EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

*Volume 32 (2013), Number 2*

# Locally Adaptive Products for All-Frequency Relighting

Yaron Inger  Zeev Farbman  Dani Lischinski
The Hebrew University of Jerusalem, Israel

**Abstract**

*Triple product integrals evaluate the shading at a point by factoring the reflection equation into incident illumination, visibility, and BRDF. By densely sampling the space of incident directions, this approach is capable of highly accurate rendering scenes lit by high-frequency environment lighting, containing complex materials and featuring intricate shadows. Efficient evaluation of triple product integrals using Haar wavelets enables near-interactive rendering of such scenes, while dynamically changing the lighting and the view. Although faster methods have been proposed in the recent real-time rendering literature, the approximations employed in these methods typically limit them to lower frequency phenomena.*

*In this paper, we present a new approach for high-frequency scene relighting within the triple product framework. Our approach breaks the computation to smaller solid angles (blocks) over most of which the triple product degenerates to a dot product. We introduce a lossless, yet compact, differential representation of the visibility function over each block, and sample the BRDF on the fly, eliminating the need to store multiple rotated copies of each BRDF. By combining these ideas, we are able to achieve true interactive performance even when running on a CPU, while supporting high frequency effects in scenes with high vertex counts.*

## 1. Introduction

High fidelity photorealistic rendering with dynamically changing lighting, viewpoint, materials, and scene geometry is still a major computational challenge. Offline physically-based renderers are able to produce images of stunning realism, but at rates that are still far from interactive, while real-time renderers powered by today's programmable graphics hardware are still unable to produce many complex shading effects. Precomputation-based rendering methods [Ram09] attempt to bridge this gap and enable fast rendering with complex illumination, materials and shadows at the price of fixing some aspects, such as the scene geometry.

An overview of precomputation-based rendering methods is provided in the next section. One major class of such methods originates from the precomputed radiance transfer (PRT) method of Sloan *et al.* [SKS02]. Methods in this class have enjoyed wide practical adoption, since they achieve real-time performance, and are able to simulate subtle inter-reflection effects. However, the shadows and reflections that they are able to produce are, in practice, limited to relatively low frequencies [MTR08].

We focus on a slower, but more accurate precomputation-based rendering approach, which is based on the concept of *triple-product integrals* [NRH04]. The idea is to approximate the direct illumination at each scene vertex by integrating the product of three functions: distant illumination, precomputed visibility of the environment, and the BRDF. Because of its dense sampling of the space of incident di-

**Figure 1:** *Bas-relief: a scene with 506K vertices rendered interactively by our system, with dynamic environment lighting, changing viewpoint, and dynamic highly glossy materials.*

rections, this approach is able to accurately handle much higher frequency illumination, shadows, and BRDFs than other PRT approaches. However, these advantages come at the price of significant storage requirements and much higher rendering time. Furthermore, efficient evaluation of triple product integrals involves a highly nontrivial dynamic programming computation at each vertex making the algorithm ill-suited for fine grain parallelism.

In this work we present a novel, simple, yet efficient, approach to all-frequency relighting within the triple-product integrals framework. The proposed approach is faster and requires considerably less storage than the original approach of Ng *et al.* [NRH04], achieving fully interactive frame rates, even on a CPU. At the same time, it enables faithfully reproducing even higher frequency shadows and specularities, and supports a wider range of materials.

The core ideas that make our approach possible are:

1. Evaluating the integral independently over small solid angles (blocks), instead of over the entire sphere (or entire cubemap faces). The three integrand functions are generally much simpler over a small solid angle, and the computation reduces to a simple dot product in most cases.
2. A compact differential representation of the visibility map at each vertex, which supports either lossless binary visibility maps, or non-binary visibility for anti-aliased shadows. A one-dimensional vector represents the visibility in each block, which is quickly expanded to the standard basis when needed.
3. The BRDF coefficients needed to perform the product over each block are quickly generated on the fly by sampling the BRDF. Thus, it is not necessary to precompute and store a large number of pre-rotated copies of each BRDF, making it possible to cope with higher-frequency and anisotropic reflectances.

Similarly to the original approach of Ng *et al.*, our method does not simulate global illumination effects, and is limited to static scene geometry, since it still requires precomputing and storing a visibility map at each scene vertex. However, we hope to convince the reader that, within the regime of all-frequency relighting of static scenes, our method produces more accurate results and copes with higher frequencies than other existing real-time methods.

## 2. Previous work

The emergence of powerful programmable graphics processors in the last decade has enabled many sophisticated rendering algorithms to enter the realm of interactive rendering. However, accurate relighting of scenes with realistic environment lighting, general BRDFs, and intricate all-frequency shadows, still typically requires some precomputation to support interactive rates. A comprehensive review of such techniques is beyond the scope of this paper, and the reader is referred to the excellent surveys by Lehtinen [Leh07] and Ramamoorthi [Ram09]. In this section we only briefly outline the main categories of such methods (heavily drawing on [Ram09]), and then focus on all-frequency relighting with triple product integrals.

In their seminal work, Sloan *et al.* [SKS02] coined the term *Precomputed Radiance Transfer* (PRT), sparking the interest in precomputation-based methods. This work made use of the spherical harmonics basis to compactly represent distant illumination given as a full environment map, as well as the precomputed transfer function at every scene vertex capturing shadowing and interreflection effects. The shading computation at each vertex thus amounts to a dot product, enabling real-time relighting, but limited to low-frequency illumination and shadows. Various extensions and improvements followed (e.g., [KSS02, SHHS03, LSSS04, WTL04, TS06]), however the fundamental low-frequency limitation remains [MTR08].

In another pioneering work, Ng *et al.* [NRH03] presented a method for all-frequency relighting that was limited to images with fixed view, or to diffuse geometry. This approach was then extended to enable all-frequency relighting of glossy objects with the ability to dynamically change the view as well [NRH04], introducing the concept of *triple product integrals*, which will be explained in more detail in Section 3. The shading at each vertex is computed by integrating the product of three functions: the lighting (environment map), the precomputed visibility of the environment, and the BRDF. Each of these functions is sparsely represented using the Haar wavelet basis on the six faces of a cubemap. Non-linear approximation (truncation of small coefficients) is used to further compress the lighting and the BRDF representations. The integral is then computed as the sum of products of triplets of non-zero Haar coefficients, where each triplet is weighted by a *tripling coefficient*: the inner product of the corresponding three basis functions.

The efficiency of this approach stems both from the sparsity of the representation of each function and from the sparsity of non-zero tripling coefficients. Ng *et al.* chose the Haar wavelet basis since it enables a sparse non-linear approximation of the illumination and the BRDF, and is also sparse in terms of its tripling coefficients. In the core of their approach lies a dynamic programming algorithm that computes the tripling coefficients in linear time, on the fly at each vertex.

While the above approach is able to produce much higher frequency effects than those of the double-product PRT approaches mentioned earlier, these advantages come at the price of significant storage requirements and much longer rendering times. The storage requirements are dominated by the high-resolution visibility maps that are precomputed and stored for each vertex. Despite using a sparse wavelet representation and quantizing the coefficients to eight bits, Ng *et al.* report using 1.5GB to store the visibility maps for a medium complexity scene (300K vertices). It should be noted that the Haar representation necessitates using mul-

tiple bits per coefficient even though the original visibility maps are binary, and that non-linear approximation was not used for visibility maps to avoid visible artifacts.

Another thorny issue is the representation of the BRDFs. In order to perform the triple product, the three integrands must be aligned in the same (typically global) coordinate system. Since the BRDF functions are typically parametrized around the surface normal, reflected direction [RH02], or half-vector [Rus98], they need to be rotated into the coordinate system of the lighting and visibility maps. Because of this, Ng *et al.* precompute rotated copies of each BRDF in the scene, which further increases the storage requirements, unless low sampling rates are used. This issue effectively limits the glossiness of the BRDFs used and prevents using anisotropic BRDFs.

Spherical harmonics can be used to store and rotate the BRDFs, but this representation is practically limited to low-frequencies. In some situations (double products), the environment map can be rotated or duplicated for different directions [CJAMJ05, JCJ09], eliminating the need to rotate the BRDF. However, this solution is not applicable for triple product integrals, since then the per-vertex visibility maps would need to be rotated instead.

Wang *et al.* [WNLH06] show how wavelet rotations can be precomputed by spanning the rotated bases functions in terms of the original ones. This solution becomes costly if one wants to support fine rotations and is prohibitive in case of anisotropic BRDFs. Xu *et al.* [XJF*08] use another precomputation strategy to store the rotation corrections that need to be done in order to represent rotations in an adaptive piecewise constant basis induced by the segmentation of the environment lightmap. However, the corrections must be recomputed when the lightmap changes.

In this paper we show that many of these issues can be resolved or avoided altogether by employing a sparse, derivative-based representation for visibility, and by evaluating the needed BRDF coefficients on the fly.

Triple product wavelet integrals were later extended to handle dynamic glossy objects [SM06], as well as to affine wavelet integrals in order to handle near-field illumination [SR09]. In this work we focus on the core triple product issues, none of which are eliminated by these extensions.

Some authors forgo perfect reconstruction of the visibility and employ lossy, but more computationally convenient alternatives. For example, Wang *et al.* [WRG*09] use PCA to extract tens to hundreds eigenvectors of the (spherical) signed distance functions which are used to represent the visibility maps and to interpolate them between vertices, thus reducing the tessellation density compared to previous triple-product systems. While this method offers a compact visibility representation and features excellent performance on moderately sized scenes (30k vertices), the visual impact of the lossy compression is visible even in simple scenes (see Figure 9 of Wang *et al.* [WRG*09]). Another disadvantage of this method is its long pre-computation times, e.g., 75 minutes for a scene with 27.5k vertices. Our method uses more memory, but offers a lossless representation of visibility and significantly lower pre-computation times (9 seconds for a scene of the same size), enabling interactive rendering of much more complex scenes.

After explaining our method, in Section 6 we further discuss the positioning of our method with respect to other PRT methods, as well as interactive global illumination methods.

## 3. Triple Product Integrals

In this section we briefly review the key ideas and equations behind triple product integrals, following the notation used by Ng *et al.* [NRH04] and Ramamoorthi [Ram09]. In the next section we present our new approach for efficient numerical approximation of these integrals.

The direct lighting in a scene is given by the following simplified version of the rendering equation [Kaj86]:

$$B(x,\omega_o) = \int_{S^2} L(x,\omega) V(x,\omega) \rho(x,\omega,\omega_o) \max(0,\omega\cdot\mathbf{n}(x)) d\omega, \tag{1}$$

where $B(x,\omega_o)$ is the radiance reflected from point $x$ in direction $\omega_o$, $L(\omega)$ is the unoccluded lighting incident from direction $\omega$, $V(x,\omega)$ specifies the visibility of the environment at $x$ in direction $\omega$, $\rho$ is the BRDF, and $\mathbf{n}(x)$ is the surface normal at $x$. By folding the cosine term into the BRDF, and fixing the position $x$ and the outgoing direction $\omega_o$, this equation may be rewritten more compactly as:

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

By expanding each of the integrands in terms of basis functions $\Phi$, $\Psi$, and $\Upsilon$, respectively,we can rewrite (2) as a sum:

$$\begin{aligned} B &= \sum_j \sum_k \sum_l L_j\, V_k\, \tilde{\rho}_l \int_{S^2} \Phi_j(\omega)\, \Psi_k(\omega)\, \Upsilon_l(\omega)\, d\omega \\ &= \sum_j \sum_k \sum_l L_j\, V_k\, \tilde{\rho}_l\, C_{jkl} \end{aligned} \tag{3}$$

The terms $L_j, V_k, \tilde{\rho}_l$ are the coefficients of the lighting, visibility, and the BRDF in their bases, while $C_{jkl}$ are the tripling coefficients, which depend only on the basis functions. The number of tripling coefficients may vary with the choice of basis, and a thorough analysis for several bases is provided by Ng *et al.* [NRH04].

Much of the original Ng *et al.* [NRH04] paper is concerned with the choice of a suitable basis (Haar wavelets) and the efficient evaluation of the triple product in (3) at each vertex. In the remainder of this paper we show that by breaking the computation into blocks (each corresponding to a limited solid angle), we can *avoid* the triple product computation in a large majority of these blocks, replacing it by a

much simpler computation. In order to explain our approach in the next section, we first need to point out two simple observations.

First, note that if the visibility function in (3) is a non-zero constant, the triple product is reduced to a double product. Furthermore, if $\Phi$ and $\Upsilon$ are chosen to be the same orthonormal basis, this double product is given by the dot product between the coefficients of $L$ and $\tilde{\rho}$. While the visibility cannot be 1 over the entire hemisphere in a non-convex scene, when considering smaller solid angles, this will be the case in a large portion of the cases.

Second, consider the case where the BRDF is constant. For reasons that will become clear below, we are not interested in representing the visibility in the same basis as the lighting. However, note that if we use biorthogonal bases to represent these two functions, the summation in (3) becomes a dot product once again. Obviously, a general BRDF is not constant over the entire hemisphere, nor is it piecewise-constant. However, for a given combination of normal, incident, and outgoing directions, a typical BRDF can be *locally* well approximated by a constant value over many smaller solid angles away from the BRDFs specular lobes. We discuss this in more detail in Section 4.1.

## 4. Method

Armed with the insights from the previous section, we are now ready to introduce our method for fast all-frequency relighting. Following Ng *et al.* [NRH04], we use a cubemap to discretize the space of all directions on the sphere, and work with cubemaps with resolution of either $64 \times 64 \times 6$, or $128 \times 128 \times 6$. But rather than computing the triple product over entire cubemap faces, we subdivide each face into blocks of size $16 \times 16$ and perform the integral over each block independently. In other words, we decompose the integration domain to either 96 or 384 disjoint solid angles, each of which may be sampled at up to 256 directions. Larger block sizes increase the sparsity of represented maps but decrease the likelihood for degenerate triple products. To account for the fact that not all cubemap pixels correspond to the same solid angle, we first scale the incident radiance at each pixel by a suitable correction factor.

All of the cubemaps are represented in the same global world coordinate system. The visibility maps are precomputed at each vertex, and the BRDF is evaluated on the fly, as described in Section 4.1. Thus, they could have been represented in any coordinate system. However, it is more efficient to have all cubemaps oriented identically, since in this way we avoid reprojecting the lightmap onto each individual cubemap.

For each block we distinguish between four conceptually simple cases. The simplest case is when the visibility map is zero over the entire block, which means the environment is occluded over the entire solid angle, and contributes nothing

**Algorithm 1** *Single Block Integral*

```
1:  if Visibility == Constant(0) then
2:      return 0
3:  end if
4:  ρ̃ = ProjectBRDF(ρ, N, ω_o)                    // Section 4.1
5:  if Visibility == Constant(1) then
6:      return Light-BRDF-Product(L_DCT, ρ̃)
7:  end if
8:  if ρ̃ = Constant(ρ̃_1) then
9:      return ρ̃_1 Light-Vis-Product(L_int, V_d)   // Section 4.2
10: else
11:     return Decompress-Integrate(L, ρ̃, V_d)    // Section 4.4
12: end if
```

| | Line 6 | Line 9 | Line 11 |
|---|---|---|---|
| Bas-relief | 43.6 | 55.7 | 0.7 |
| Round Table | 54.2 | 45.7 | 0.1 |
| Sappho | 59.5 | 38.9 | 1.6 |
| David | 47.5 | 50.0 | 2.5 |
| Piano | 55.3 | 43.9 | 0.8 |
| Skeleton | 60.8 | 39.2 | 0.0 |
| Dining Table | 51.0 | 49.0 | 0.0 |
| Bicycle | 46.6 | 52.0 | 1.4 |
| Museum | 42.4 | 57.6 | 0.0 |
| Lucy | 67.7 | 30.8 | 1.5 |

**Table 1:** *Breakdown (in percents) of the blocks into the three main cases of Algorithm 1, averaged over a flyby of several test scenes.*

to the integral. The second case is when the entire block is unoccluded. In this case, the computation amounts to a dot product between the coefficients of the environment lighting and the BRDF, both represented in the same orthonormal basis. The third case is when the visibility map is non-constant over the block, but the BRDF can be well approximated by a constant. Here, we compute a dot product between the coefficients of the lighting and the visibility represented in biorthogonal bases. Finally, only if neither of the three cases above hold, we efficiently evaluate a triple product integral using the standard (pixel) basis, which is made possible by our representation of the visibility (Section 4.2). The logic described above is summarized in pseudocode in Algorithm 1. The breakdown to the different cases outlined above varies with the scene geometry and with the materials used. Table 1 summarizes the breakdown recorded during a flyby of several test scenes, showing that the cheaper dot products in lines 6 and 9 are used for the majority of the blocks.

In the remainder of this section we describe various aspects of the solution outlined above in more detail. In particular, in order to reduce as much as possible the amount of work in each of the cases, we should carefully choose the basis functions with which to perform the different products.

### 4.1. Projecting the BRDFs

As mentioned earlier, in our approach it is not necessary to precompute and store rotated duplicates of each BRDF in the scene. We support both analytic and measured BRDFs, including anisotropic and spatially varying ones, by sampling the BRDF at each vertex and generating the coefficients needed to perform the products in lines 6, 9, and 11 of Alg. 1 on the fly, as we process each block.

When sampling the BRDF over a block, we first determine whether it is *effectively band-limited* there: i.e., whether it can be faithfully approximated using the first 1, 4, or 16 DCT basis functions over this $16 \times 16$ block. If this is found to be the case, we project the BRDF onto the corresponding DCT basis functions [Mal08]. Since in this case we know that both the signal (BRDF) and the basis functions are band limited over the block, we perform the projection efficiently by subsampling both the signal and the projection matrix. For example, to approximate the projection onto the first four DCT basis functions, we evaluate the BRDF at $2 \times 2$ locations inside the block, and multiply the resulting four values by a $4 \times 4$ projection matrix.

More formally, let $\rho(x,y)$ denote the value of the BRDF corresponding to the $(x,y)$ pixel in a block of $N \times N$ pixels. We solve the following linear equations,

$$\sum_{u,v} A_{uv} \cos\left(x\frac{2\pi u}{N}\right) \cos\left(y\frac{2\pi v}{N}\right) = \rho(x,y) \qquad (4)$$

where $A_{uv}$ are the DCT coefficients that we solve for, corresponding to the few lowest frequencies $0 \le u,v \le l$. The choices of 1,4, and 16 coefficients mentioned earlier correspond to $l = 1,2$, and 4, respectively. Each block dimension is sampled with $l$ samples, each corresponding to a pixel at $(x,y)$, resulting in $l^2$ equations of the form in (4). The inverse matrices for the three values of $l$ are precomputed once and for all.

If the first 16 DCT basis functions are unable to accurately represent the BRDF inside a block (e.g., the block contains a narrow spike of a highly-specular BRDF), we forgo any attempts for a band limited approximation and project the BRDF onto the standard basis (sample it at each of the 256 directions). Note that in these "worst case scenarios" our computations use the same resolution as the finest level Haar coefficients in the original method of Ng *et al.*, and hence we are able to handle the same frequencies of lighting and BRDF as they do.

Thus, our first task is to determine the number of coefficients needed in order to represent BRDF in a given block. For analytical BRDFs, where the diffuse and the specular components are easily decoupled, we simply increase the sampling rate in the vicinity of a highlight. In the case of a Lambertian BRDF we approximate the BRDF as a constant over each block. When using the Phong model the number of coefficients is determined by two factors: the angle between the view direction and the reflected vector (obtained by reflecting the block's center direction around the normal), as well as the value of the Phong specular exponent. The exact logic to determine the number of coefficients is given in Appendix A. Similar strategies can be tuned for other analytical BRDFs.

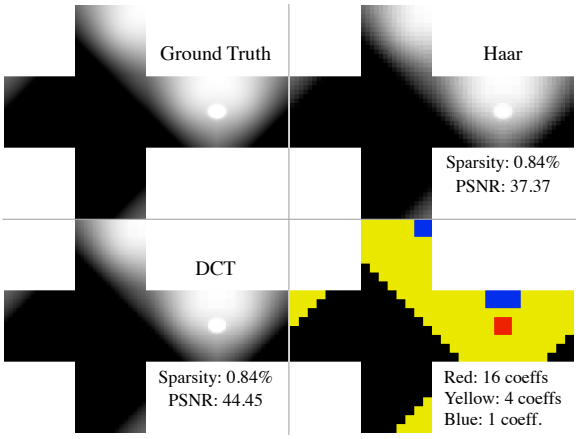



**Figure 2:** *Comparison between a per-block DCT-based representation of a highly specular Phong BRDF ($s = 200$) and the Haar wavelet representation used by Ng et al. [2004]. Using the same sparsity (0.84 percent of non-zero coefficients), the DCT-based representation achieves better accuracy. The bottom right image shows the number of DCT coefficients used in each block.*

Figure 2 demonstrates that the accuracy of our DCT-based approximation for Phong reflectance with a high specular exponent ($s = 200$) is actually slightly better than the Haar-based approximation, when using the same number of coefficients (around 1 percent). Note that the number of DCT coefficients used in each block adapts itself to the view direction, and depends only on how constant or smooth the BRDF is within the block, but not on the magnitude of the BRDF. In practice, we found that using even fewer coefficients for Phong and Ward BRDFs still produces images which are visually almost indistinguishable from those generated with a full evaluation of the integral, so long as the specular lobes are sampled densely enough, as described in Appendix A.

Measured BRDFs are typically provided as a table of measurements. In most cases the angular resolutions of such tables is rather low. For example, the BRDFs in the CURET database [DVGNK99], which were used by Ng *et al.* [NRH04] contain only 205 samples each, and these BRDFs are therefore quite smooth. In particular, the angular resolution of our cubemaps typically exceeds the number of measurements. Thus, given a measured BRDF, we can precompute a 4D lookup table whose angular resolution roughly matches the number of blocks that we use. As each block is processed, we can sample the lookup table using the

direction corresponding to the center of the block, and use a constant approximation of the BRDF over that block.

Table 2 reports the breakdown of the number of BRDF samples recorded during a flyby of several test scenes, showing that in a vast majority of cases the BRDF is only sampled once per block.

| Samples | 1 | 4 | 16 | 256 |
|---|---|---|---|---|
| Bas-relief | 89.4 | 9.2 | 0.7 | 0.7 |
| Round Table | 98.8 | 0.9 | 0.2 | 0.1 |
| Sappho | 95.2 | 0.3 | 0 | 4.5 |
| David | 94.7 | 0 | 0 | 5.3 |
| Piano | 97.9 | 0.6 | 0 | 1.5 |
| Skeleton | 97.2 | 2.8 | 0 | 0 |
| Dining Table | 99.5 | 0.5 | 0 | 0 |
| Bicycle | 93.5 | 4.0 | 0.3 | 2.2 |
| Museum | 96.5 | 3.5 | 0 | 0 |
| Lucy | 97.0 | 0 | 0 | 3.0 |

**Table 2:** *Breakdown (in percents) of the BRDF sampling rates, averaged over a flyby of several test scenes.*

### 4.2. Representing the visibility maps

Our representation of visibility is based on the observation that occluding geometry tends to create contiguous spans in the visibility map. Thus, given a visibility map at a vertex, we can represent it compactly by storing only the *changes* in the map. This idea is applicable to both binary and non-binary (grayscale) visibility maps.

Formally, concatenating together the rows of the visibility map inside a block we obtain a one-dimensional *visibility vector* $V$, and transform it into a vector of derivatives $V_d$:

$$V_d = D \cdot V = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{pmatrix} V \qquad (5)$$

As observed earlier, if the lighting and the visibility are represented in biorthogonal bases, their double product becomes a dot product. Thus, in order to perform the computation in line 9 of Alg. 1 as a dot product, we similarly concatenate the lighting environment map inside the block into a one-dimensional vector $L$, and transform $L$ by the transposed inverse of the operator $D$:

$$L_{int} = D^{-T} L = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 0 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} L \qquad (6)$$

Thus, $L_{int}$ is a vector of partial integrals of $L$ (partial sums of a one-dimensional vector). The actual dot product $L_{int}^T V_d$ is computed while skipping over any entries where $V_d$ is zero.

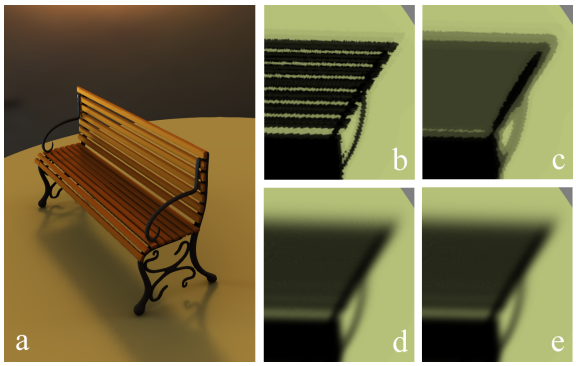



**Figure 3:** *Shadow appearance under binary and non-binary visibility. (a) Binary visibility is sufficient for environments without highly concentrated light sources. (b) Under harsh-lighting (one pixel lit in the environment map) it creates aliasing artifacts. (c) Shadow produced using 2-bit visibility, (d) 6-bit visibility, and (e) grayscale visibility with no quantization.*

Our representation is lossless and particularly sparse in the case of binary visibility functions maps, which were used by Ng *et al.* [NRH04]. In this (binary) case, we avoid storing the visibility derivative values and store only the indices where the value changes from zero to one, or vice versa, resembling the CCITT group 3 format for fax transmission. This is in contrast to representing the visibility using the Haar basis (as done in [NRH04]), where the quantized non-zero coefficients need to be stored, besides the indices.

Our scheme does not support random access, but in our method we only need to access the visibility values sequentially (in lines 9 and 11 of Alg. 1). Our implementation uses visibility vectors of length 256, so the indices are unsigned 8-bit integers. For compact secondary storage of these indices we use the base 128 varints of protocol buffer encoding [Goo12]. Table 3 reports the numbers of visibility indices and the amount of storage needed to store them for several of our test scenes.

While binary visibility maps suffice for environment lighting without strongly concentrated light sources (Figure 3a), harsher lighting might result in aliased shadows, as shown in Figure 3b. In this case, it is better to use a non-binary visibility map. A higher resolution binary map is smoothed and downsampled to yield a grayscale visibility map, whose derivatives are then quantized and stored in addition to the indices. As demonstrated in Figure 3d-e (and the supplementary video) using derivatives quantized to six bits typically suffices for smooth anti-aliased dynamic shadows even under harsh lighting.

Compared to binary visibility, non-binary visibility reduces rendering performance by roughly 15–20 percent, and increases the required visibility storage up to a factor of 2 (when quantizing the visibility coefficients to 8 bits).

### 4.3. Representing the lighting

As may be seen in Alg. 1, the lighting (environment map) $L$ participates in three different products (lines 6, 9, and 11). In order to evaluate each of these products efficiently, we maintain three different representations simultaneously for each block: (i) a set of DCT coefficients for the product in line 6; (ii) projected onto the 1D integral basis inside each block, as described in Section 4.2, for the product in line 9; (iii) standard (pixel) basis with the rows of each block concatenated into a one-dimensional array, for the product in line 11.

Since the lightmap is shared by the entire scene, these three simultaneous representations are computed on the fly when the environment lightmap is changed, supporting dynamically changing the lighting. Note that since we never use more than 16 DCT coefficients to represent the BRDF, it suffices to compute 16 DCT coefficients to approximate the lightmap in each block; whenever the BRDF has higher frequency inside the block we perform the product in the standard basis anyway. Thus, effectively, the lightmap is represented losslessly.

### 4.4. Fast Triple Product — Back to Standard Basis

In blocks where neither the visibility nor the BRDF are constant (Alg. 1, line 11), we perform a full evaluation of the triple product using the standard basis. The standard basis (delta functions) is the most localized orthonormal basis, and thus can handle the product of any three signals over the block, with the only limits imposed by the original pixel resolution of the block. In our implementation we use $16 \times 16$ blocks, so in the worst case the computation involves sampling the BRDF up to 256 times and multiplying the corresponding environment map pixels for each unoccluded direction. As evident from the statistics in Table 2 this case is far from frequent.

The advantage of our visibility representation of visibility is that it can be easily converted to the standard basis on the fly. For binary visibility, we start from the first index in the visibility vector where the visibility is 1, and accumulate the sum of products between the BRDF and lightmap pixels until the next index is reached, meaning that the visibility changes its value to zero. We continue in this manner until the end of the visibility vector, thus quickly skipping any occluded directions. In the case of non-binary visibility, the process is essentially the same up to the fact that the flip-bit is replaced with a counter, which is increased or decreased depending on the value of the differential coefficient. The current value of the counter is used to multiply the BRDF-lighting product. Whenever the counter becomes zero, we can directly skip to the next non-zero visibility index.

## 5. Performance and Results

We implemented our method in C++, on an Intel Core i7-2600K CPU, running at 3.4GHz. Our implementation is almost entirely CPU-based, with only the initial vertex culling phase computed on the GPU (Nvidia GeForce GTX 480). An executable demo of our system with one of our scenes is included as part of the supplementary materials. We take advantage of all four cores in this machine by distributing the vertices among them when rendering each frame. We use the SIMD capabilities of the processors to significantly increase both the BRDF sampling throughput and the actual integration process. For Phong and Ward BRDFs we are using the polynomial approximation for the standard library powf function, obtained by the Remez algorithm [Rem34].

| Model | FPS | Verts | Pre. time | Coeffs | DB Size |
|---|---|---|---|---|---|
| Bas-relief | 5 | 506k | 28 | 10.9 | 700MB |
| Round Table | 14 | 280k | 11 | 13.1 | 290MB |
| Sappho | 17 | 113k | 1.6 | 6.9 | 116MB |
| David | 11 | 293k | 11 | 10.3 | 285MB |
| Piano | 14 | 195k | 4.8 | 9.5 | 148MB |
| Skeleton | 9 | 635k | 42 | 12.2 | 546MB |
| Dining Table | 17 | 381k | 17 | 11 | 285MB |
| Bicycle | 24 | 206k | 5.4 | 12.5 | 152MB |
| Museum | 22 | 231k | 6.7 | 13.7 | 271MB |
| Lucy | 9 | 367k | 16 | 8.0 | 303MB |

**Table 3:** *Statistics for four different test scenes. For each scene we report the average rendering rate, the number of vertices, the pre computation time in minutes, the average number of visibility coefficients per block and the size of the database.*

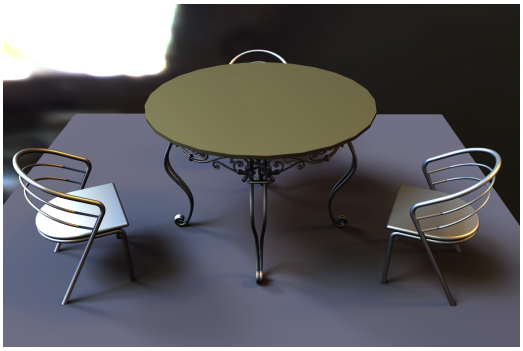
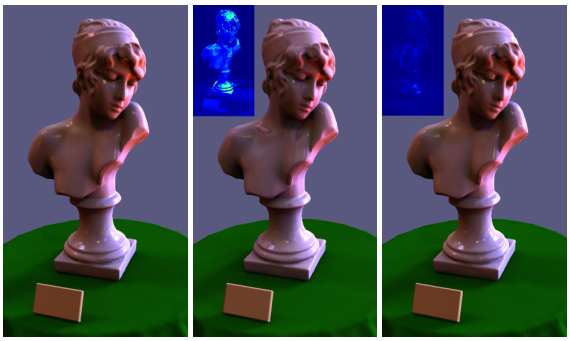


**Figure 4:** *Round table and chairs: a scene similar to the most complex scene shown by Ng et al. [2004].*

**Precomputation:** The simplicity of our differential visibility representation results in relatively fast precomputation times, compared to previous precomputation-based methods. In addition to computing the visibility maps themselves, these methods spend a considerable amount of time converting the maps into an appropriate basis (Haar wavelets, spherical harmonics, spherical signed distance functions, etc.). In contrast, in our method, the transformation to the differential representation takes negligible time compared to the computation of visibility. The visibility map computations, the differentiation of the visibility vectors, and the extraction of



reference Haar (PSNR=33.54) our (PSNR=45.67)

**Figure 5:** *Accuracy of Haar-based triple products and our method. The Haar-based triple product is computed with 1% of the coefficients. The inset in the top left corner shows a color-coded difference image with respect to the reference.*

the non-zero coefficient indices are all done on the GPU. The precomputation times and other statistics for our different test scenes are reported in Table 3.

**Comparison to Haar-based triple products [NRH04]:** Since the models shown by Ng *et al.* were not available to us, we constructed a scene shown in Figure 4 that is comparable in its geometry, number of vertices, and materials to their most complex scene. This scene contains many glossy surfaces with specular exponents up to 200. The binary visibility storage requirements reported by Ng *et al.* for their scene were 1.5GB, and the rendering times were on the order of 3–5 seconds per frame (in 2004). The size of our visibility database is 290MB. The rendering rates that we measured for this scene are around 14 frames per second.

In order to assess the speedup introduced by the wavelet-based triple products integrals, Ng *et al.* compare the running times of their algorithm with various degrees of wavelet compression to a "baseline", which is the time it takes to perform the triple product in the standard basis. With non-linear wavelet approximations of the BRDF and the lightmap that retain 1 percent of the coefficients, their triple products are faster by a factor of roughly 10 than this baseline. Our method, which represents the lightmaps losslessly, achieves a speedup of x40 over the baseline, on average over scenes with specular exponents up to 200. If we use lower specular exponents (up to 10), our speedup factor increases to x50, and if all surfaces are diffuse, we become 120 times faster than the baseline. In addition to the improved rendering speed, our results are also more accurate than those of the Haar-based triple products. For example, for the image shown in Figure 5 the Haar-based approximation results in an PSNR of 33.54, while the PSNR of our result is 45.67. The supplementary materials contain additional accuracy comparisons using a variety of scenes.

The accuracy of our representation, and the efficiency of our product computations enable us to render diffuse surfaces with intricate shadows, along with highly specular surfaces, with dynamically changing full environment lightmaps, viewpoint, and materials, at interactive rates. Figures 1 and 6 show still frames from a number of scenes we have experimented with, and dynamic sequences for several scenes are included in the supplementary video. Note that we deliberately avoid applying textures to the surfaces in these scenes, so as to more clearly show the quality of the shadows and reflections. The vertex counts vary from 113K to 635K, and the frames rates between 5 and 24 frames per second. Tables 1, 2, and 3 report a variety of statistics for several of these test scenes.

Since our system supports dynamically changing BRDF parameters, the frame rates depend on the actual BRDFs used in the scene. The worst case scenario occurs when all the surfaces in the scene have both a non-negligible diffuse component and a highly concentrated specular lobe (diffused mirror/metals), since this scenario requires the highest number of BRDF samples. For example, if all the materials on the frame of the bicycle in Figure 6 are changed to a diffuse, the frame rates go up from 24 to 38.

## 6. Discussion and Limitations

Having presented our method and its results, it is interesting to compare it with several other state-of-the-art real-time rendering methods. In particular, there are now a number of interactive global illumination algorithms, which in principle may be used to compute the illumination of a scene by environment lighting. While a comprehensive survey of such techniques is beyond the scope of this paper (we refer the reader to [RDGK12]), below we discuss a few representative examples. In a nutshell, we argue that these methods typically employ a variety of drastic approximations in order to achieve interactive performance, and this of course comes at a price of reduced accuracy or lower frequency illumination effects.

As a first example, consider state-of-the-art interactive ray tracers (e.g., [PBD*10]), which are able to cast around 100 million incoherent rays per second [RDGK12]. Thus, in order to shade a million points (fragments) per frame at 10fps, such a ray tracer can afford no more than 10 rays per shaded point. Even with importance sampling of the lighting and the BRDF, this clearly results in a fairly coarse approximation of the integral, especially if non-trivial occlusions are involved.

Another example is interactive instant radiosity methods, such as *imperfect shadow maps* [RGK*08]. Leveraging the fact that indirect lighting is typically characterized by low frequencies, this work uses a small number of virtual point lights (VPLs) to represent the indirect lighting and a small number of geometry samples to compute shadow maps. While environment lighting may also be approximated this way, the small number of VPLs (512–1024) and geometry samples (8K per shadow map) effectively limits this approximation to low frequency BRDF.

**Figure 6:** *Additional test scenes we experimented with. From top left: Sappho, David, Piano, Skeleton, Dining Table, Bicycle, Museum. The statistics and rendering rates for these scenes are reported in Table 3. Several of these scenes are shown with dynamic lighting and viewpoint in the supplementary video.*

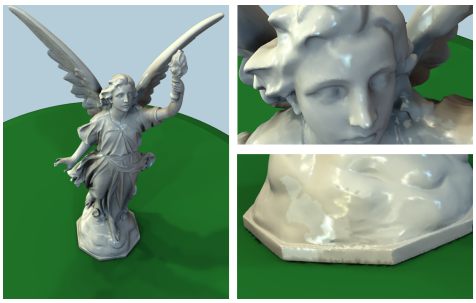


**Figure 7:** *The limits of per-vertex shading may be revealed in zoom-in views (right). The shape of the highlights and the shadows begin to reveal the underlying mesh structure.*

**Limitations:** Like most of the previous triple product methods, the shading in our system is computed on a per-vertex basis. Thus, in order to capture intricate shadow patterns, the receiving surfaces should be tessellated densely enough to avoid interpolation artifacts. Similarly, highly specular surfaces also require fine tessellation to correctly capture fine highlights and reflections. Figure 7 shows that zooming in too strongly might reveal some artifacts in the shadows on the diffuse floor and in the highlights on the specular statue, revealing the underlying triangulation.

Methods that compute the shading product on a per-fragment basis, such as Wang *et al.* [WRG*09] avoid this issue by interpolating the visibility maps between the vertices at which they were computed. This has the advantage of smaller visibility storage and the ability of producing sharp shadows and reflections. It can also be crucial for creating realistic surface reflections, especially when using spatially varying BRDFs. However, as pointed out earlier, this method uses lossy compression of visibility and other approximations which lead to visible inaccuracies. Additionally, the pre-computation times are quite significant, even for scenes with under 30K vertices.

We did not attempt to extend our method to products of more than three functions, such as the generalized wavelet products of Sun and Mukherjee [SM06]. Increasing the number of functions may potentially result in fewer degenerate product cases, and this direction is left for future work.

## 7. Summary and Future Work

We have presented a new approach for all-frequency relighting within the framework of triple product integrals. By evaluating the product independently over separate blocks at each vertex, we are able to flexibly switch between different representations, and avoid sampling and storing rotated copies of each BRDF. Our results show that triple product based systems can produce compelling interactive renderings of complex scenes with dynamic environment lighting, and the simultaneous ability to change the viewpoint, as well as the materials in the scene.

As already mentioned in the discussion, developing a fast, yet accurate, visibility interpolation method may further improve the results of the method, by allowing even faster pre-computation times, in addition to the ability to operate on fragments (and not vertices). We believe that recent work on displacement interpolation [BvdPPH11], as well as the work of Wang *et al.* [WRG*09] show that this direction holds promise.

## References


[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using Lagrangian mass transport. *ACM Trans. Graph. 30* (Dec. 2011), 158:1–158:12.

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Trans. Graph. 24* (July 2005), 1166–1175.

[DVGNK99] DANA K. J., VAN-GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real world surfaces. *ACM Trans. Graph. 18*, 1 (Jan 1999), 1–34.

[Goo12] GOOGLE: Protocol buffers. http://developers.google.com/protocol-buffers, 2012.

[JCJ09] JAROSZ W., CARR N. A., JENSEN H. W.: Importance sampling spherical harmonics. *Computer Graphics Forum 28*, 2 (Apr. 2009), 577–586.

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proc. ACM SIGGRAPH 86* (1986), ACM, pp. 143–150.

[KSS02] KAUTZ J., SLOAN P.-P., SNYDER J.: Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proc. EGRW '02* (2002), pp. 291–296.

[Leh07] LEHTINEN J.: A framework for precomputed and captured light transport. *ACM Trans. Graph. 26* (October 2007), Article 13.

[LSSS04] LIU X., SLOAN P.-P., SHUM H., SNYDER J.: All-frequency precomputed radiance transfer for glossy objects. In *Proc. EGSR 2004* (2004), pp. 337–344.

[Mal08] MALLAT S.: *A wavelet tour of signal processing*, 3rd ed. Academic Press, 2008.

[MTR08] MAHAJAN D., TSENG Y., RAMAMOORTHI R.: An analysis of the in-out brdf factorization for view-dependent relighting. *Computer Graphics Forum 27*, 4 (2008), 1137–1145.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22* (July 2003), 376–381.

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. 23* (Aug. 2004), 477–487.

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: Optix: a general purpose ray tracing engine. *ACM Trans. Graph. 29*, 4 (July 2010), 66:1–66:13.

[Ram09] RAMAMOORTHI R.: Precomputation-based rendering. *Found. Trends. Comput. Graph. Vis. 3* (April 2009), 281–369.

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The State of the Art in Interactive Global Illumination. *Computer Graphics Forum 31*, 1 (Feb. 2012), 160–188.

[Rem34] REMEZ E. Y.: Sur la détermination des polynômes d'approximation de degré donnée. *Comm. Soc. Math. 10* (1934).

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H. P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *SIGGRAPH Asia '08: SIGGRAPH Asia 2008 papers* (Sept. 2008), 1–8.

[RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency space environment map rendering. *ACM Trans. Graph. 21* (July 2002), 517–526.

[Rus98] RUSINKIEWICZ S. M.: A new change of variables for efficient BRDF representation. In *Proc. EGRW '98* (1998), pp. 11–22.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22* (July 2003), 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21* (July 2002), 527–536.

[SM06] SUN W., MUKHERJEE A.: Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Trans. Graph. 25* (July 2006), 955–966.

[SR09] SUN B., RAMAMOORTHI R.: Affine double- and triple-product wavelet integrals for rendering. *ACM Trans. Graph. 28* (May 2009), 14:1–14:17.

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25* (July 2006), 967–976.

[WNLH06] WANG R., NG R., LUEBKE D., HUMPHREYS G.: Efficient wavelet rotation for environment map rendering. In *Proc. EGSR 2006* (2006), Springer-Verlag, pp. 173–182.

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 133:1–133:10.

[WTL04] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proc. EGSR 2004* (2004), pp. 345–354.

[XJF*08] XU K., JIA Y.-T., FU H., HU S.-M., TAI C.-L.: Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE TVCG 14*, 2 (march-april 2008), 454–467.


## Appendix A: BRDF Approximation Strategy

In this appendix we describe in more detail the strategy used in our implementation to determine the required number of DCT coefficients for the Lambertian and Phong BRDFs.

Let $\mathbf{N}$ be the normal at a vertex and $\mathbf{V}$ denote the current viewing direction. We reflect the direction vector corresponding to the center of the block around the normal $\mathbf{N}$ to obtain a reflected vector $\mathbf{R}$. For Lambertian BRDFs, we use a single coefficient over each block where $\mathbf{N}\cdot\mathbf{R} > 0$. To approximate the Phong BRDF over the block, we consider the angle between $\mathbf{V}$ and $\mathbf{R}$, as well as the value of the specular exponent $s$. Specifically, for moderate exponents ($s < 30$) we use four coefficients when $\arccos(\mathbf{V}\cdot\mathbf{R}) < \pi/4$, and treat it as constant (one coefficient) in all other blocks, where it behaves essentially as Lambertian. For stronger specular exponents ($30 < s < 250$), we use four coefficients if $\pi/8 < \arccos(\mathbf{V}\cdot\mathbf{R}) < \pi/4$, and sixteen coefficients if $0 \leq \arccos(\mathbf{V}\cdot\mathbf{R}) < \pi/8$. If the specular exponent exceeds 250, resulting in a highly specular, mirror-like reflectance, we work in the standard basis, sampling the BRDF for each of the directions corresponding to the pixels of the cubemap block, whenever $\arccos(\mathbf{V}\cdot\mathbf{R}) < \pi/8$.