Triple Product Wavelet Integrals for All-Frequency Relighting

Ren Ng
Stanford University

Ravi Ramamoorthi
Columbia University

Pat Hanrahan
Stanford University

Abstract

This paper focuses on efficient rendering based on pre-computed light transport, with realistic materials and shadows under all-frequency direct lighting such as environment maps. The basic difficulty is representation and computation in the 6D space of light direction, view direction, and surface position. While image-based and synthetic methods for real-time rendering have been proposed, they do not scale to high sampling rates with variation of both lighting and viewpoint. Current approaches are therefore limited to lower dimensionality (only lighting or viewpoint variation, not both) or lower sampling rates (low frequency lighting and materials). We propose a new mathematical and computational analysis of pre-computed light transport. We use *factored* forms, separately pre-computing and representing visibility and material properties. Rendering then requires computing *triple product integrals* at each vertex, involving the lighting, visibility and BRDF. Our main contribution is a general analysis of these triple product integrals, which are likely to have broad applicability in computer graphics and numerical analysis. We first determine the computational complexity in a number of bases like point samples, spherical harmonics and wavelets. We then give efficient linear and sublinear-time algorithms for Haar wavelets, incorporating non-linear wavelet approximation of lighting and BRDFs. Practically, we demonstrate rendering of images under new lighting and viewing conditions in a few seconds, significantly faster than previous techniques.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; G.1.2 [Numerical Analysis]: Approximation—Wavelets and Fractals, Nonlinear Approximation

Keywords: Relighting, Pre-computed Radiance Transfer, Image-Based Rendering, Haar Wavelets, Non-linear Approximation

1 Introduction

Detailed natural lighting, realistic materials, and intricate soft shadowing are important effects in realistic computer graphics, as shown in Figure 1. However, computer-generated images, especially in interactive applications like modeling packages or electronic commerce, often omit view-dependent effects such as the interaction between shadows and glossy reflections. Conventional rendering can take hours, greatly increasing the design cycle in applications like lighting design. In this paper we take a step towards interactive manipulation of lighting and viewpoint, allowing creation of realistic images in a few seconds, which is orders of magnitude faster than previous methods.

Our approach is based on pre-computing information about a static scene, followed by real-time rendering with dynamically varying viewpoint and environment map lighting. This basic approach was originally introduced by Nimeroff et al. [1994] and Dorsey et al. [1995], and has led to much recent work [Sloan et al. 2002; Ng et al. 2003; Sloan et al. 2003a; Sloan et al. 2003b]. Our approach is also related to image-based rendering methods, where the pre-computation is usually replaced by acquisition of real images. All of these methods encounter a fundamental problem, in that they exist in a 6D space (light direction, view direction and surface position are all 2D) that must be densely sampled for intricate shadows and specularities. A sampling rate of 100 in each dimension would require a trillion (10^{12}) total samples—intractable to pre-compute, store or relight with.

Figure 1: *A 300,000 vertex scene lit by a $6 \times 64 \times 64$ environment map. We render this scene in 3–5 seconds with dynamic lighting and camera position. Figure 5 illustrates shadow movement with changing view.*

A number of interactive and even real-time algorithms have been developed in previous work. The majority address the 6D problem by using very low sampling rates, such as the low frequency lighting and materials used by Sloan et al. [2002; 2003a; 2003b]. This approach is relatively fast and compact, but it blurs lighting details by band-limiting the illumination. A different approach is to simply ignore some of the six dimensions, to allow for high sampling. For instance, Ng et al. [2003] eliminate view variation by fixing the camera. Coupled with a wavelet lighting approximation, they allow all-frequency relighting from a $6 \times 64 \times 64$ cubemap—a thousand times higher resolution than the 25 term spherical harmonic approximations of Sloan et al. Such a high resolution resolves lighting effects that are blurred out with low-frequency illumination, such as sharp shadows and specularities.

In this paper, our goal is to render these rich lighting effects, but with changing view. We develop a new mathematical and computational analysis of pre-computed light transport to overcome the difficulties with high dimensionality and sampling rate inherent in previous methods. Our specific contributions are:

Efficient Re-Rendering from Compact Factored Forms: We focus on *factored representations*, where visibility and materials (BRDF) are each pre-computed and represented separately in an appropriate basis. Each of these datasets is 4D rather than 6D, enabling us to handle both lighting and view variation with essentially the same pre-computation time and storage as previous fixed-view approaches [Ng et al. 2003]. The conceptual notion bears similarities to factorizations of the light transport operator for global illumination [Arvo et al. 1994], and has also been appreciated by Sloan et al. [2002] who propose (but do not implement) using Clebsch-Gordan coefficients [Inui et al. 1990] with spherical harmonics. Factorizations have also been proposed for compression in BRDF representation, such as in the work of Kautz and McCool [1999].

Triple Product Integral Representations: Our main contribution is introducing the general study of the *triple product integrals* that must be computed in relighting from factored forms. Specifically, at each vertex, we must compute an integral (over the incident hemisphere) of three factors: the lighting, visibility and BRDF, each of which is represented in basis functions such as spherical harmonics or wavelets.

Similar triple product integrals also arise in many other applications in computer graphics and applied mathematics. For instance, they arise whenever we multiply together two functions, each represented in a basis like Fourier or wavelets, and express the result also in the same basis. Three examples in graphics are multiplying two images for image processing or compositing, multiplying illumination and reflectance spectra for high fidelity color reproduction, and multiplying texture or emission maps and irradiance in a radiosity solver [Gershbein et al. 1994]. Our mathematical analysis of triple products is valid for any application, domain and choice of basis functions, and is likely to be broadly relevant to problems in computer graphics, numerical analysis and signal processing.

Analysis and Algorithms: Double product integrals are relatively simple because they reduce to a dot product of the basis coefficients, as exploited in previous work [Sloan et al. 2002; Ng et al. 2003]. However, triple product integrals are considerably more complicated. In spherical harmonics, a framework of Clebsch-Gordan coefficients has been devised in the context of angular momentum in quantum mechanics [Inui et al. 1990], but the focus has been on analytic formulae for low orders rather than efficient evaluation of all frequency effects. One recent application of Clebsch-Gordan coefficients is the work of Thornber and Jacobs [2001]. They derive formulae for the spherical harmonic coefficients of glossy reflection functions, taking attached shadows into account. We study the computational complexity of evaluating triple product integrals in a number of bases. We then develop an efficient sublinear computation algorithm in Haar wavelets. Finally, we present a practical implementation of efficient rendering using wavelet triple product integrals, allowing us to create new images with arbitrary all-frequency lighting (up to $6 \times 256 \times 256$ cubemaps), materials and viewpoint in a few seconds.

Readers who are primarily interested in relighting may wish to read about our implementation and results first (sections 7 and 8), and return to earlier sections for mathematical detail as needed.

2 Related Work

We are not aware of any significant previous work on analyzing general triple product integrals in graphics or applied mathematics. However, numerical analysis of their falloff in wavelet bases is related to the multilinear operator results of Meyer et al. [1997]. Good introductions to wavelets and the non-linear approximations that we use are given by Mallat [1999] and DeVore [1998].

In terms of our practical application, the most closely related works are current approaches for real-time rendering using pre-computed light transport. While these methods report higher frame rates than we do, note that they work only at low frequencies or for only lighting or view variation. In fact, it is impractical to scale them to our resolutions and dimensionality. A simple extension of Ng et al. [2003] to view variation would be to sample lighting variation for a number of views, and interpolate between the closest in a manner analogous to Lehtinen and Kautz [2003]. However, we would need thousands of views to capture high-frequency specularities, and relighting from a single view already requires hours of pre-computation and a significant amount of memory.

The situation is even worse for spherical harmonic methods [Sloan et al. 2002; Kautz et al. 2002]. The cost per vertex increases quadratically with resolution, since both lighting and view can vary. Since our goal is approximately a thousand to ten thousand times higher resolution than Sloan et al., that approach would become a million times slower, and require an impractical amount of storage and pre-computation. Note that any compression algorithm, such as clustered PCA [Sloan et al. 2003a] must operate on the full dataset. For a resolution of $25{,}000 \times 25{,}000$, this corresponds to a billion-element vector at each vertex.

Finally, there has also recently been interest in accelerating standard Monte Carlo sampling methods for environment maps. Agarwal et al. [2003] reduce the environment to hundreds of directional light sources and the total rendering time to a few minutes. It is possible that our analysis could be used in future to further accelerate Monte Carlo techniques.

3 Mathematical Framework

We introduce our framework and notation, deriving the key triple product integrals that we analyze from the reflection equation. The next section analyzes the complexity of estimating these integrals.

We start with the reflection equation with direct lighting,

$$B(\mathbf{x}, \boldsymbol{\omega_o}) = \int_{\Omega_{2\pi}} L(\mathbf{x}, \boldsymbol{\omega}) \rho(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega_o}) V(\mathbf{x}, \boldsymbol{\omega}) (\boldsymbol{\omega} \cdot \mathbf{n}) \, d\omega, \quad (1)$$

where B is the reflected light, a function of position $\mathbf{x}$ and outgoing direction $\boldsymbol{\omega_o}$, L is the lighting, $\boldsymbol{\omega}$ is the incident direction, ρ is the BRDF, V is the binary visibility, and $\mathbf{n}$ is the surface normal, with the integral over the visible hemisphere.

We now make a number of simplifications in nomenclature. First, we will incorporate the cosine term $\boldsymbol{\omega} \cdot \mathbf{n}$ in the BRDF definition. Second, we will allow the integrals to be over the full sphere, with the BRDF set to 0 in the lower hemisphere. Third, we will assume the lighting is distant (an environment map, as in most previous work), and so depends only on $\boldsymbol{\omega}$. Fourth, we assume a uniform BRDF over the surface. We can always multiply through later by the albedo and handle a linear combination of basic BRDFs. Finally, we assume for simplicity that both $\boldsymbol{\omega}$ and $\boldsymbol{\omega_o}$ are expressed in a global coordinate frame. Technically, we must rotate the BRDF, which is a local function of the surface, by the surface normal to align with the global reference frame. For simplicity in exposition, we will ignore this issue for now, returning to it when we discuss our actual implementation. To clarify that the BRDF must be rotated according to the surface normal, we denote it as $\tilde{\rho}$ and write

$$B(\mathbf{x}, \boldsymbol{\omega_o}) = \int_{S^2} L(\boldsymbol{\omega}) \tilde{\rho}(\boldsymbol{\omega}, \boldsymbol{\omega_o}) V(\mathbf{x}, \boldsymbol{\omega}) \, d\omega, \quad (2)$$

where the integrand is explicitly a product of three factors—the lighting, visibility and BRDF. We now consider a given vertex, with a fixed value for $\mathbf{x}$ and $\boldsymbol{\omega_o}$. In that case, we obtain the simple form,

$$B = \int_{S^2} L(\boldsymbol{\omega}) \tilde{\rho}(\boldsymbol{\omega}) V(\boldsymbol{\omega}) \, d\omega, \quad (3)$$

where the integrand factors implicitly depend on surface location $\mathbf{x}$, normal $\mathbf{n}$, and viewing direction $\boldsymbol{\omega_o}$. Equation 3 corresponds to the basic triple product integral that we will study. Note that while we have derived it using spherical integration in the context of light transport, the basic ideas and form of Equation 3 apply over any domain. Before discussing it in more detail, we first review the double product integral approximations used in previous methods.

3.1 Double Product Integral Computation

One general approach is to define a transport operator $T(\boldsymbol{\omega}) = \tilde{\rho}(\boldsymbol{\omega})V(\boldsymbol{\omega})$, reducing Equation 3 to a double product integral, where the integrand is simply $L(\boldsymbol{\omega})T(\boldsymbol{\omega})$. The functions L and T are then expanded in appropriate orthonormal basis functions $\Psi_i(\boldsymbol{\omega})$,

$$L(\boldsymbol{\omega}) = \sum_i L_i \Psi_i(\boldsymbol{\omega}) \qquad T(\boldsymbol{\omega}) = \sum_j T_j \Psi_j(\boldsymbol{\omega}). \tag{4}$$

We also define the **coupling coefficients**,

$$C_{ij} = \int_{S^2} \Psi_i(\boldsymbol{\omega})\Psi_j(\boldsymbol{\omega})\, d\omega, \tag{5}$$

which are simply $C_{ij} = \delta_{ij}$, Kronecker deltas, for orthonormal basis functions. We now write the reflection integral in terms of basis functions,

$$\begin{aligned} B &= \int_{S^2} \left(\sum_i L_i \Psi_i(\boldsymbol{\omega})\right)\left(\sum_j T_j \Psi_j(\boldsymbol{\omega})\right) d\omega \\ &= \sum_i \sum_j L_i T_j \int_{S^2} \Psi_i(\boldsymbol{\omega})\Psi_j(\boldsymbol{\omega}) d\omega \\ &= \sum_i \sum_j C_{ij} L_i T_j = \sum_i \sum_j \delta_{ij} L_i T_j = \sum_i L_i T_i \\ &= T \cdot L, \end{aligned} \tag{6}$$

i.e. simply a dot product of the coefficient vectors of T and L.

Ng et al. [2003] use Haar wavelets for Ψ, with a non-linear approximation of L, using only the 200 or so largest coefficients. This speeds computation of the above dot product. They fix the view, writing a matrix equation $B = TL$, that compactly denotes the dot product for all image pixels. They also allow viewpoint variation, but only for Lambertian surfaces, since in that case one can ignore the BRDF and still treat the reflection equation as a two-term integral. Sloan et al. [2002] use spherical harmonics for Ψ instead of wavelets, with a linear approximation of only 25 low frequency terms, to make the dot product real time. They also use clustered PCA over similar vertices [Sloan et al. 2003a] to compress T. This allows one to encode the view variation (with $\boldsymbol{\omega_o}$) of T as well, but only for low frequencies.

3.2 Triple Product Integral Computation

We now return to the triple product integrals of Equation 3, with the corresponding basis function expansions,

$$L(\boldsymbol{\omega}) = \sum_i L_i \Psi_i(\boldsymbol{\omega}) \quad V(\boldsymbol{\omega}) = \sum_j V_j \Psi_j(\boldsymbol{\omega}) \quad \tilde{\rho}(\boldsymbol{\omega}) = \sum_k \tilde{\rho}_k \Psi_k(\boldsymbol{\omega}). \tag{7}$$

In analogy with the coupling coefficients for double product integrals, we also define the **tripling coefficients** for triple product integrals,

$$C_{ijk} = \int_{S^2} \Psi_i(\boldsymbol{\omega})\Psi_j(\boldsymbol{\omega})\Psi_k(\boldsymbol{\omega})\, d\omega. \tag{8}$$

In contrast to coupling coefficients, *there is no simple general formula for tripling coefficients*, and there has been relatively little previous work on studying them. For spherical harmonics, the C_{ijk} correspond to Clebsch-Gordan coefficients, whose analytic values, forms and properties are well studied, but not in terms of computational efficiency. In essence, the tripling coefficients for general bases are generalizations of the Clebsch-Gordan coefficients for spherical harmonics.

We now write the reflection equation in terms of basis functions,

$$\begin{aligned} B &= \int_{S^2} \left(\sum_i L_i \Psi_i(\boldsymbol{\omega})\right)\left(\sum_j V_j \Psi_j(\boldsymbol{\omega})\right)\left(\sum_k \tilde{\rho}_k \Psi_k(\boldsymbol{\omega})\right) d\omega \\ &= \sum_i \sum_j \sum_k L_i V_j \tilde{\rho}_k \int_{S^2} \Psi_i(\boldsymbol{\omega})\Psi_j(\boldsymbol{\omega})\Psi_k(\boldsymbol{\omega}) d\omega \\ &= \sum_i \sum_j \sum_k C_{ijk} L_i V_j \tilde{\rho}_k. \end{aligned} \tag{9}$$

Unlike double product integrals, which simply reduce to a dot product of coefficients, we see that triple product integrals are substantially more complicated to estimate. In essence, the rest of this paper is devoted to computing Equation 9 efficiently.

Note that the same machinery is relevant to another basic operation—multiplying together two functions, where both the functions and the result are represented in orthonormal basis functions Ψ. As an example, let us say we wish to compute the product of two images, E and F, with $G = E \cdot F$, where all three images are expanded in an orthonormal basis as in Equation 7. For instance, they may be stored in compressed wavelet form. We can compute the i^{th} basis coefficient for product G by integrating against the i^{th} basis function as follows:

$$\begin{aligned} G_i &= \iint \Psi_i(\mathbf{x})G(\mathbf{x})\, d\mathbf{x} = \iint \Psi_i(\mathbf{x})E(\mathbf{x})F(\mathbf{x})\, d\mathbf{x} \\ &= \iint \Psi_i(\mathbf{x}) \left(\sum_j E_j \Psi_j(\mathbf{x})\right)\left(\sum_k F_k \Psi_k(\mathbf{x})\right) d\mathbf{x} \\ &= \sum_j \sum_k E_j F_k \iint \Psi_i(\mathbf{x})\Psi_j(\mathbf{x})\Psi_k(\mathbf{x})\, d\mathbf{x} \\ G_i &= \sum_j \sum_k C_{ijk} E_j F_k. \end{aligned} \tag{10}$$

We assume that $\mathbf{x}$ is defined on a 2D image domain. The point here is that the final equation is essentially the same form as Equation 9. Note that this relation is generally valid for any application, domain and suitable orthonormal basis expansion—only the *tripling coefficients* C_{ijk} will differ.[1]

4 Analysis of Computational Complexity

We now analyze the computational complexity of the triple product integral approximation, using a number of different functional bases. Readers who are uninterested in the mathematical details may consult the summary of our analysis in section 4.6.

Note that the actual numerical values of the tripling coefficients C_{ijk} can be pre-computed for a given basis (in some cases with a

[1] If Ψ are complex-valued, such as complex exponentials or complex spherical harmonics, it is conventional to define coupling coefficients as $C_{ij} = \int \Psi_i^* \Psi_j$, and tripling coefficients as $C_{ijk} = \int \Psi_i^* \Psi_j \Psi_k$, where one of the terms in the integrand has its complex conjugate taken.

If Ψ are not orthonormal, then we find G_i by integration against $\tilde{\Psi}_i$, the i^{th} *dual* basis function. (In Equation 10, $\tilde{\Psi}_i \equiv \Psi_i$ because the basis is orthonormal.) Thus, in full generality, for multiplication of signals, we are interested in tripling coefficients $C_{ijk} = \int \tilde{\Psi}_i^* \Psi_j \Psi_k$.

simple analytic formula). The difficult part is efficiently computing the sum on the last line of Equation 9. In this section, we will assume a linear approximation to the original functions using N basis functions. Section 5 will discuss a specific efficient algorithm for Haar wavelets, and how non-linear approximation, using only the largest n wavelet terms, can be effectively incorporated.

4.1 General Basis

We first make some general remarks, assuming an arbitrary basis, such as one constructed using principal component analysis. In that case, there will be no special structure to the basis functions, and we can do little better than compute Equation 9 by brute force. The computational complexity will be prohibitively expensive, at $O(N^3)$, since there are N^3 coefficients C_{ijk} (each of i, j, k has N terms). This should be contrasted with the N multiplications and adds of the dot product used in computing double product integrals in general orthonormal basis functions. This highlights the considerably greater difficulties in computing triple product integrals. The efficiencies in the ensuing subsections derive from the special structure of particular bases that make C_{ijk} sparse, i.e. many tripling coefficients are 0.

4.2 Points or pixels

Perhaps the simplest function representation is tabulation, where values are point-sampled. Mathematically, we can think of the basis functions Ψ as being non-overlapping step functions, with Ψ_i centered about $\boldsymbol{\omega}_i$. The tripling coefficients C_{ijk} are 1 only when $i = j = k$. Evaluating Equation 9 reduces to a three-way dot product of multiplying together the three functions at each of the tabulated points and adding up the result. It is easy to see that this is the simplest possibility, with the most efficient $O(N)$ algorithm.

In spite of this simplicity and efficiency, it is difficult to use tabulated representations, because of the size of these representations. In numerical terms, points are a poor basis for (non-linear) approximation of smooth functions, as we explore in section 6.

4.3 2D Fourier Series

While we are interested in light transport, with integration on a spherical domain, the mathematical framework presented here is independent of domain or application. We therefore now discuss general Fourier series. We begin with the 1D series and, for convenience, define them in complex form ($I = \sqrt{-1}$) on an azimuthal domain $[0, 2\pi]$,

$$\Psi_l(x) = \sqrt{\frac{1}{2\pi}} e^{Ilx}. \tag{11}$$

In complex form, an expression for the tripling coefficients is

$$\begin{aligned} C_{lmn} &= \int \Psi_l \Psi_m \Psi_n^* = (2\pi)^{-3/2} \int_0^{2\pi} e^{Ilx} e^{Imx} e^{-Inx}\, dx \\ &= \sqrt{\frac{1}{2\pi}} \delta_{l+m-n}, \end{aligned} \tag{12}$$

which is nonzero if and only if $n = l + m$. Thus, there is sparsity, with n being determined uniquely by the values of l and m. Intuitively, this shows that there are only $O(N^2)$ nonzero tripling coefficients.

The 2D case follows directly from the fact that the 2D Fourier basis is separable. The number of basis functions and non-zero coefficients are both just the squares of their 1D counterparts, and the computational complexity remains $O(N^2)$. In Appendix B.1 we show that there are exactly

$$\frac{1}{16}(1 + 3N)^2 \tag{13}$$

non-zero tripling coefficients among the N lowest frequency 2D Fourier basis functions. In summary, the computational complexity in the Fourier basis is significantly more expensive than in the point or pixel basis, but more efficient than in the general case.

4.4 Spherical Harmonics

The spherical harmonics in complex form are [MacRobert 1948]

$$Y_{lm}(\theta, \phi) = \alpha_{l|m|} P_m^l(\cos\theta) e^{Im\phi}, \tag{14}$$

where $l \geq 0$, $-l \leq m \leq l$, $\alpha_{l|m|}$ is a normalizing factor, and P_m^l are associated Legendre polynomials. For $l \leq L$, there are a total of $N = (L+1)^2$ harmonics. The tripling coefficients are

$$\begin{aligned} &C_{l'm',l''m'';lm} = \\ &\int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi/2} Y_{l'm'}(\theta,\phi) Y_{l''m''}(\theta,\phi) Y_{lm}^*(\theta,\phi)\, \sin\theta d\theta d\phi. \end{aligned} \tag{15}$$

Note that while we use two indices as is conventional, we could also define single index notation in the standard manner, $Y_p = Y_{lm}$, with $p = l^2 + l + m$ to be consistent with the previous examples. These tripling coefficients are well studied for spherical harmonics, and are known as Clebsch-Gordan coefficients [Inui et al. 1990]. They satisfy two important properties. First, similarly to the Fourier case, the nonzero coefficients satisfy a phase constraint: $m = m' + m''$. Second, to be nonzero they satisfy a triangular inequality on the order, $|l' - l''| \leq l \leq l' + l''$. Hence, there is less sparsity than in the Fourier case, since l is not fixed, and the number of nonzero coefficients are $O(N^2 L)$. Since $N \sim L^2$, $L \sim \sqrt{N}$. Hence, the net computational complexity of triple product integrals is $O(N^{5/2})$. We have derived an expression for the exact number of terms (see Appendix B.2):

$$\frac{9}{20} N^{5/2} + \frac{1}{4} N^{3/2} + \frac{3}{10} N^{1/2} \tag{16}$$

4.5 2D Haar Wavelets

Here we consider the 2D (nonstandard) Haar basis on the unit square. Generalizations to more sophisticated wavelet filters are beyond the scope of this paper (see section 9).

4.5.1 Notation

We define the Haar basis as follows [Stollnitz et al. 1996].

- The scaling basis function is of unit value over the unit square:

$$\Phi(x, y) = 1 \text{ for } (x, y) \in [0, 1]^2 \tag{17}$$

- The wavelet basis functions at level l are scales and dilations of three Haar mother wavelets:

$$\Psi_M^{l\,i\,j}(x, y) = 2^l \Psi_M(2^l x - i, 2^l y - j) \tag{18}$$

where the offsets i and j are integers in the range $[0, 2^l)$ and the type M takes one of three values — 01, 10 or 11, corresponding to one of the three mother wavelets for horizontal, vertical and diagonal differencing:

$\Psi_{01}(x, y) =$  $\Psi_{10}(x, y) =$  $\Psi_{11}(x, y) =$ 

These graphically-defined functions are +1 where white and –1 where black in the unit square shown, and implicitly zero outside that domain.

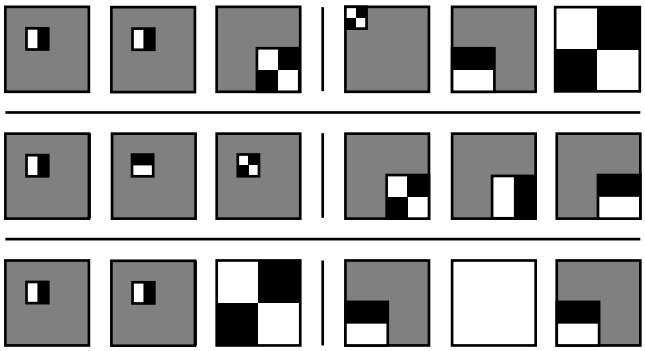


Figure 2: *Examples of triples of 2D Haar basis functions. White is positive, black negative and gray zero. For simplicity we ignore exact magnitude in the diagrams.* Row 1*: Examples having zero triple product integral because not all basis functions overlap.* Row 2*: Examples of triples with non-zero triple product integral by case 2 of the theorem.* Row 3*: Examples of triples with non-zero integral by case 3 of the theorem. In the second example, the middle square is the scaling function.*

Each triplet (l, i, j) defines a **wavelet square** at level l and offset (i, j). Squares at level l have area $1/4^l$, and are disjoint. We use the term *coarser* to describe basis functions in squares at lower levels, and *finer* for those at higher levels. For convenience, the scaling function is assumed to be at the coarsest level 0.

For a square image with N pixels, there are:

- A total of $\log_4 N$ levels: $0, 1, \ldots, (\log_4 N - 1)$. There are 4^l squares at level l.
- A total of $(N-1)\ /\ 3$ wavelet squares, and exactly N wavelet basis elements including the scaling function.

For example, Figure 2 illustrates some of the basis functions for an 8×8 pixel image. There are three levels (0, 1 and 2) with 1, 4 and 16 squares each. In total, there are 21 squares and 64 basis functions.

4.5.2 Haar Tripling Coefficients

The following theorem, which we prove in appendix A, characterizes the tripling coefficients for the basis defined above.

Haar Tripling Coefficient Theorem *The integral of three 2D Haar basis functions, which we label the tripling coefficient C_{uvw} for those three basis elements, is non-zero if and only if one of the following three cases hold:*

1. *All three are the scaling function. In this case, $C_{uvw} = 1$.*
2. *All three occupy the same wavelet square and all are different wavelet types. $C_{uvw} = 2^l$, where the square is at level l.*
3. *Two are identical wavelets, and the third is either the scaling function or a wavelet that overlaps at a strictly coarser level. $C_{uvw} = \pm 2^l$, where the third function exists at level l.*

Figure 2 illustrates examples from cases 2 and 3.

In case 3, the sign of the tripling coefficient is the sign of the coarser basis function in the quadrant that the finer pair of functions fall into.

An implication of the theorem is that most Haar tripling coefficients are zero. The intuition for this phenomenon is the fact that wavelets have compact support, most pairs of basis functions do not overlap, and hence the triple product integrand is the zero function in most cases.

In Appendix B.3 we apply the theorem to show that the exact number of non-zero Haar tripling coefficients for the first N basis functions is

$$2 - N + 3N \log_4 N. \tag{19}$$

Thus, the complexity of evaluating Equation 9 in the Haar basis is bounded by $O(N \log N)$. In section 5, however, we show that the structure of the sum in the Haar basis leads to an algorithm that reduces the complexity of computing the sum to $O(N)$, and even less for *estimating* it accurately.

4.6 Summary and Comparison

The following table collects the computational complexity results for evaluating triple product integrals via Equation 9 in the various bases we have considered. We assume that in a general basis or spherical harmonics, evaluating the integral takes 3 multiplies and 1 add for each non-zero tripling term. In pixels, Fourier series and Haar wavelets, we assume it takes only 2 multiplies and 1 add, because the tripling coefficients are constants.

For comparison, we also present the results for the optimized, linear-time Haar algorithm that we will derive in section 5.2.

Basis	Complexity	Muls and Adds for N
General	$O(N^3)$	$4N^3$
Pixels	$O(N)$	$3N$
Fourier	$O(N^2)$	$\frac{3}{16}(1+3N)^2$
Sph. Harmonics	$O(N^{5/2})$	$\frac{36}{20}N^{5/2} + N^{3/2} + \frac{12}{10}N^{1/2}$
Haar	$O(N \log N)$	$6 - 3N + 9N \log_4 N$
Linear Haar	$O(N)$	$13N$

The main point is that efficient summation using Equation 9 exists only for pixels and wavelets.

5 Algorithm Development

The end product of this section is a simple algorithm for estimating triple product integrals in the Haar basis, where the order of the algorithm is proportional to the number of Haar basis terms used to approximate each of the three factors. We begin with a log-linear algorithm by directly applying the Haar Tripling Theorem (5.1), optimize it to an exact linear algorithm through factorization and dynamic programming (5.2), and derive the final sublinear estimation algorithm by applying non-linear approximation techniques (5.3).

In the following subsections we compute the triple product integral of functions L, V and $\tilde{\rho}$ defined over a unit square domain. To compute the relighting integral of equation 3, we project each spherical function onto a cubemap parameterization and integrate each square face separately (applying the algorithm below 6 times).

5.1 Direct Log-Linear-Time Haar Algorithm

In the following pseudo-code, we assume that `L_scale` is the precomputed scaling function coefficient for the lighting, and `L[s,M]` with `s = (l,i,j)` is the wavelet coefficient for the basis function of type M in square (l, i, j). Similarly, the visibility and BRDF basis coefficients are defined by `V_scale`, `p_scale` and arrays `V` and `p`. The value of the integral is computed in variable `integral`.

We begin by initializing `integral` with the contribution from case 1 of the theorem.

```
integral = p_scale * L_scale * V_scale;
```

We add the six permutation terms corresponding to case 2:

```
for each wavelet square s = (l,i,j)
  C_uvw     = 2^l;
  integral += C_uvw *
   (p[s,01]*L[s,10]*V[s,11] + p[s,01]*L[s,11]*V[s,10]
  + p[s,10]*L[s,11]*V[s,01] + p[s,10]*L[s,01]*V[s,11]
  + p[s,11]*L[s,01]*V[s,10] + p[s,11]*L[s,10]*V[s,01]);
```

Finally we add the contribution from case 3.

```
for each wavelet (s,M)
  integral += ( p[s,M] * L[s,M] * psum(V,s)
              + L[s,M] * V[s,M] * psum(p,s)
              + V[s,M] * p[s,M] * psum(L,s) );
```

This code snippet depends on the following helper function:

```
psum(X, s) for wavelet square s = (l,i,j)
  sum = X_scale;
  for each wavelet (os,oM), with os that overlaps s
    // Compute which quadrant (qx,qy) of square
    // os = (ol,oi,oj) that square s = (l,i,j) lies in
    (qx,qy) = (i,j) / 2^(l-ol-1) - 2 * (oi, oj);
    C_uvw   = sign_of_quadrant(oM,qx,qy) * 2^ol;
    sum += C_uvw * X[os,oM];
  return sum;
```

This function computes what we call the **parent sum** of `X` at the square `(l,i,j)`, which is simply the sum of the coefficients of basis functions overlapping that square, scaled by the appropriate tripling coefficient. The form of this function mirrors the definition of the non-zero terms in case 3 of the theorem.

Here we make the sign of the tripling coefficient explicit. `(qx,qy)` contains which quadrant (`(0,0)`, `(0,1)`, `(1,0)` or `(1,1)`) of the coarser wavelet that the finer wavelet overlaps. The `sign_of_quadrant(oM,qx,qy)` helper function, whose simple definition we omit here, simply returns +1 or –1 for the sign of the `oM` mother wavelet in the given quadrant.

The logarithmic complexity factor of the overall algorithm is implicit in the loop of the `psum` function.

5.2 Linear-Time Haar Algorithm

Now we apply dynamic programming to eliminate this logarithmic factor. The critical observation is that `psum(X,(l,i,j))` can be computed in constant time from the value of the parent square, `psum(X,(l-1,i/2,j/2))`. We make this explicit by redefining `psum` with a recursive algorithm:

```
psum(X, s) where s = (l,i,j)
  if (l,i,j) = (0,0,0) return X_scale;
  os = (ol,oi,oj) = (l-1, i/2, j/2)
  (qx,qy)    = (i,j) - 2 * (oi, oj);
  return psum(X, os) + 2^ol *
    ( X[os,01] * sign_of_quadrant(01,qx,qy)
    + X[os,10] * sign_of_quadrant(10,qx,qy)
    + X[os,11] * sign_of_quadrant(11,qx,qy) );
```

We use this recursive formula and a dynamic programming approach to fill in the table of `psum` values for every wavelet square in linear time. With these modifications, the overall algorithm is $O(N)$.[2] It is worth noting that we cannot perform this dynamic programming approach to optimize the computation of equation 9 for an arbitrary basis. The Haar basis permits this optimization because of the regularity of its tripling coefficients.

5.3 Final Sublinear-Time Haar Algorithm

So far, we have considered only mathematically *exact* calculation of the triple product integral. Now we develop a sublinear wavelet algorithm that computes an *accurate estimate* of the integral much more quickly. Our approach exploits approximate representations of the BRDF and the lighting to accelerate the linear algorithm.

5.3.1 Non-linear Approximation

The central idea is to use what is known in the numerical literature as *non-linear* approximation, where one computes N basis function coefficients, but then chooses only the best $n \ll N$ of them, usually the largest ones. Thus, one does not determine the basis functions to use a-priori, but adapts to the data.

It is possible to carry out non-linear approximation in any basis. However, as we discuss more fully in section 6.1, non-linear approximation in pixels or spherical harmonics does not reliably compress signals.

Ng et al. [2003] demonstrate the efficiency of non-linear *wavelet* approximation for representing lighting. Here we use non-linear wavelet approximations of both the lighting and the BRDF.

5.3.2 Algorithm Optimizations

The modifications are actually relatively minor, so we simply present the full algorithm with changes highlighted:

```
integral = p_scale * L_scale * V_scale;

for each wavelet square s = (l,i,j) in approx of brdf
  C_uvw     = 2^l;
  integral += C_uvw *
   (p[s,01]*L[s,10]*V[s,11] + p[s,01]*L[s,11]*V[s,10]
  + p[s,10]*L[s,11]*V[s,01] + p[s,10]*L[s,01]*V[s,11]
  + p[s,11]*L[s,01]*V[s,10] + p[s,11]*L[s,10]*V[s,01]);

for each wavelet (s,M) in approx of brdf
  integral += ( p[s,M] * L[s,M] * psum_lazy(V,s)
              + p[s,M] * V[s,M] * psum_lazy(L,s) );

for each wavelet (s,M) in approx of lighting
  integral +=   V[s,M] * L[s,M] * psum_lazy(p,s);
```

In short, we reduce the complexity of each loop by iterating just over the chosen non-linear coefficients rather than the full number of basis functions.

An important point to note is that the lighting is actually exact, not approximate, in the first two for loops. This feature helps the approximation converge more quickly.

A second point concerns the asymmetry in the last two loops. The reason for this asymmetry is that we cannot drive efficiency in the last loop through approximation of the BRDF, `p`. This is because the `psum` table for `p` need not be sparse even if `p` itself is sparse. For instance, if `p_scale` were large then `psum(p,s)` would be large for *every* square `s` even if every `p[s,M]` were zero. For this reason, we approximate `L` in the last loop for efficiency.

Returning to the question of symmetry, we could have created a three-way symmetric algorithm by extracting the second line of the second loop into a separate loop over `(s,M) in` **`approx of visibility`**. However, this would be less efficient because we would have to compute and store additional non-linear approximations of the visibility.

A final point concerns the use of the **`psum_lazy`** helper function instead of `psum`. The algorithm would remain $O(N)$ if we were to use dynamic programming to fill in the entire `psum` tables for `p`, `V` and `L`. Instead, `psum_lazy` uses lazy evaluation to fill in the `psum` table only for entries that are actually requested.

The complexity of a single call to this procedure is bounded by $\log N - 1$, but of course N calls can take no more than $O(N)$ time because that would simply reduce to the dynamic programming approach. Hence if n is the number of terms chosen in the

[2] In the table of section 4.6, we report that the linear Haar algorithm uses approximately $13N$ multiplies and adds. Though this depends on the exact implementation, we estimate this number as follows. The algorithm performs for each square, approximately 12 multiplies and 6 adds to accumulate the case 2 terms, and 3 multiplies and 3 adds to calculate the parent sum value. Furthermore, there are 18 multiplies and 12 adds to accumulate the case 3 terms at half the squares (there are no case 3 terms at the finest level). This totals $8N$ multiplies and $5N$ adds.

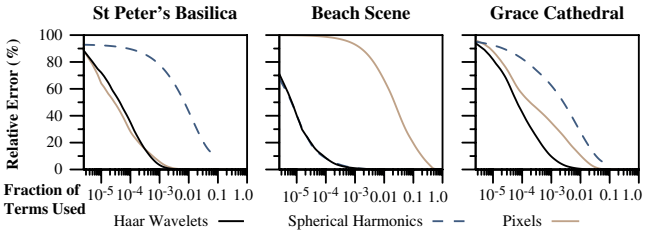
non-linear approximation of the lighting and BRDF, then the overall algorithm complexity is bounded above by $O(n \log N)$, and converges quickly to $O(n)$.

6 Numerical Comparison and Validation

A purely mathematical analysis in section 4 showed that frequency-based Fourier and spherical harmonic methods are intractable because their tripling series complexity is quadratic or worse. Here we show numerically that pixel-based methods are also intractable because of poor compression performance. We also show that the Haar integration algorithm of 5.3 is numerically efficient.

6.1 Compression Performance

The graphs below report accuracy in non-linear approximation of measured lighting [Debevec and Malik 1997]. Our main focus is comparing pixels and wavelets; harmonic data are provided for didactic purposes. We compute the optimal non-linear approximation by sorting the basis coefficients by magnitude and retaining the largest. The graphs plot the residual error as we increase the number of terms retained. Note that the horizontal axis is plotted on a log scale.



From a signal-processing standpoint, pixels localize only spatial information and require many terms to represent low-frequency functions, such as the bright sky in the Beach Scene. Conversely, spherical harmonics localize only frequency information and require many terms for small bright lights, such as the small windows in the St Peter's Basilica environment. In contrast, wavelets contain basis functions that are localized in both space and frequency domains, enabling more reliable compression. Wavelets perform as well as either pixels or harmonics for these two scenes (note that the plots for wavelets and harmonics overlap in the middle graph).

As a composite example, Grace Cathedral contains broad area illumination as well as concentrated spotlights. Both pixels and harmonics perform poorly compared to wavelets, which converge at least an order of magnitude faster for reasonable accuracy.

We have focused here on lighting environments, because pixels are likely to perform best in this space of functions where the energy can be extremely concentrated. Pixels will in general perform worse for visibility and BRDFs, where the functions are likely to contain broad areas of significant energy.

This observation leads to the important conclusion that pixel based methods are intractable for our application, because the raw pre-computed visibility and BRDF datasets (see 7.1) are too large to fit in memory. Hence, pixel relighting would require compressed visibility and BRDF representations, and would incur the cost of decompression in the inner loop.

Even if pre-computed dataset size were not an issue, the graphs show that essentially all pixel terms are required for accuracy in common datasets. A full pixel integration is slower than accurate wavelet integration, as shown in the next section.

6.2 Numerical Performance of Haar Integration

This section measures the performance of the sublinear Haar estimation algorithm. We directly implement the algorithm described

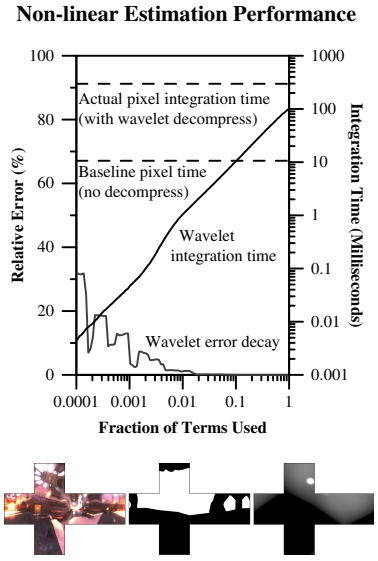


Figure 3: Top*: Error decay and running time for triple product integral estimation algorithm (section 5) for one representative set of cubemaps. Note that the running time has unit slope on the log-log plot, indicating that the algorithm is linear in the number of coefficients used. Horizontal dashed lines are running times for linear pixel estimation, with and without decompression cost. Note that the wavelet algorithm produces accurate results orders of magnitude faster than simple pixel integration.* Bottom*: The three* $6 \times 256 \times 256$ *cubemaps which were multiplied and integrated in the experiment.*

in section 5 to integrate one set of lighting, visibility and BRDF cubemaps. In Figure 3, we plot on a log-log graph the running time of our algorithm as a function of the number of approximation terms used, n. The first feature to note is that the graph converges to a line with slope 1, reflecting the fact that for practical use the algorithm is $O(n)$ as discussed at the end of section 5.3.2.

For comparison, we also plot in Figure 3 the time to compute a full pixel integration, both with and without decompression cost. In this experiment, decompression corresponds to an inverse transform from a compressed wavelet representation. As seen in the figure, for reasonable error, wavelet integration is 10–30 times faster than simple pixel integration. Note that it is not possible to include a meaningful plot of *non-linear* pixel performance, because results vary so widely with input lighting, and because there is the unknown cost of reading from compressed visibility and BRDF representations.

In summary, this section shows that wavelets perform reliable and efficient compression, and that wavelets allow efficient relighting directly from the compressed representation.

7 Implementation

At a high level we relight a scene vertex by vertex by using the algorithm in section 5.3. The major issue, addressed in 7.1, is how we represent and pre-compute the visibility and BRDF. 7.2 deals with final rendering given these pre-computed datastructures.

7.1 Representation and Pre-computation

7.1.1 Visibility Field

We pre-compute what we refer to as the 4D **visibility field** $V(\mathbf{x}, \boldsymbol{\omega})$, as a function of the surface location $\mathbf{x}$ and the global incident direction $\boldsymbol{\omega}$. The visibility is computed at each vertex, $\mathbf{x}$, on a model and each incident direction on a cubemap (up to $6 \times 256 \times 256$).

For a given, coarsely tessellated input model, we refine each of its triangles so that we have vertices densely over all surfaces. We then pre-compute $V(\mathbf{x}, \boldsymbol{\omega})$ using hardware rasterization of hemicubes at each vertex. As an important optimization, we rasterize the original, coarsely tessellated model. Note that this produces exactly the same results as rasterizing the refined model.

Other than this optimization, our pre-computation is quite similar to previous methods like Ng et al. [2003]. For the cubemap

at each vertex, we Haar transform each face, quantize to 8 bits, and store only the non-zero coefficients. Because visibility is a binary function, our visibility cubemaps are more amenable to Haar wavelet transforms than the full transport operator used by Ng et al. As a result, we obtain somewhat higher compression and sparsity.

We store the coefficients for each visiblity function in a hashtable, so that the datastructure is compact and yet provides fast indexed lookup.

7.1.2 Material Field (BRDF)

We also pre-compute what we refer to as the **material field**, corresponding to a particular BRDF. Correctly representing the BRDF in global coordinates is more complicated than in the case of the visibility. In principle, one could represent the 4D BRDF (multiplied by the incident cosine) $\rho(\boldsymbol{\omega}, \boldsymbol{\omega_o})$ in wavelets. For each vertex, one could then find $\boldsymbol{\omega_o}$, to determine $\rho(\boldsymbol{\omega})$. However, for uniformity with the lighting and visibility, this must be in global coordinates, while the BRDF is usually stored in local coordinates with respect to the local surface normal. Hence, we would need to apply a rotation corresponding to the surface normal to obtain $\tilde{\rho}(\boldsymbol{\omega})$ that we need—not a trivial operation in wavelets.

We address this challenge as follows. First, we write the BRDF as a 6D function, $\tilde{\rho}(\boldsymbol{\omega}, \boldsymbol{\omega_o}, \mathbf{n})$, where we explicitly note that it depends on $\mathbf{n}$, the global coordinates of the surface normal, and use $\tilde{\rho}$ to emphasize that all values are now in global coordinates. Of course, the intrinsic BRDF of the surface is only 4D, depending only on the relative orientation of $\boldsymbol{\omega}$ and $\boldsymbol{\omega_o}$ with respect to $\mathbf{n}$. However, it is convenient to trade storage for computation (avoiding the required rotation) by explicitly representing the 6D function $\tilde{\rho}(\boldsymbol{\omega}, \boldsymbol{\omega_o}, \mathbf{n})$.

The problem is that the dataset grows too large because we must tabulate 4D functions for a sampling of $\mathbf{n}$ directions. Part of the problem is that parameterization by $\boldsymbol{\omega}$ and $\boldsymbol{\omega_o}$ is not very compact for most BRDFs, as noted by Rusinkiewicz [1998].

Reparameterization As such, we use a reparameterization that makes most BRDFs low-frequency in the $\mathbf{n}$ parameter. Specifically, we replace the view parameter, $\boldsymbol{\omega_o}$, with the reflection vector, which we denote $\boldsymbol{\omega_r}$. The reflection vector is the reflection of $\boldsymbol{\omega_o}$ about $\mathbf{n}$. $\tilde{\rho}$ is compact in the new parameterization. In particular, Ramamoorthi and Hanrahan [2002], and earlier Cabral et al. [1999], show that a sparse sampling of normal directions (in their case actually view directions) is accurate for essentially all isotropic BRDFs, including measured ones. This is because the $\boldsymbol{\omega_r}$ sampling tracks the most common high-frequency effects, which are specular-like lobes centered near $\boldsymbol{\omega_r}$. We use resolutions in $\mathbf{n} \times \boldsymbol{\omega_r} \times \boldsymbol{\omega}$ of up to $(6 \times 3 \times 3) \times (6 \times 32 \times 32) \times (6 \times 64 \times 64)$.

For certain special cases like the Lambertian or Phong models, the reparameterization reduces the 6D datasets back to 4D. Lambertian shading is given by $(\boldsymbol{\omega} \cdot \mathbf{n})$, which is independent of $\boldsymbol{\omega_r}$, and in Phong shading, $\tilde{\rho}(\boldsymbol{\omega}, \boldsymbol{\omega_r}, \mathbf{n})$ is of the form $(\boldsymbol{\omega} \cdot \boldsymbol{\omega_r})^s$, which is independent of $\mathbf{n}$. We treat these standard shading models as special cases, handling much higher resolutions than general isotropic BRDFs. For Lambertian BRDF, we use a resolution in $\mathbf{n} \times \boldsymbol{\omega}$ of $(6 \times 8 \times 8) \times (6 \times 64 \times 64)$. For Phong, we use resolutions in $\boldsymbol{\omega_r} \times \boldsymbol{\omega}$ of up to $(6 \times 64 \times 64) \times (6 \times 128 \times 128)$ to handle specular exponents up to 200.

Non-linear Approximation To reduce the storage size further, for all sampled $\boldsymbol{\omega_r}$ and $\mathbf{n}$, we represent $\tilde{\rho}(\boldsymbol{\omega})$ in wavelets, using a non-linear approximation with a small number of wavelet terms. For 99% accuracy, we find that we require only 0.1% - 1% of the full number of terms. Once we use non-linear approximation, the number of terms needed, while large, is a fraction of the storage required for the visibility field.

7.2 Relighting

The following steps occur at every frame to relight under changing lighting, view and possibly visibility and material fields.

Input Lighting Analysis The input to relighting is a distant illumination environment cubemap (at a resolution of up to $6 \times 256 \times 256$). We transform the cubemap into Haar wavelets. We also compute a non-linear approximation of the Haar coefficient set, choosing the n wavelet squares with the largest coefficients. This step is almost identical to that in Ng et al. [2003].

Computing Relit Mesh Colors We relight only the set of vertices that contribute to final screen pixels. We calculate this set by rendering all triangles with different colors, and marking the corresponding vertices for all colors that appear as screen pixels. This optimization reduces work by 60–70% for typical views of our scenes.

For each vertex in the set, we perform the following steps to compute its final color:

- We look up $V(\boldsymbol{\omega})$ for the current vertex directly in the pre-computed visibility field.
- We compute $\tilde{\rho}(\boldsymbol{\omega})$ by looking up the vertex's surface normal, $\mathbf{n}$, and computing the reflected direction, $\boldsymbol{\omega_r}$ given the eye point. We use quadrilinear interpolation between the 16 nearest sampled $(\mathbf{n}, \boldsymbol{\omega_r})$ pairs in the pre-computed material field, to ensure that our reconstruction of $\tilde{\rho}(\boldsymbol{\omega}, \boldsymbol{\omega_r}, \mathbf{n})$ is continuous and to avoid visual blocking artifacts over the surface.
- We then directly implement the algorithm in section 5 to compute the three-term integral for the current vertex, corresponding to Equation 3.

Final Rendering We use graphics hardware in the standard way to rasterize the object and interpolate vertex colors. We can also modulate by texture maps, as shown in Figures 1 and 5.

8 Practical Results

In this section, we present some results from our system for all-frequency relighting with changing view. We do not compare with previous pre-computed transport algorithms in this section, since they are impractical to implement at these resolutions for changing both lighting and viewpoint, as already discussed in the introduction and section 6.

8.1 Accuracy

We first verify the accuracy and visual quality of the images produces by our Haar relighting algorithm (see Figure 4), when we vary the numbers of terms used in non-linear approximation of the lighting and BRDF. Artifact-free images are produced with as few as 0.01% of the terms, because of the continuous reconstruction of our BRDF representation as described in section 7. Images that are very close to a reference (created with exact integration) require just 0.1% – 1% of the wavelet terms.

8.2 Statistics and Performance

The following tables summarize the pre-computation statistics for our scenes. The chair and table scenes were modelled by Jeff Mancuso. The orange and velvet BRDFs are converted from the CURET database [Dana et al. 1999]. The resolutions of directional parameters ($\boldsymbol{\omega}$, $\mathbf{n}$ and $\boldsymbol{\omega_r}$) are given as the size of a single face of a cubemap. In the case of visibility, sparsity is the fraction of terms retained for exact representation of each cubemap, subject to quantization error. In the case of BRDFs, sparsity refers to the average fraction retained for at least 95% accuracy (99% accuracy for Lambertian and Phong).

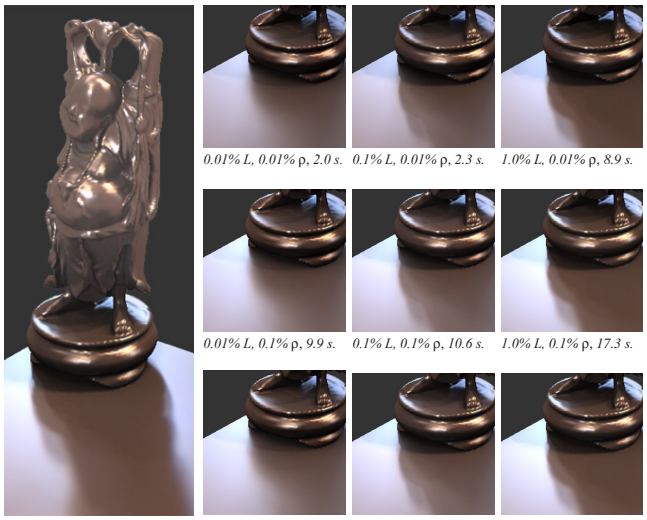


Figure 4: *Comparison of accuracy with varying numbers of lighting and BRDF terms increasing right and down. Numbers below each image are the fraction of the total number number of lighting and BRDF wavelets used to generate each image, and the running time (in seconds). For this comparison we use a relatively high cubemap resolution ($6\times 256\times 256$), which gives slightly longer render times. 1% coefficients or less in each is indistinguishable from the reference, but even fewer are perceptually acceptable. In practice, we use between 0.1% and 1%.*

Visibility	Fine	Coarse	ω res	Size	Sparsity	Pr. Time
Single chair	143K	4K	64^2	816 MB	5.4%	21 min
Single chair	143K	4K	128^2	1.4 GB	3.1%	45 min
Table scene	300K	23K	64^2	1.5 GB	6.2%	1 hr
Buddha	93K	93K	64^2	250 MB	3.4%	1 hr
Buddha	93K	93K	256^2	1 GB	0.8%	4 hrs
Material	**n res**	**ω_r res**	**ω res**	**Size**	**Sparsity**	**Pr. Time**
Lambertian	8^2	—	64^2	250 KB	0.12%	25 sec
Phong (64)	—	8^2	128^2	440 KB	0.05%	3 min
Phong (200)	—	64^2	128^2	19 MB	0.50%	18 min
Orange	3^2	8^2	64^2	16 MB	0.60%	1 hr
Orange	3^2	32^2	64^2	250 MB	0.60%	16 hrs
Velvet	3^2	8^2	64^2	18 MB	0.67%	1 hrs
Velvet	3^2	32^2	64^2	290 MB	0.67%	16 hrs

Note that our total pre-computation times and storage requirements, while high, are essentially the same as those of Ng et al. [2003]. In fact our visibility pre-computation is significantly faster than theirs because we render coarse visibility meshes, as described in 7.1.1. Our pre-computation times are also favorable relative to those of Sloan et al. [2003a], when their method is scaled to our resolutions. The pre-computation for the buddha is slowest because the original mesh, which is used for the visibility rasterization, already exists at the maximum resolution.

The following table presents representative rendering speeds with BRDF and lighting qualities that produce accurate images. Both scenes have an angular resolution of $6\times 64\times 64$.

Scene	BRDF (ρ)	Light (L)	Sp. (ρ)	Sp. (L)	Render Time
Table sc.	Multiple	Kitchen	0.1%	0.5%	3-5 s / fr.
Buddha	Phong (64)	Grace	0.1%	0.5%	3 s / fr.

The lighting environments are high dynamic range cubemaps from Paul Debevec's Light Probe gallery [Debevec and Malik 1997]. The table scene contains Lambertian, Phong and orange BRDFs.

Even for extremely high resolutions, our method produces relit images in a few seconds. While this is not real-time yet, it is two to three orders of magnitude shorter than the design cycle currently available in applications like lighting and material design, and faster than any competing technique.

Figure 5: *More images of the scene in Figure 1, at 3–5 seconds per frame. These closely match reference images, which are omitted.* Left*: Two images where we change only the viewpoint (lighting is static). We omit texture to enhance shadow subtleties. As one of the characteristic features captured by our technique, note that the glossy shadows in front of the right chair change dramatically with viewpoint, but that the diffuse shadows on the table and behind the left chair remain stationary on the surfaces.* Right*: Two pairs of images where only the lighting changes.*

8.3 Images

Figure 5 shows some additional images produced with our technique. Once we loaded the pre-computed data, we could create relit images in one to ten seconds, typically with about 0.5% of the lighting terms and 0.1% of the BRDF terms. The speed can be controlled approximately linearly by interactively increasing and decreasing the number of BRDF and lighting terms used.

We were able to continuously manipulate lighting and viewpoint without restriction. We could also change materials assigned to different parts of the scene. Our examples focus on detailed soft shadows and view-dependent glossiness that emerge at the high resolutions we consider.

9 Research Directions

Haar Filter Our analysis thus far has been limited to the simplest wavelet filter, Haar. The main reason that we have focused on Haar is that its tripling coefficients have a very simple analytic form and permit optimization through dynamic programming. General wavelets do not have analytically simple tripling coefficients, nor do they obviously permit the optimizations we use. It is worth noting that even though Haar approximation suffers from block artifacts in image compression applications, this is not a direct problem here. We never see the approximate functions directly, and artifacts largely disappear through integration in the relighting operator.

Nevertheless, smoother wavelets are crucial in other domains. One potentially important application is multiplication of images in the compressed domain, without decompression, via Equation 10. Such an application would require an understanding of tripling coefficients for the higher order wavelets used in image compression.

Material Representation Our general BRDF representation limits how much resolution we can afford in the reflection direction, for the simple reason that we also tabulate normal dependence. As a related limitation, since we tabulate rotations only for different normal directions (and not local tangent directions), we cannot handle anisotropic BRDFs. This suggests that it would be well worth looking directly for efficient algorithms to rotate wavelet representations of spherical functions. This would allow us to store the BRDF more compactly and precisely in local surface coordinates (where it assumes its intrinsic 4D form) and compute its rotation into the global frame at each vertex.

10 Conclusions

We have explored pre-computed light transport methods for changing illumination and viewpoint, while including all-frequency shadows, reflections and lighting. We show that current techniques do not scale effectively to the high sampling rates and dimensionality of the problem, so we need to relight from factored representations of visibility and materials.

This factored approach motivates our primary contribution: introducing triple product integrals (here, involving the lighting, visibility and the BRDF) as a mathematical object of study. We analyze the complexity of these integrals in a number of bases, showing why wavelets are a computationally attractive alternative, and developing efficient sublinear-time algorithms for Haar wavelets. We demonstrate the practical utility of our framework, reducing the iteration cycle for applications like lighting design by orders of magnitude, to near interactive rates.

Our analysis suggests broader implications for computer graphics. First, the idea of using factored representations of the full 6D transport operator might be more generally applied to image-based rendering, where a huge number of images are required to represent high-frequency effects with both lighting and view variation. If one could separately measure the 4D visibility and BRDF, those could be used for factored IBR, requiring much sparser datasets. Besides the practical utility, we also believe factorization to be an important theoretical idea that is worth investigating futher.

Second, the systematic study of triple product integrals and their computational complexity is likely to have wider uses in both computer graphics and applied mathematics. We believe this to be a basic operation that is important to perform efficiently, but which has not previously received attention. The same mathematical machinery provides a sound way to analyze another basic operation—multiplying two functions represented with sparse basis coefficient sets. We predict much future work and applications.

Finally, our practical application is to fast rendering of high quality images with arbitrary lighting, viewpoint and materials. We believe the use of realistic lighting, shadows, and materials to be of increasing importance in both real-time applications and in potentially interactive applications like product design. We see our work as a step in allowing designers to interact with realistic images at truly interactive rates, simultaneously manipulating the illumination, viewpoint and material properties.

Acknowledgments

We would especially like to thank Jeff Mancuso for modeling the furniture scene, and Thos. Moser for allowing us to copy the likeness of their Continuous Arm Chair. We would also like to thank Jeff Klingner, Pradeep Sen, and Dave Akers for their help in developing software and editing the paper and video. Thanks to David Donoho and Doron Levy for discussions. This work was supported by grants from the NSF (0085864-2 on *Interacting with the Visual World*, and 0305322 on *Real-Time Visualization and Rendering of Complex Scenes*) and Intel Corporation (*Real-Time Interaction and Rendering with Complex Illumination and Materials*).

References

AGARWAL, S., RAMAMOORTHI, R., BELONGIE, S., AND JENSEN, H. 2003. Structured importance sampling of environment maps. *ACM Transactions on Graphics 22*, 3, 605–612.

ARVO, J., TORRANCE, K., AND SMITS, B. 1994. A framework for the analysis of error in global illumination algorithms. In *SIGGRAPH 94*, 75–84.

CABRAL, B., OLANO, M., AND NEMEC, P. 1999. Reflection space image based rendering. In *SIGGRAPH 99*, 165–170.

DANA, K., GINNEKEN, B., NAYAR, S., AND KOENDERINK, J. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (January), 1–34.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97*, 369–378.

DEVORE, R. 1998. Nonlinear approximation. *Acta Numerica 7*, 51–150.

DORSEY, J., ARVO, J., AND GREENBERG, D. 1995. Interactive design of complex time dependent lighting. *IEEE Computer Graphics and Applications 15*, 2 (March), 26–36.

GERSHBEIN, R., SCHRÖDER, P., AND HANRAHAN, P. 1994. Textures and radiosity: Controlling emission and reflection with texture maps. In *SIGGRAPH 94*, 51–58.

INUI, T., TANABE, Y., AND ONODERA, Y. 1990. *Group theory and its applications in physics*. Springer Verlag.

KAUTZ, J., AND MCCOOL, M. 1999. Interactive rendering with arbitrary BRDFs using separable approximations. In *Eurographics Rendering Workshop 99*, 247–260.

KAUTZ, J., SNYDER, J., AND SLOAN, P. 2002. Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *Eurographics Rendering Workshop 2002*, 291–296.

LEHTINEN, J., AND KAUTZ, J. 2003. Matrix radiance transfer. In *Symposium on Interactive 3D graphics*, 59–64.

MACROBERT, T. 1948. *Spherical harmonics; an elementary treatise on harmonic functions, with applications*. Dover Publications.

MALLAT, S. 1999. *A Wavelet Tour of Signal Processing*. Academic Press.

MEYER, Y., COIFMAN, R., AND SALINGER, D. 1997. *Wavelets: Calderon-Zygmund and Multilinear Operators*. Cambridge Univ. Press.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics 22*, 3, 376–381.

NIMEROFF, J., SIMONCELLI, E., AND DORSEY, J. 1994. Efficient re-rendering of naturally illuminated environments. In *Eurographics Workshop on Rendering 94*, 359–373.

RAMAMOORTHI, R., AND HANRAHAN, P. 2002. Frequency space environment map rendering. *ACM Transactions on Graphics (SIGGRAPH 02 proceedings) 21*, 3, 517–526.

RUSINKIEWICZ, S. 1998. A new change of variables for efficient BRDF representation. In *Eurographics Rendering Workshop 98*, 11–22.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3, 527–536.

SLOAN, P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics 22*, 3, 382–391.

SLOAN, P., LIU, X., SHUM, H., AND SNYDER, J. 2003. Bi-scale radiance transfer. *ACM Transactions on Graphics 22*, 3, 370–375.

STOLLNITZ, E., DEROSE, T., AND SALESIN, D. 1996. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann.

THORNBER, K., AND JACOBS, D. 2001. Broadened, specular reflection and linear subspaces. Tech. Rep. TR#2001-033, NEC.

Appendices

A Proof of Haar Tripling Coefficient Theorem

In this section we prove the theorem stated in section 4.5.2 that characterizes the Haar tripling coefficients. The proof is straightforward though somewhat long, so we adopt a terse style of proof.

Preliminaries We begin with the following observations, which are either definitions or easily verified and stated without proof:

1. The wavelets are orthonormal: $\iint \Psi_M^{l\,i\,j} \Psi_{M'}^{l'\,i'\,j'}\, dx\, dy = \delta_{M-M'}\delta_{l-l'}\delta_{i-i'}\delta_{j-j'}$
2. The wavelets have vanishing integrals: $\iint \Psi_M^{l\,i\,j} dx\, dy = 0$
3. The product of the scaling function and any wavelet is simply the wavelet: $\Phi \cdot \Psi_M^{l\,i\,j} = \Psi_M^{l\,i\,j}$.
4. The product of two wavelets is zero if they do not overlap.
5. The product of two different wavelets in the same wavelet square (l, i, j), is the third wavelet in that square scaled by 2^l. For example, $\Psi_{01}^{l\,i\,j} \cdot \Psi_{11}^{l\,i\,j} = 2^l \Psi_{10}^{l\,i\,j}$.
6. The square of a wavelet in (l, i, j) is equal to the scaling function restricted to (l, i, j) and scaled by 4^l.
7. The product of two overlapping wavelets at different levels $l < l'$ is the finer wavelet scaled by $\pm 2^l$. The sign is made precise in section 5.1.

We now proceed to prove the theorem through a case analysis on the choice of the basis functions in the tripling coefficient integral C_{uvw} as defined in equation 8.

A.1 All three basis functions are wavelets

For cases 1 through 3 we assume that all three functions in the integral are wavelets (not the scaling function).

Case 1: All wavelets at same level Since all Haar squares at the same level are disjoint, observation 4 shows that all three basis functions must be in the same square if C_{uvw} is non-zero.

If at least two are the same, then observation 6 shows that the integrand of C_{uvw} is the third wavelet scaled by 4^l, where l is the level. In this case, observation 2 shows that the integral is zero.

As a result, if C_{uvw} is non-zero then all three basis functions must be different. Applying observations 5 and 1, we calculate the value of C_{uvw} as 2^l. This establishes case 2 of the theorem.

Case 2: Exactly two wavelets at same level As in case 1, observation 4 shows that the two wavelets must occupy the same square if C_{uvw} is non-zero.

If the third wavelet is at a finer level, then two applications of observation 7 shows that the integrand of C_{uvw} is a scaling of the finer wavelet. Observation 2 concludes that C_{uvw} is zero.

Thus, the third wavelet must be at a coarser overlapping level if C_{uvw} is non-zero. In this case, observation 7 and orthonormality (observation 1) show that the two same-square wavelets must be identical, and the integral is $\pm 2^l$. This establishes part of case 3 in the statement of the theorem.

Case 3: All three wavelets at different levels Two applications of observation 7 shows that the integrand of C_{uvw} is a scaling of the finest level wavelet. This integral is zero by observation 2.

A.2 Some basis functions are the scaling function

Now we consider the cases where at least one of the factors in the integrand of C_{uvw} is the Haar scaling function. If exactly one is the scaling function, then two are wavelets. Thus, the scaling property (observation 3) and orthonormality (observation 1) show that the integral has value 1 if the wavelets are identical, and is zero otherwise. This completes the proof of case 3 of the theorem statement.

If exactly two of the basis functions are the scaling function, then observations 3 and observation 2 show that $C_{uvw} = 0$.

Finally, if all three are the scaling function we have case 1 of the theorem statement. A trivial computation shows that $C_{uvw} = 1$. This concludes the case analysis and the proof of the theorem.

B Details of Computational Complexity Analysis

This section provides details on calculating the number of non-zero tripling coefficients in section 4.

B.1 Fourier Series

First for 1D, assume that Ψ_l is bandlimited with $|l| \leq M$. In summary from section 4.3, our constraints for C_{lmn} to be non-zero are: $n = l + m$ and $|l, m, n| \leq M$. Thus, the total number of non-zero coefficients is given by $\sum_{l=-M}^{M} \sum_{m=-M}^{M} \sum_{n=-M}^{M} \delta_{l+m-n}$. It is easy to verify that this summation is equal to $1 + 3M + 3M^2$, and we omit these details.

This completes the 1D analysis. Since the 2D Fourier basis is separable, the number of non-zero coefficients is simply the square of the 1D case. Precisely, for bandlimited 2D Fourier series, $\Psi_{lk}(x, y) = 1/2\pi\, e^{Ilx} e^{Iky}$ where $|l, k| \leq M$, the total number of non-zero tripling coefficients is $(1 + 3M + 3M^2)^2$. This formula is specified in terms of the bandlimit, M, however, rather than the number of basis functions N. In the 2D case, $N = (2M + 1)^2$, and so simple substitution and algebra show that the total number of non-zero terms is $1/16\,(1 + 3N)^2$, as stated in section 4.3.

B.2 Spherical Harmonics

We have derived a polynomial for the number of non-zero harmonic tripling coefficients as a function of the order. This polynomial is a fifth degree expression that can be verified using a symbolic mathematical program like Mathematica to predict the number of nonzero Clebsch-Gordan terms. The exact expression is $9/20L^5 + 9/4L^4 + 19/4L^3 + 21/4L^2 + 33/10L + 1$, where L is the maximum harmonic order. Since the number of basis functions is $N = (L + 1)^2$, the number of non-zero coefficients relative to the basis size is $9/20\, N^{5/2} + 1/4\, N^{3/2} + 3/10\, N^{1/2}$.

B.3 Haar Wavelets

We calculate the exact number of non-zero coefficients, by counting the contributions from each of the three cases of the Haar Tripling Coefficient Theorem (section 4.5.2). Case 1 contributes exactly one non-zero coefficient.

Case 2 contributes a linear number of terms because the only overlaps of Haar basis functions at the same level are the three mother wavelets occupying the same square. Precisely, case 2 contributes $3!\,(N-1)\,/\,3 = 2(N-1)$ coefficients, where the 3! term is due to permutation choices in assigning the three types to the basis functions, and $(N-1)\,/\,3$ is the total number of wavelet squares.

Case 3 contributes, in total, a log-linear number of terms from the basis functions that overlap at a coarser level because each square is covered by a log number of squares at coarser levels. To be precise, case 3 contributes exactly $\sum_{l=0}^{L} N_S[l] \cdot N_M \cdot N_O[l]$ non-zero tripling coefficients; in this equation $L = (\log_4 N - 1)$ is the finest wavelet level, $N_S[l] = 4^l$ is the total number of wavelet squares at level l, $N_M = 3$ is the number of wavelets in each square, and $N_O[l] = (3l + 1)$ is the number of basis functions overlapping each square at level l (3 for each overlapping square plus the scaling function). With these substitutions, it can be verified that the summation evaluates to $3\,(1 - N + N \log_4 N)$ non-zero coefficients.

Adding the contributions from the three cases of the theorem, the number of non-zero Haar tripling coefficients for the first N basis functions is $2 - N + 3N \log_4 N$, as stated at the end of section 4.5.2.