Volume xxx, (1997) number yyy pp. 000–000

Generating Reflected Directions from BRDF Data

Paul Lalonde

Alain Fournier

Department of Computer Science
University of British Columbia
2366 Main Mall,
Vancouver, B.C. V6T 1Z4
Canada
{lalonde|fournier}@cs.ubc.ca

Abstract
Monte-Carlo path tracing algorithms for computer graphics require that given an incident light ray at a surface an outgoing direction can be computed with a distribution given by the magnitude of the bidirectional reflectance distribution function (BRDF). For analytic reflectance functions this can be done using various techniques including inverting the function, or tabulating some representation of the inverse. However, measured BRDF data sets are too large for this to be practical.
We present a method to generate reflection rays distributed according to the magnitude of the BRDF. The method relies on a wavelet-based representation of the BRDF. This representation is efficient and compact, allowing large, anisotropic measured BRDF data sets to be represented with a few thousand coefficients. In particular, we exploit the wavelet representation to quickly compute integrals over ranges of the BRDF.

Keywords: Bidirectional reflectance distribution functions; Wavelets; Monte Carlo path tracing; Reflection; Local Illumination; Random deviates

1. Introduction

Image synthesis algorithms based on Monte-Carlo path tracing[7, 2, 6, 11] have been used to generate impressively realistic images of artificial environments. As these environments become more complex and as the techniques are applied more and more to modeling real environments, the use of reflectance functions measured from real materials becomes more important. To adequately represent these functions a large number of incident and reflected directions must be measured[13, 14].

Monte-Carlo path tracing techniques can benefit dramatically from accurate generation of reflected directions. Generating reflected directions according to the distribution of the BRDF can dramatically reduce the variance of the process. Lafortune and Willems show how using a probability distribution proportional to the BRDF and a cosine factor can reduce variance, but do not give a method for efficiently generating these directions [8]. Shirley *et al.* [11] examine the problem of sampling light sources, but do not include the BRDF's mediating effect on the computation.

1.1. The Bidirectional Reflectance Distribution Function

The bidirectional reflectance distribution function (BRDF) characterizes the reflection of light at a surface [5], assuming that there is no subsurface scattering at a scale larger than the surface element being considered. The BRDF gives the ratio of incident light from a particular solid angle to reflected light at another solid angle at all wavelengths. The bidirectional transmission distribution function (BTDF)similarly describes the transmission of light through a surface. The union of the BRDF and BTDF, the bidirectional transmission and reflectance distribution function (BTRDF) describes both reflection and transmission at a point on a surface. Most of our discussion will refer to the BRDF, but the results are symmetrically applicable to BTDFs and so also to BTRDFs.

More precisely, consider the geometry in Figure 1. When light arrives at a differential area dA through an incident solid angle $d\omega_i$, in direction $\vec{\omega_i}$ (differential solid angles have associated with them a unique direction — when we write $\vec{\omega}$ we are referring to the direction asso-

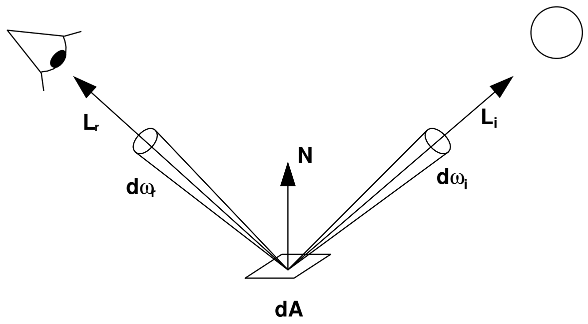


Figure 1: *Shading Geometry*

ciated with the differential solid angle $d\omega$), the amount reflected in some other direction $\vec{\omega_r}$ from dA is proportional to the incident irradiance from $\vec{\omega_i}$:

$$dL_r(\vec{\omega_r}) \propto dE(\vec{\omega_i}), \tag{1}$$

which is to say that an increase in incident light energy per unit area will lead to a proportional increase in the reflected energy. If we include the dependence on wavelength λ, Equation 1 can be written as

$$dL_r(\vec{\omega_r}, \lambda) \propto dE(\vec{\omega_i}, \lambda) \tag{2}$$

The meaning now changes slightly: when light arrives at dA through $d\omega_i$, the amount of that light reflected in some other direction $\vec{\omega_r}$ at wavelength λ is proportional to the incident irradiance from $\vec{\omega_i}$ at wavelength λ The constant of proportionality in Equation 2 is the BRDF. The ANSI/IES standard defines reflection as "the process by which the incident flux leaves a [surface] from the incident side, without change in frequency"[4].Some reflection-like processes do cause a change in frequency but these will not be addressed here. The extensions to do so are straightforward. Likewise, these expressions can be written to account for polarization of light. Again, these will not be examined here. Rewriting and substituting the change in irradiance as a result of the illumination of the patch from $d\omega_i$:

$$dE(\vec{\omega_i}, \lambda) = L_i(\vec{\omega_i}, \lambda) \cos\theta_i d\omega_i \tag{3}$$

we find the BRDF f_r:

$$f_r(\vec{\omega_i} \rightarrow \vec{\omega_r}, \lambda) = \frac{L_r(\vec{\omega_r}, \lambda)}{L_i(\vec{\omega_i}, \lambda) \cos\theta_i d\omega_i}$$

The dependence on four angles is frequently made explicit by writing the BRDF as $f_r(\theta_i, \phi_i, \theta_r, \phi_r, \lambda)$.

For a thorough treatment of the BRDF and its variations see [4].

2. A Wavelet Representation of BRDF Data

A raw tabulated representation of BRDF data is impractical because of the large amount of the data required to adequately represent a BRDF. It is not uncommon to have several hundred different surfaces with different surface properties appear in single frames of large scale commercial animations. If each of these were represented using raw data the scenes could not be computed in a timely fashion because of memory constraints.

Instead we want to find a space efficient representation that is faithful to the original data and quick to use. The method has to be competitive in speed with analytic and algorithmic shaders. In addition, in order to use the BRDF in a Monte Carlo path tracing environment we need a fast method to generate reflected rays with a distribution matching the distribution of the BRDF.

We propose a wavelet based solution for a number of reasons:

1. the representation allows a good control of the accuracy/space trade-off;
2. The point-wise reconstruction is logarithmic in the number of original BRDF samples;
3. error metrics are easy to compute;
4. incremental levels of accuracy can be used;
5. BRDFs are largely smooth with important discontinuities, which can be interpreted as a signal which is mostly low-frequency, but with localized high frequencies — wavelets are ideally suited to represent this kind of data; and
6. the representation allows easy generation of reflected rays whose distribution is given by the BRDF.

2.1. Parameterization

The topology of the BRDF function space is an hemisphere crossed with an hemisphere, or if we consider the transmission as well, a sphere cross a sphere ($S^2 \times S^2$). The most common wavelet formulations are applied on Cartesian spaces. Fortunately there are straightforward embeddings of $S^2 \times S^2$ into R^4 if one is willing to compromise on the matter of redundancy at the poles. For our experiments we chose to use the *Nusselt* embedding, which is closely related to the direction cosines [9].

To express a direction given by a vector $\vec{v}$ in a local coordinate frame given by the normal $\vec{N}$ and surface tangent $\vec{T}$, we let $\mu_x = \vec{v}.\vec{T}$ and $\mu_y = \vec{v}.(\vec{N} \times \vec{T})$. We normalize these to the range (0..1), and call them κ and λ:

$$\kappa = (\mu_x + 1.0)/2.0$$

$$\lambda = (\mu_y + 1.0)/2.0$$

κ and λ then index in the unit square. Directions then cover only the circle at the center of the square. We define the value of any function of directions for which $\kappa^2 + \lambda^2 > 1$ as zero.

2.2. Wavelets and BRDFs

We express a BRDF as its projection onto a multiresolution wavelet space:

$$F(\kappa_i, \lambda_i, \kappa_r, \lambda_r) = \sum_j f_j B_j(\kappa_i, \lambda_i, \kappa_r, \lambda_r)$$

where $B_j(.)$ are the appropriate basis functions, and $\tilde{B}_j(.)$ are their duals. Then the f_j are given by

$$\begin{aligned} f_j &= < F(\kappa_i, \lambda_i, \kappa_r, \lambda_r) | \tilde{B}_j(\kappa_i, \lambda_i, \kappa_r, \lambda_r) > \\ &= \int_0^1 \int_0^1 \int_0^1 \int_0^1 F(.) \tilde{B}_j(.) d\kappa_i d\lambda_i d\kappa_r d\lambda_r \end{aligned} \quad (4)$$

Since our functions are defined as zero when $\mu_x^2 + \mu_y^2 >= 1$, which corresponds to a range of κ and λ of between zero and one, we can safely narrow the bounds of integration from $(-\infty..\infty)$ to $(0..1)$. For a univariate wavelet basis, the basis functions are all derived from two functions by scaling by a factor l (we will us $l = 0$ as the coarsest level of the pyramid, that is a larger l in the discrete case corresponds to a larger number of elements) and translations by m, which can range from 0 to 2^l. One group comes from the *scaling function* $\phi()$:

$$\phi_{lm}(p) = 2^{l/2} \phi(2^l p - m)$$

and the other from the *mother wavelet function* $\psi()$:

$$\psi_{lm}(p) = 2^{l/2} \psi(2^l p - m)$$

To manage the notation for multidimensional wavelet bases, we will use a four-variable vector $\mathbf{q}$:

$$\mathbf{q} = (\kappa_i, \lambda_i, \kappa_r, \lambda_r)$$

and a multi-resolution index $\mathbf{j}$:

$$\mathbf{j} = (\nu^j, l^j, m^j_{\kappa_i}, m^j_{\lambda_i}, m^j_{\kappa_r}, m^j_{\lambda_r})$$

Here ν is the *selector index* for the wavelet functions, l is the level in the wavelet subspaces, 0 being the coarsest, and m_p the offset at that level for parameter p. The level is the same for all variables because we use non-standard multidimensional wavelet bases[3, 12]. Since they all have an hyper-cube support (same extent for all variables at a given level) the data structure is simpler, as are many geometric operations. The selector ν determines the product of wavelet and smooth basis functions that together define $B_j(\mathbf{q})$. We use a special notation to indicate the selection, where $\nu[h]$ is interpreted as the hth bit of ν:

$$\Gamma^h_{l,m,\nu}(p) = \left\{ \begin{array}{ll} \phi_{l,m} & \text{if } \nu[h] = 0 \\ \psi_{l,m} & \text{if } \nu[h] = 1 \end{array} \right\}$$

Then we can write the basis function $B_j(\mathbf{q})$ as:

$$\begin{aligned} B_j(\mathbf{q}) &= \Gamma^0_{l^j, m^j_{\kappa_i}, \nu^j}(\kappa_i) \times \Gamma^1_{l^j, m^j_{\lambda_i}, \nu^j}(\lambda_i) \times \\ &\quad \Gamma^2_{l^j, m^j_{\kappa_r}, \nu^j}(\kappa_r) \times \Gamma^3_{l^j, m^j_{\lambda_r}, \nu^j}(\lambda_r) \end{aligned} \quad (5)$$

2.3. Point Reconstruction

The difficulty with this formulation is that although it is efficient to reconstruct the entire original BRDF from the wavelet coefficients using the fast wavelet transform[1] we can not afford the space or time to do so while rendering, unless we know that most directions will be used, which is unlikely. Instead we want the BRDF only for specific incident and reflected angles. Likewise we would prefer not to examine every non-zero coefficient during the evaluation step - which could be required if our basis functions $\psi()$ have a large width of support.

Fortunately orthogonal wavelet bases can have very narrow width of support (the range within which the coefficients are non-zero) in the reconstruction wavelets allowing us to avoid looking at those coefficients which fall outside the width of support during the reconstruction. For simplicity of exposition, let us consider the one dimensional transform. A given signal $f(x)$, sampled at 2^n points and represented by its wavelet transform coefficients $c_k, k = 0..2^n$, can be reconstructed at x by evaluating:

$$f(x) = c_0 \phi(x) + \sum_{l=0}^{n-1} \sum_{m=0}^{2^l - 1} c_{2^l+m} \psi_{l,m}(x).$$

Knowing that $\psi_{l,m}(x) = 0$ outside of the range $m_1 \leq m \leq m_2$ for some m_1 and m_2 and x we can rewrite this as

$$f(x) = c_0 \phi(x) + \sum_{l=0}^{n-1} \sum_{m=m_1(l)}^{m_2(l)} c_{2^l+m} \psi_{l,m}(x). \quad (6)$$

If our chosen wavelet has finite width of support then the reconstruction cost is logarithmic in the number of coefficients of our representation, since at most a constant number of coefficients need to be examined at each level of the reconstruction. If the width of the wavelet is w and the value x at which we are evaluating is closest to coefficient i at a given level l then $m_1 = i - w \bmod 2^l$ and $m_2 = i + w \bmod 2^l$. The modulus is taken since standard wavelet transforms and reconstructions are defined on a space that wraps at the boundary.

2.4. Choice of Basis

Our choice of wavelet basis functions is determined by two properties: the width of support of the basis, and the amount of compression a basis can provide. Since one of our goals in using the wavelet representation is to reduce the memory requirements we see immediately why greater compression is a goal. It is not so obvious why the width of support of the basis is important, or how it relates to compression rates.

Firstly, the narrower the width of support of a wavelet, the less computational work is required in the reconstruction, since the scaled and translated wavelets will cover fewer terms. When extending to higher dimensions the amount of work required by wider bases increases with the power of the dimension. The number of terms examined then becomes a dominant factor in the cost of the reconstruction. In the case of bi-orthogonal wavelets all we care about is the width of the reconstruction wavelet. The analysis wavelet can be as wide as necessary, since the wavelet transform is performed off-line using the fast wavelet transform.

Secondly, wider bases tend to compress better. The number of vanishing moments of a wavelet tells us the highest degree polynomial that that wavelet basis can represent exactly with only as many terms as the width of the basis. This ability to represent high-order polynomial correlates strongly with the amount of compression that a basis provides. In general, wavelets with more vanishing moments have a wider width of support. Thus, wider bases provide better compression, but at the cost of much more expensive computation.

It follows then that the best bases for our application will compress the data well, while being as narrow as possible. It is only possible to determine experimentally which bases are most appropriate for our application, since the structure of the BRDF data is not fixed or easily characterizable.

2.5. Representation

The remaining difficulty is in representing the coefficients. The major advantage of using a wavelet based representation is that the representation is sparse, with many of the wavelet coefficients being zero or small. To increase our compression rate at the cost of accuracy we threshold small coefficients to zero. A property of the wavelet transform is that the sum of these thresholded coefficients gives us the root mean square of the error induced in the reconstruction by ignoring the thresholded terms. The difficulty is that there are as many wavelet coefficients as original data points, even though a large number of them may be zero. Thus we want to find a representation that stores only the non-zero coefficients, allowing us considerable economies in storage.

An efficient option for the representation is as a *Wavelet Coefficient Tree*. This tree is a sparse hexadecary tree (16 ordered children), where the depth of the nodes indicates the depth of the coefficients in the representation, and where each node stores the non-zero coefficients with the same offset m, indexed by ν, the wavelet basis selector.

We use a structure called a *wavelet node* that stores all non-zero wavelet coefficients with the same index vector (l, m_0, m_1, m_2, m_3), and different basis selector ν. The wavelet node also maintains a list of non-empty subtrees rooted at a wavelet node with indices $l+1, 2m_0+b:0, 2m_1+b:1, 2m_2+b:2, 2m_3+b:3$ where $b = 0..15$ and $b:i$ indexes the ith bit of b. The addition of 2 16-bit masks indicating which coefficients (the value mask) and which children (the child mask) are present allow very compact storage of the tree.

This method is both simple to implement and efficient. The tree traversal for an evaluation is comprised of a number of bit-mask operations and recursion on the found pointer (Figure 2).

```
function Eval(WaveletNode wn, q)
r = 0
for b = 0..15
    if bit b of wn.ValueMask is set
        j = (b, wn.l, wn.m[0..4])
        r+ = B_j(q)
for b = 0..15
    if bit b of wn.ChildMask is set
        j = (b, wn.l + 1, 2wn.m[0] + b : 0, 2wn.m[1] + b : 1,
                    2wn.m[2] + b : 2, 2wn.m[3] + b : 3)
        if q is in the support of B_j(.)
            r += Eval(wn.Child[b], q)
return r
```

Figure 2: *Reconstruction at a point of a 4-variable Wavelet compressed function at point* $\mathbf{q}$

3. Generating Reflected Rays

When performing Monte Carlo path tracing operations it is important to generate rays in a distribution determined by the BRDF. In general an incident direction is provided, and a reflected ray must be computed. These reflected rays should be distributed according to the magnitude of the BRDF when the incident direction is held fixed. Generating rays distributed can be difficult. In general the problem requires that the BRDF be integrated and tabulated. Although it is feasible, the resulting table is by nature very large.

There is however some advantage to be gained from using our wavelet representation. The wavelet representation can be used to integrate the function in the same amount of time as an evaluation of the BRDF. In particular, a Haar transformed version of the data lends itself to efficient computation of reflected data.

We will begin by examining the univariate case, and from there extend the result to higher dimensions and show how it works with our BRDF data in a Monte-Carlo path tracing framework.

3.1. Generating Random Deviates

The problem is to generate random deviates according to given probability distribution. Given a uniform probability distribution function so that the probability of generating a number between x and $x+dx$, denoted $p(x)dx$ is uniform on the range $0 < x < 1$, we can take a given function of it, say $y(x)$ and find that its probability distribution $y(x)dy$ is given by

$$p(y) = p(x)|\frac{dx}{dy}|$$

To generate some distribution of y then, say one with $p(y) = f(x)$ we need to solve

$$\frac{dx}{dy} = f(y).$$

If the indefinite integral of $f(x)$ is $F(x)$ then

$$y(x) = F^{-1}(x).$$

The geometric interpretation is that one should choose a uniform random x and find the value y at which the area under the curve $f(x)$ to its left has fraction x of the total area [10].

3.2. Univariate

In most cases we do not have an analytic representation of $F^{-1}(x)$. In our particular case though, we can easily compute some values of $\int_a^b f(x)dx$ which we can use to our advantage to let us evaluate $F^{-1}(x)$ by searching.

The key observation is that the smooth terms of a Haar wavelet-compressed signal gives the average value of the signal in the region covered by the smooth terms. By performing a binary search down the hierarchy of wavelet coefficients we can then find the point at which the area to the left of that point is equal to our uniform random deviate x. The detail that must be attended to is a scaling terms for the total area under the curve. Fortunately the top-level smooth term gives us the area under the entire signal. The cost of this algorithm is then $O(\log_2 n)$ where n is the number of samples in the original signal. The cost can be lowered at the cost of accuracy by thresholding the wavelet coefficients.

Consider the Haar transform of a discrete signal $f(x)$ defined over the domain 0..1, whose wavelet coefficients are $w_{l,m}, l = 0..d, m = 0..2^l - 1$, and w_s is the smooth term. Then

$$f(x) = w_s\phi(x) + \sum_{l=0}^{d}\sum_{m=0}^{2^l-1} w_{l,m}\psi_{l,m}(x) \tag{7}$$

Integrating we find:

$$\int_a^b f(x)dx = w_s\int_a^b \phi(x)dx + \sum_{l=0}^{d}\sum_{m=0}^{2^l-1} w_{l,m}\int_a^b \psi_{l,m}(x)dx \tag{8}$$

It must also be recalled that for any wavelet $\int_{-\infty}^{\infty}\psi(x)dx = 0$. In the particular case of the Haar basis a wavelet $\psi_{l,m}(x)$ is supported entirely in the range $(m/2^l, (m+1)/2^l)$, and zero everywhere else. Consider what happens when we reconstruct our function $\int_a^b f(x)dx$ only considering the wavelet coefficients $w_{l,m}, l = 0..k$, effectively truncating the evaluation at a certain level. Let the function $f^k(n)$ represent the reconstruction of $f(x)$ obtained by truncating the evaluation at level k:

$$f^k(x) = \phi + \sum_{l=0}^{k}\sum_{m=0}^{2^l-1} w_{l,m}\psi_{l,m}(x) \tag{9}$$

The error induced is then

$$e^k(x) \;=\; f(x) - f^k(x) \tag{10}$$

```
function Search( y, l, r, total, φ, d, m )
if total = y return r
   if total + (r−l)/2 (φ + w_{d,m}) < y
      return Search( y, (l + r)/2, r,
          total + (r−l)/2 (φ + w_{d,m}), φ − w_{d,m}, d + 1, 2m + 1)
   else
      return Search( y, l, (l + r)/2, total,
          φ + w_{d,m}, d + 1, 2m )
```

Figure 3: *Generating a random deviate from a Haar-transformed signal.*

$$= \sum_{l=k+1}^{d}\sum_{m=0}^{2^l-1} w_{l,m}\psi_{l,m}(x) \tag{11}$$

If we now consider the error induced by integrating $f^k(x)$ rather than $f(x)$ we find:

$$\int_a^b f(x)dx - \int_a^b f^k(x)dx = \int_a^b e^k(x)dx \tag{12}$$

and

$$\int_a^b e^k(x)dx = \sum_{l=k+1}^{d}\sum_{m=0}^{2^l-1} w_{l,m}\int_a^b \psi_{l,m}(x)dx \tag{13}$$

However, $\int_a^b \psi_{l,m}(x)dx = 0$ unless the interval (a, b) intersects the interval $(\frac{m}{2^l}, \frac{m+1}{2^l})$, since that is the entire width of support of the wavelet $\psi_{l,m}(x)$. It is then the case that

$$\int_{\frac{m}{2^k}}^{\frac{m+1}{2^k}} f(x)dx = \int_{\frac{m}{2^k}}^{\frac{m+1}{2^k}} f^k(x)dx \tag{14}$$

In our particular application we are interested in performing a search to find some value b such that given y:

$$y = \int_0^b f(x)dx \tag{15}$$

We can do this using a binary search technique Observing that the binary search will follow the dyadic divisions of the wavelet transform, we find that we can compute $\int_l^{(l+r)/2} f(x)dx$ incrementally at each iteration, by keeping track of the smooth term during the traversal: Other wavelet bases are not as accommodating as Haar, and require that the evaluation be performed to the bottom level of the wavelet coefficient tree. The cost then becomes $O(\log^2 n)$ where n is the number of terms, since the search requires an $O(logn)$ reconstruction at each of the $O(logn)$ levels.

3.3. Multivariate

Extending the previous result to higher dimensions is straightforward. In particular, in 4 dimensions we use a hexa-decary search instead of a binary search, and find that the search space is again divided along the dyadic subdivisions of the input. It is easy to show that because this subdivision is dyadic that the finer-scale terms need not be examined to find the integral over the subdivision regions. The key observation is that if any of the Γ^h functions that comprise the basis functions $B_{l,m}(.)$ at a depth l of the search is the wavelet then the value of the integral of $B_{l,m}(.)$ over the range of the search is zero. Likewise the integral of $B_{l',m'}(.)$ for $l' > l$ will also be zero. The only time that the $B()$'s don't include a wavelet component is when the $B(.)$ is a pure smoothing term, but this only happens at the root of the wavelet tree and can be dealt with as a special case.

3.4. Multivariate with Fixed Inputs

The problem becomes more interesting when two variables are held fixed, which is the case when we are performing Monte-Carlo path tracing, and requesting a representative distribution of reflected rays. The integral to search for a reflected ray is:

$$f(\kappa_i, \lambda_i) = \int \int F(\kappa_i, \lambda_i, \kappa_r, \lambda_r) d\kappa_r d\lambda_r$$

Writing this in wavelet terms and leaving within the integral only function of κ_r and λ_r we find:

$$\begin{aligned} f(\kappa_i, \lambda_i) = & \sum_l \sum_m \sum_{\nu=0}^{15} f_{l,m,\nu} \Gamma^0_{l,m_{\kappa_i},\nu}(\kappa_i) \Gamma^1_{l,m_{\lambda_i},\nu}(\lambda_i) \\ & \times \int \Gamma^2_{l,m_{\kappa_r},\nu}(\kappa_r) d\kappa_r \int \Gamma^3_{l,m_{\lambda_r},\nu}(\lambda_r) d\lambda_r \end{aligned}$$

The terms within the integrals are easy to compute. The integrals of the basis functions can be tabulated for fast lookup, or evaluated analytically in the case of simpler bases, such as Haar. Pseudo-code for the integration is given in Figure 4. The search itself is a straightforward quad-tree search over the range of the integration variables κ_r and λ_r.

4. Results

Searching using the algorithm presented above can be slow - a great many terms might examined. There are two ways to reduce the amount of work required to generate the reflection rays. By thresholding the wavelet representation of the BRDF the number of terms that need to be examined can be reduced. The result is that the distribution of reflected rays is the same as some simplified version of the BRDF. Figure 5 examines the root-mean-squared error (normalized by the number of samples) in reconstructions of the BRDF obtained by generation large numbers of reflected directions for different levels of thresh-holding (the reconstructions used here used 10000 samples per incident direction).

Figure 6 shows a ray-traced environment in which the direction of the reflected rays is determined using our method. The sphere's surface is a shiny brushed metal (isotropic), the floor a Phong reflector with an exponent of 30, the rear wall a twill weave, and the side walls Lambertian. The brushed metal and twill weave were sampled with a virtual gonioreflectometer. The image on the left is computed using Haar-transformed versions of the BRDFs; the image on the right uses a linear spline basis. Both images were generated at 9 samples per pixel.

5. Conclusion

We have presented a method that allows the use of real BRDF data to generate reflected rays in a Monte-Carlo path tracing context. The method uses a thresholded wavelet representation that allows the user to select select for accuracy versus running time. Experiments have shown that the reflected rays are generated in a distribution that is close to the original BRDF, even in cases when the thresholding is radical, or when the evaluation procedure is terminated early. The value of the BRDF in the reflected direction is easily obtainable from the same dataset.

References

1. G. Beylkin, Ronald R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.

2. Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 165–174, July 1991.

3. Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1992.

4. Andrew Glassner. *Principles of Image Synthesis*. Morgan Kaufmann, San Francisco, 1995.

5. American National Standards Institute. ANSI standard nomenclature and definitions for illuminating engineering,. ANSI/IES RP-16-1986, Illuminating Engineering Society, 345 East 47th Street, New York, NY 10017, June 1986.

6. Henrik Wann Jensen. Global illumination using photon maps. In *Seventh Eurographics Workshop on Rendering*, pages 22–30, Porto, Portugal, June 1996.

7. James T. Kajiya. The rendering equation. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.

8. Eric P. Lafortune and Yves D. Willems. A 5D tree to reduce the variance of monte carlo ray tracing. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

9. Robert R. Lewis. Wavelet radiative transfer and surface interaction. In *Proceedings of the Seventh Western Computer Graphics Symposium*, pages 73–83, 1996.

10. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992. ISBN 0-521-43108-5.

```
function Integrate(WaveletNode wn, κ_i, λ_i, κ_r^min, κ_r^max, λ_r^min, λ_r^max )
r = 0
for b = 0..15
   if bit b of wn.ValueMask is set
      r += Γ^0_{wn.l,wn.m_{κ_i},b}(κ_i) × Γ^1_{wn.l,wn.m_{λ_i},b}(λ_i)
         × ∫_{κ_r^min}^{κ_r^max} Γ^2_{wn.l,wn.m_{κ_r},b}(κ_r)dκ_r × ∫_{λ_r^min}^{λ_r^max} Γ^3_{wn.l,wn.m_{λ_r},b}(λ_r)dλ_r
for b = 0..15
   if bit b of wn.ChildMask is set
      if κ_i is in the support of Γ^0_{wn.l+1,2wn.m[0]+b:0,b}(·)
         and λ_i is in the support of Γ^1_{wn.l+1,2wn.m[1]+b:1,b}(·)
         r += Integrate(wn.Child[b], κ_i, λ_i, κ_r^min, κ_r^max, λ_r^min, λ_r^max)
return r
```

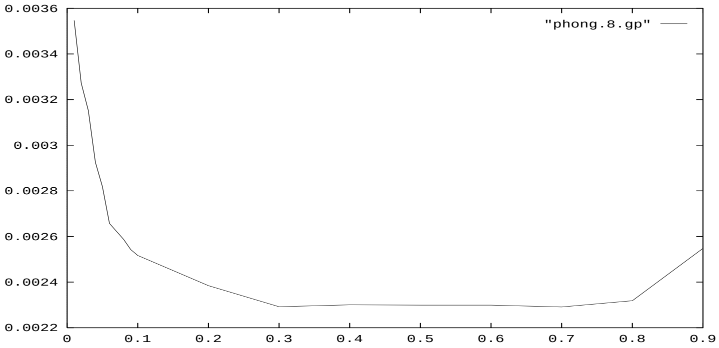
Figure 4: *Integration over exiting directions of the BRDF from a 4-variable Wavelet compressed BRDF*

11. Peter Shirley, Chang Yaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996. ISSN 0730-0301.

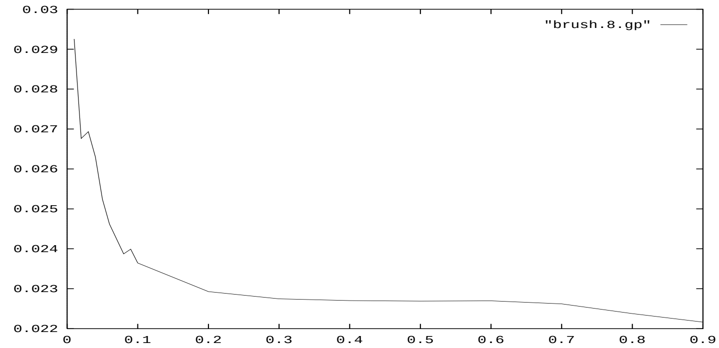
12. Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer. *IEEE Computer Graphics and Applications*, 15(3), 1995.

13. Gregory J. Ward. Measuring and modeling anisotropic reflection. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 265–272, July 1992.

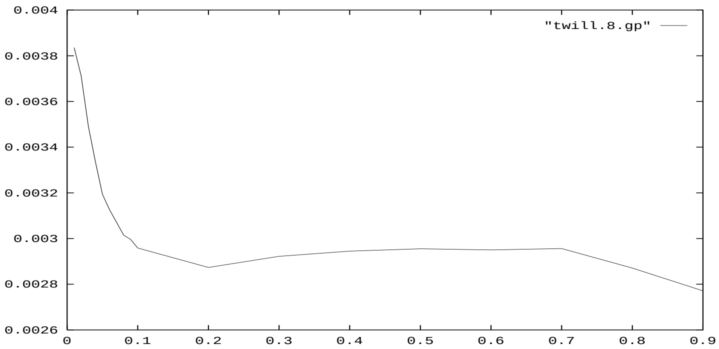
14. Gregory J. Ward. Towards more practical reflectance measurements and models. In *Graphics Interface '92 Workshop on Local Illumination*, pages 15–21, May 1992.



(a) A Phong shader with an exponent of 30



(b) A brushed metal



(c) A twill weave

Figure 5: *RMS Error induced by thresh-holding. Thresholding expressed as fraction of coefficients retained. Haar Basis.*

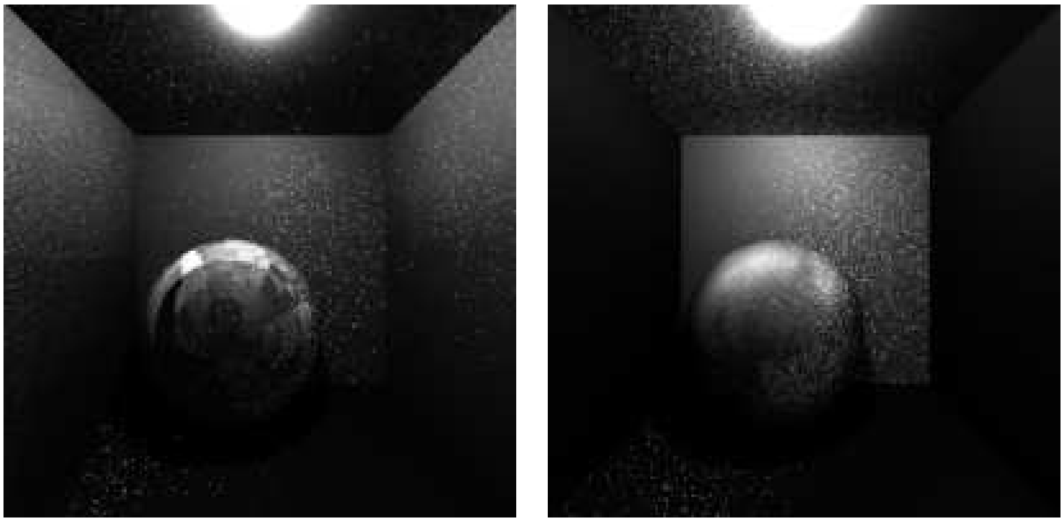


Figure 6: *Ray-traced environment with reflected directions computed using our method*