide in the vicinity of a shadow discontinuity limiting errors due to the non-smooth nature of the kernel to a small region.

When the two patches that are linked up are close to the singularity in k, quadratures will encounter numerical difficulties if they are not properly adapted to the singularity. In particular a Gauss-Legendre rule will produce large errors and an adapted quadrature rule is required. This phenomenon is not unique to wavelet radiosity but applies to all GR methods. Special Gauss rules can be designed for the particular singularity found in the radiosity kernel. Zatz [25] uses such custom rules and notes the need for an automatic decision procedure as to when to switch the type of integration. In our implementation of flatlets, we use a closed form solution for the form factor [21] whenever the patches border on the singularity. While this computation is expensive, it only needs to be invoked in a small fraction of interaction computations and contributes little to overall runtime. For multiwavelets we have no such closed form available. In this case the oracle forces subdivision to small enough patches at the singularity that the resulting errors contribute very little to the overall error. Alternative constructions for singular transports are discussed in [19].

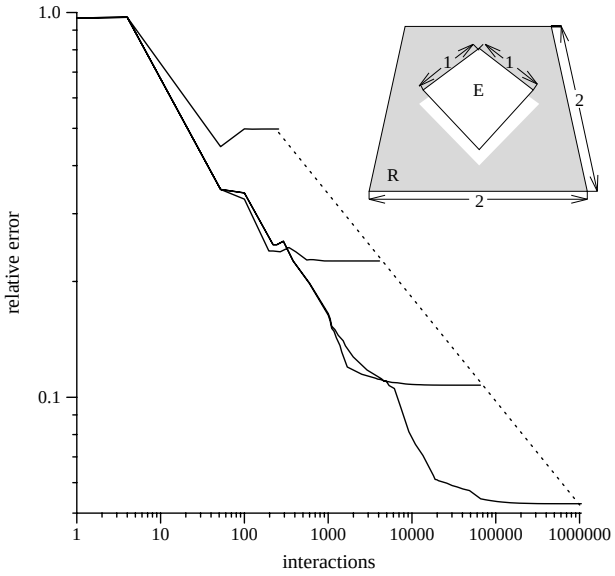

Figure 9: Relative L_1 error as a function of the number of interaction links for the haar basis with $h = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ (top to bottom). The test configuration is depicted in the upper right corner.

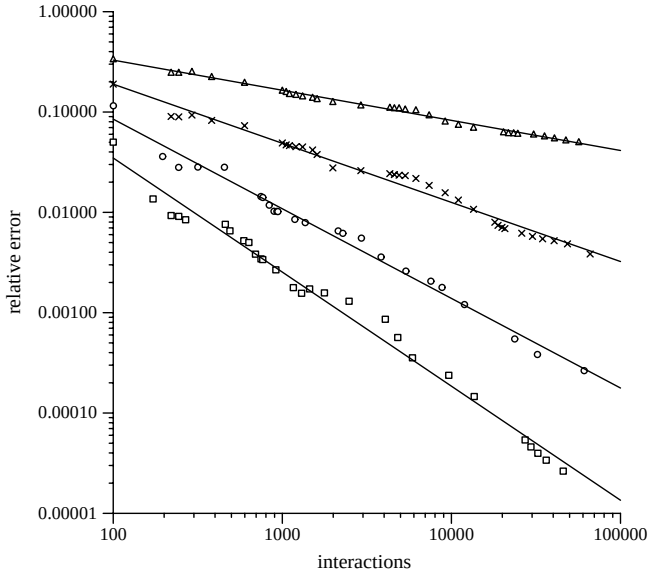

Figure 10: Relative L_1 error as a function of the number of interactions for the wavelet bases $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ (top to bottom) using the same test configuration as in Figure 9. Here $h = \frac{1}{32}$.

8 Experimental Results

In this section we present findings that compare how radiosity behaves using different wavelet bases. We give results from the analysis of a simple 3D configuration, for which we have an analytic solution against which to check our results. We finish with an image of a full environment.

One test case used the configuration depicted in the inset in

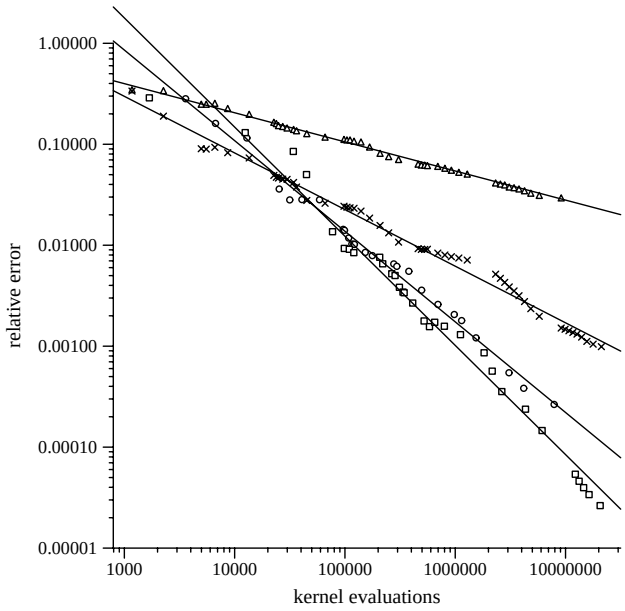

Figure 11: Relative L_1 error as a function of work.

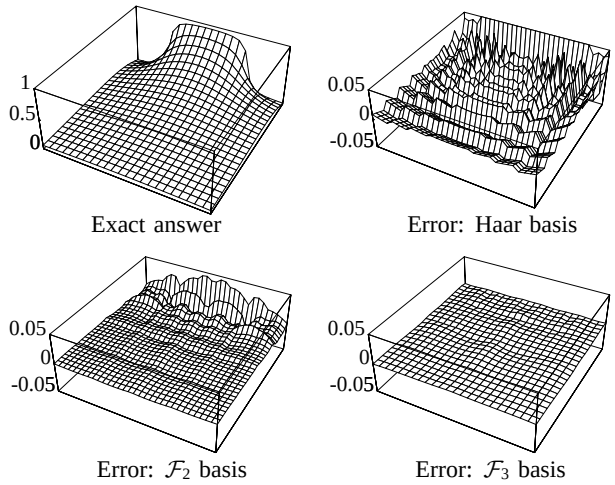

Figure 13: Heightfield error plots for perpendicular emitter and receiver.

Figure 12: 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 we have not performed any post processing such as Gouraud shading.

Figure 9. A pure emitter of side length 1 is placed 0.1 units above a pure receiver of side length two. For this particular configuration the radiosity on the receiver is given by the differential area to finite area form factor at every point. Figure 9 shows the behavior of the relative L_1 error for the Haar basis as a function of the number of interactions for various grid sizes h. The far point on each of the lines corresponds to a full matrix solution. Note in particular that the final accuracy is reached well before all matrix elements are computed. Plots for higher order basis functions exhibit the same overall shape but with steeper slopes and overall lesser error. Figure 10 shows the behavior of the $\mathcal{M}_M$ bases for $M = 1, \ldots, 4$ and $h = \frac{1}{32}$. The ratio of successive slopes (as fitted to the points) is almost precisely $1:2:3:4$, as one would expect from the order of basis functions employed. For both plots we have depicted error as a function of number of interactions. However a user experiences error as a function of work which is more accurately measured by the number of kernel evaluations. Since the amount of work increases for higher order methods it is not clear a priori whether a higher order method will always yield better results in a shorter time. Figure 11 shows error as a function of kernel evaluations for the same data as that used in Figure 10. The plot for $M = 1$ is translated with respect to all others since we always use at least a two point quadrature rule even if the basis functions are constant. The plot shows that if sufficient accuracy is required higher order basis functions achieve lower error for the same amount of work.

Figure 14: Architectural scene computed with the $\mathcal{M}_2$ basis and rendered directly from the basis functions.

We have also examined the behavior of our methods near the singularity of an environment consisting of perpendicular polygons (Figure 12). The emitter was chosen to be half as wide as the receiver to create more variation in the radiosity function. The grid size was set to $\frac{1}{32}$. The upper left plot in Figure 13 shows the exact solution plotted as a height field over the receiver. On the top right is a plot of the difference between exact solution and the computed solution for the Haar basis. On the bottom are similar error surfaces for the $\mathcal{F}_2$ and $\mathcal{F}_3$ bases (left and right respectively). The amount of work was approximately constant (8000 interactions) for all three solutions. The graphs show clearly the lesser and smoother error for the $\mathcal{F}_2$ and $\mathcal{F}_3$ bases demonstrating the effectiveness of bases with more vanishing moments. This is also illustrated by the rendered images in Figure 12.

The algorithm has also been run on a more complex environ-

ment (Figure 14). This picture, as well as Figure 12, does not use any postprocessing such as Gouraud shading. Instead the surface brightness is computed directly from the basis functions and associated coefficients.

9 Conclusion and Future Work

In this paper we have presented the basic theory of projections of integral operators into hierarchical bases, and laid out the theoretical foundation of a new set of techniques involving wavelets. With this in hand, we introduced a new set of linear time algorithms we have called *wavelet radiosity*, and shown that the hierarchical radiosity described by Hanrahan et al. was an instance of a first order wavelet approach.

We have introduced a new family of wavelets, dubbed flatlets and also experimented with a second family of wavelets, multiwavelets. Both lead to efficient algorithms. Future work includes examining various wavelet bases which may have better properties than the multiwavelets and flatlets. For example the Coiflet functions of [7, 3] allow for fast one point quadrature methods. The tree wavelets that we implemented do not enforce any kind of continuity at element boundaries, possibly leading to blocky artifacts. Spline wavelets [4] might provide a basis which would alleviate this.

While our initial implementation was limited to quadrilateral polygons there is nothing in the underlying algorithms that prevents the use of any surface whose parameter domain is rectilinear, such as for example bicubic patches. The only change involves the reparameterization (change of variable) in the coupling integrals. It would be very desirable to design bases which work with triangular domains since triangles are a common primitive in meshing algorithms.

There are still fundamental questions that have yet to be addressed. We would like to gain a better understanding of how wavelet expansions interact with the visibility term in the kernel. It is also important to find methods that remain efficient when the environment consists of a large number of small polygons.

Acknowledgements

The research reported here was partially supported by Apple, Silicon Graphics Computer Systems, and the National Science Foundation (CCR 9207966). We would like to thank S. V. Krishnan for his useful comments. We would also like to thank Ju-sung Lee, Jonathan McAllister, and Michael Neufeld for creating the model of the room.

References

[1] ALPERT, B. A Class of Bases in L^2 for the Sparse Representation of Integral Operators. *SIAM Journal on Mathematical Analysis 24*, 1 (Jan 1993).

[2] ALPERT, B., BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. Wavelet-like Bases for the Fast Solution of Second-kind Integral Equations. *SIAM Journal on Scientific Computing 14*, 1 (Jan 1993).

[3] BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics 44* (1991), 141–183.

[4] CHUI, C. K. *An Introduction to Wavelets*, vol. 1 of *Wavelet Analysis and its Applications*. Academic Press Inc., 1992.

[5] COHEN, M., CHEN, S. E., WALLACE, J. R., AND GREENBERG, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics 22*, 4 (August 1988), 75–84.

[6] COHEN, M. F., AND GREENBERG, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics 19*, 3 (July 1985), 31–40.

[7] DAUBECHIES, I. *Ten Lectures on Wavelets*, vol. 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1992.

[8] DELVES, L. M., AND MOHAMED, J. L. *Computational Methods for Integral Equations*. Cambridge University Press, 1985.

[9] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. Modelling the Interaction of Light between Diffuse Surfaces. *Computer Graphics 18*, 3 (July 1984), 212–222.

[10] GORTLER, S. J., COHEN, M. F., AND SLUSALLEK, P. Radiosity and Relaxation Methods; Progressive Refinement is Southwell Relaxation. Tech. Rep. CS-TR-408-93, Department of Computer Science, Princeton University, February 1993.

[11] HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 25*, 4 (July 1991), 197–206.

[12] HECKBERT, P. S. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, University of California at Berkeley, January 1991.

[13] HECKBERT, P. S. Radiosity in Flatland. *Computer Graphics Forum 2*, 3 (1992), 181–192.

[14] KAJIYA, J. T. The Rendering Equation. *Computer Graphics 20*, 4 (1986), 143–150.

[15] LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. A Discontinuity Meshing Algorithm for Accurate Radiosity. *IEEE CG&A 12*, 4 (July 1992).

[16] MALLAT, S. G. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (July 1989), 674–693.

[17] PRESS, W., TEUKOLSKI, S., VETTERLING, W., AND FLANNERY, B. *Numerical Recipies in C, The Art of Scientific Computing*, 2 ed. Cambridge University Press, 1992.

[18] SALESIN, D., LISCHINSKI, D., AND DEROSE, T. Reconstructing Illumination Functions with Selected Discontinuities. *Third Eurographics Workshop on Rendering* (1992), 99–112.

[19] SCHRÖDER, P. Numerical Integration for Radiosity in the Presence of Singularities. In *Fourth Eurographics Workshop on Rendering* (1993).

[20] SCHRÖDER, P., GORTLER, S. J., COHEN, M. F., AND HANRAHAN, P. Wavelet Projections For Radiosity. In *Fourth Eurographics Workshop on Rendering* (June 1993).

[21] SCHRÖDER, P., AND HANRAHAN, P. On The Form Factor Between Two Polygons. In *Computer Graphics, Annual Conference Series, 1003* (August 1993), Siggraph.

[22] STOER, J., AND BULIRSCH, R. *Introduction to Numerical Analysis*. Springer Verlag, New York, 1980.

[23] SZELISKI, R. Fast Surface Interpolation Using Hierarchical Basis Functions. *IEEE Trans. PAMI 12*, 6 (June 1990), 513–439.

[24] YSERENTANT, H. On the Multi-level Splitting of Finite Element Spaces. *Numerische Mathematik 49* (1986), 379–412.

[25] ZATZ, H. R. Galerkin Radiosity: A Higher-order Solution Method for Global Illumination. In *Computer Graphics, Annual Conference Series, 1003* (August 1993), Siggraph.