Interactive Rendering with Arbitrary BRDFs using Separable Approximations

Jan Kautz and Michael D. McCool
Computer Graphics Laboratory; Department of Computer Science; University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{jnkautz,mmccool}@cgl.uwaterloo.ca
http://www.cgl.uwaterloo.ca

Abstract. *A separable decomposition of bidirectional reflectance distributions (BRDFs) is used to implement arbitrary reflectances from point sources on existing graphics hardware. Two-dimensional texture mapping and compositing operations are used to reconstruct samples of the BRDF at every pixel at interactive rates.*
A change of variables, the Gram-Schmidt halfangle/difference vector parameterization, improves separability. Two decomposition algorithms are also presented. The singular value decomposition (SVD) minimizes RMS error. The normalized decomposition is fast and simple, using no more space than what is required for the final representation.

1 Introduction

Traditionally hardware renderers only support the Phong lighting model [19] in combination with Gouraud shading. However, the Phong lighting model is strictly empirical and physically implausible. Gouraud shading also tends to undersample the highlight unless a highly tessellated surface is used.

In general, surface reflectance can be described using a bidirectional reflectance distribution, or BRDF. The reflectance equation describes the outgoing radiance $L_o(\hat{\omega}_o, \underline{\mathbf{x}})$ in direction $\hat{\omega}_o$ at a surface point $\underline{\mathbf{x}}$ as an integral over the irradiance $L_i(\hat{\omega}_i, \underline{\mathbf{x}}) \cos_+ \theta_i$ at that surface point weighted by the BRDF $f(\hat{\omega}_o, \underline{\mathbf{x}}, \hat{\omega}_i)$:

$$L_o(\hat{\omega}_o, \underline{\mathbf{x}}) \quad = \quad \int_\Omega f(\hat{\omega}_o, \underline{\mathbf{x}}, \hat{\omega}_i) L_i(\hat{\omega}_i, \underline{\mathbf{x}}) \cos_+ \theta_i \, d\hat{\omega}_i,$$

with $\cos_+ \theta_i = \max(\hat{\omega}_i \cdot \hat{\mathbf{n}}, 0)$ where $\hat{\mathbf{n}}$ is the surface normal at $\underline{\mathbf{x}}$ and Ω is the hemisphere of incoming directions. For M point sources, the reflectance integral reduces to

$$L_o(\hat{\omega}_o, \underline{\mathbf{x}}) \quad = \quad \sum_{i=1}^{M} f(\hat{\omega}_o, \underline{\mathbf{x}}, \hat{\omega}_i) \cos_+ \theta_i \frac{I_i}{r_i^2} \tag{1}$$

where r_i is the distance to light source i and I_i is its intensity. While point sources are not ideal (glossy surfaces don't really look glossy unless they reflect the environment), we would still at *least* like to render surfaces at interactive rates using this model, with arbitrary BRDFs, evaluated at per-pixel resolution.

Neglecting $\underline{\mathbf{x}}$ (and wavelength), BRDFs are parameterized by at least four degrees of freedom. Analytic evaluation of a BRDF is possible using programmable shaders

[18], but such capabilities are not yet widely available in hardware and measured data still cannot be used directly. Tabulated BRDFs could be implemented in hardware using four-dimensional texture maps, but this would be expensive: 50MB would be required for (the relatively low) resolution of 64^4 at three bytes per sample. To render multiple surfaces with different reflectance functions at interactive rates, a compressed representation that uses existing two-dimensional hardware texturing capabilities is desirable.

Recently separable decompositions have been proposed for compressing BRDFs [4, 5, 22]. Separable decompositions approximate (to arbitrary accuracy) a high-dimensional function f using a sum of products of lower-dimensional functions g_k and h_k:

$$f(x, y, z, w) \quad \approx \quad \sum_{k=1}^{N} g_k(x, y) h_k(z, w). \tag{2}$$

Separable decompositions are capable of high compression rates if good approximations can be found for small N. Separable representations are also much easier to evaluate pointwise than other representations, such as k-nearest neighbors [6], wavelets, or spherical harmonics [26].

As we will demonstrate, under certain changes of variables many BRDFs are highly separable, and so a small number N of low-dimensional functions can be used to represent them accurately. In fact, $N = 1$ has proven to be visually adequate for many interesting BRDFs; see colour Figure 12. Because of the simplicity of the reconstruction process, we can perform it at interactive rates using existing hardware support for texturing, compositing, and diffuse lighting. Of course, this method is also applicable to software rendering; fast evaluation of BRDFs can be performed using only a few texture lookups, multiplications and additions.

2 Overview

The technique described in this paper is composed of two distinct phases.

In the first phase a target BRDF is analyzed and a suitable separable representation found. Algorithms to accomplish this are discussed in Section 4.

Once a representation is found, it can be used in image synthesis. In this paper we focus on *interactive* rendering, on existing hardware, with respect to the point-source lighting model (eq. 1), although the separable representation of BRDFs is also very useful in software rendering.

The basic algorithm for hardware rendering replaces the evaluation of the BRDF by the sum of products of lower dimensional functions (substituting eq. 2 into eq. 1). The functions g_k and h_k are held in texture maps, the multiplications are done using either compositing or multitexturing, the cosine term is evaluated using diffuse lighting and texture modulation, and the summations (if $N > 1$ or more than one light source is required) are done with an accumulation buffer or compositing.

Hardware-accelerated interactive rendering imposes a number of constraints. The most serious is that the parameterization of the texture maps must be consistent with linear interpolation of texture coordinates so Phong shading (per-pixel tangent and normal interpolation) can be accurately approximated. These constraints interact with changes of variables that are useful for increasing the separability of BRDFs. Suitable parameterization choices and their effects on quality are discussed in Section 5. In Section 6 we describe briefly how the capabilities of existing graphics hardware can be exploited to achieve interactive rendering performance.

3 Prior Work

Representations of reflectance functions fall into two categories:

1. Parameterized models for specific kinds of BRDFs.
2. General approximation techniques.

The most familiar specialized parametric representation is probably the Phong model [19], which was one of the first reflectance models. Ward [25] has presented a more sophisticated model based on anisotropic Gaussian lobes fitted to various BRDFs. He *et al.* [7] have derived a physically based model based on Kirchhoff diffraction (called the HTSG model here), which also takes wavelength into account. Poulin and Fournier [20] have proposed a model based on self shadowing of microcylinders. There are other models, but we use these for examples in this paper.

There are also many BRDF approximation techniques. Schröder and Sweldens [23] have represented BRDFs using spherical wavelets. Koenderink *et al.* [12] have expressed BRDFs in terms of an orthonormal basis using Zernike polynomials. Lafortune *et al.* [13] have used an approximation based on the summation of generalized Phong cosine lobes. Cabral *et al.* [3] were the first to use spherical harmonics to represent BRDFs. Fournier [5] used a sum of separable functions for representing reflectance models.

None of the more general approximation models have been used in interactive rendering. The problem with many of the above representations is that interactive hardware implementations would require completely new hardware—they do not build on existing capabilities. The exception to this is Fournier's separable representation, which can be implemented using existing support for texture mapping and compositing.

To our knowledge there has only been one attempt to incorporate more sophisticated reflectance models into interactive rendering without general shader support. Heidrich and Seidel [8] analytically separated the Banks anisotropic model [2] and have proposed a single pass rendering algorithm using texture mapping. Our approach is similar but we consider the more general case of arbitrary BRDFs.

4 Decomposition

The first phase in the application of this technique is generation of a separable decomposition of each BRDF. Decomposition is done in advance of rendering, and need only be performed once per BRDF. The result is a compressed representation that can be stored until needed.

Neglecting position and wavelength dependence, a general anisotropic BDRF f is a function of four degrees of freedom corresponding to incident direction $\hat{\omega}_i$ and view direction $\hat{\omega}_o$. In Section 5 we will look at several reparameterizations that can increase the effective separability of a BRDF, so assume that the parameters of the BRDF are $\mathbf{x} = \mathbf{P}_x(\hat{\omega}_i, \hat{\omega}_o)$ and $\mathbf{y} = \mathbf{P}_y(\hat{\omega}_i, \hat{\omega}_o)$ for vector functions $\mathbf{P}_x$ and $\mathbf{P}_y$.

A separable decomposition approximates a multivariate function f as a sum of products of functions g_k and h_k of lower dimensionality:

$$\begin{aligned} f(\hat{\omega}_i, \hat{\omega}_o) &= f_P(\mathbf{P}_x(\hat{\omega}_i, \hat{\omega}_o), \mathbf{P}_y(\hat{\omega}_i, \hat{\omega}_o)); \\ f_P(\mathbf{x}, \mathbf{y}) &\approx \sum_{k=1}^{N} g_k(\mathbf{x}) h_k(\mathbf{y}). \end{aligned} \tag{3}$$

Our method does *not* assume that a BRDF is single-term separable. A BRDF can *always* be represented accurately using a separable expansion, if enough terms are used. However, we have found that a good approximation can be achieved with only a few terms and often a single term *is* sufficient if a good parameterization can be found.

We will consider two algorithms for finding appropriate functions g_k and h_k: singular value decomposition and normalized decomposition. The singular value decomposition (SVD) can produce optimal approximations, but is relatively expensive in time and space. We have developed an approach called normalized decomposition (ND), which is a much simpler technique that can produce good decompositions in time linear in the number of BRDF samples taken. It uses no more space than required to store the output factors.

4.1 Singular Value Decomposition

Given a matrix M, the singular value decomposition (SVD) [1, 21] of M is the factorization $\mathsf{M} = \mathsf{USV}^T$ where the columns of $\mathsf{U} = [\mathbf{u}_k]$ and $\mathsf{V} = [\mathbf{v}_k]$ are orthonormal and $\mathsf{S} = \text{diag}\,(\sigma_k)$ is a diagonal matrix of singular values σ_k. The matrix product USV^T can be written as a sum:

$$\mathsf{M} \quad = \quad \sum_{k=1}^{K} \sigma_k \mathbf{u}_k \mathbf{v}_k^T .$$

Note that each term $\mathbf{u}_k \mathbf{v}_k^T$ is an *outer product*, i.e. a matrix whose elements are products of an element of $\mathbf{u}_k$ and an element of $\mathbf{v}_k$.

The singular values σ_k are positive and monotonically decreasing in magnitude. Truncating the above sum results in an optimal root mean square approximation of M.

Assume we have a tabulated, reparameterized BRDF $f_P(\mathbf{x}, \mathbf{y})$ that has been sampled at some collection of $K \times K$ parameter values. Define a matrix $\mathsf{M} = [m_{ij}]$ with $m_{ij} = f_P(\mathbf{x}_i, \mathbf{y}_j)$; in other words, let $\mathbf{x}$ be constant for each row of the matrix, and let $\mathbf{y}$ be constant for each column:

$$\mathsf{M} \quad = \quad \begin{pmatrix} f_P(\mathbf{x}_1, \mathbf{y}_1) & \cdots & f_P(\mathbf{x}_1, \mathbf{y}_K) \\ \vdots & \ddots & \vdots \\ f_P(\mathbf{x}_K, \mathbf{y}_1) & \cdots & f_P(\mathbf{x}_K, \mathbf{y}_K) \end{pmatrix}$$

If we interpolate $\mathbf{u}_k$ and $\mathbf{v}_k$ of the SVD factorization of this matrix into the two-dimensional functions $u_k(\mathbf{x})$ and $v_k(\mathbf{y})$ and then truncate the series at $N < K$ terms, we have the approximation

$$f_P(\mathbf{x}, \mathbf{y}) \quad \approx \quad \sum_{k=1}^{N} \sigma_k u_k(\mathbf{x}) v_k(\mathbf{y}).$$

There are several major drawbacks to the SVD. First, it always results in a least root mean square approximation of M; it is not possible to specify a fundamentally different norm.

Secondly, the expansion contains negative factors that are not compatible with the strictly positive nature of reflectance, nor most graphics hardware. For the first term these negative values can usually be cancelled out, but not for later terms.

Finally, the memory consumption of the matrix M grows rapidly with the desired resolution. If we sample the BRDF 64 times along each parameter, the resulting matrix

consumes about 67MB, assuming 4 bytes (a float) per sample. If we sample the BRDF 128 times in each dimension, the matrix would take up about 1GB. The resulting texture maps would be only 64×64 or 128×128 pixels in size. Smaller texture maps than this may result in visual artifacts.

4.2 Normalized Decomposition

The SVD is too expensive for high-resolution factorizations and it always computes a full approximation, which is often not necessary, as a few terms usually suffice for a good approximation. The Normalized Decomposition (ND) algorithm can be used instead in many situations.

Consider first a single-term approximation

$$f_P(\mathbf{x}, \mathbf{y}) \approx \tilde{f}_{P1}(\mathbf{x}, \mathbf{y}) = g_1(\mathbf{x}) h_1(\mathbf{y}).$$

If $\mathbf{x}$ is fixed, $g_1(\mathbf{x})$ is a constant, scaling a profile given by $h_1(\mathbf{y})$. To find a single-term separable approximation, find the average normalized profile along $\mathbf{y}$ and store it in $h_1(\mathbf{y})$, then store the normalization factors in $g_1(\mathbf{x})$.

Although the ND method does not guarantee optimality, tests show that single-term approximations using the SVD are in most cases visually similar to single-term approximations found using the ND.

A wide class of approximations can be computed using the p-norm:

$$\begin{aligned} g_1(\mathbf{x}) &= \left(\int_Y |f_P|^p (\mathbf{x}, \mathbf{y}) \, d\mathbf{y} \right)^{\frac{1}{p}}, \\ h_1(\mathbf{y}) &= \frac{1}{|X|} \int_X \frac{f_P(\mathbf{x}, \mathbf{y})}{g_1(\mathbf{x})} \, d\mathbf{x}. \end{aligned}$$

To implement the ND algorithm these integrals must be computed numerically.

Normalized decomposition (ND), besides being considerably faster than an SVD, takes much less memory. We can sample the BRDF and compute the average profile and norms incrementally, rather than having to store and operate on a large matrix. The only memory needed is that for the two output functions $g_1(\mathbf{x})$ and $h_1(\mathbf{y})$. The averaging process used to compute the output functions also reduces noise, so the technique can be used with good results on noisy measured data.

A single-term ND expansion contains only positive factors, since the BRDF f must be positive everywhere. For multi-term expansions, approximation of sequential residuals can be used. However, since the residuals will contain negative values, the additional terms will contain factors with negative values.

5 Parameterization

The parameterization of the BRDF can significantly affect separability. Figure 1 visualizes the effect of reparameterization. Two images are decomposed and reconstructed by the ND algorithm. The original image is not reconstructed very well. The reparameterized image, which is rotated in order to align the object with the boundaries, is much better approximated.

As we focus on hardware rendering the parameterization of the functions of the resulting separable form must also be compatible with the linear interpolation of texture map coordinates performed in hardware. If this technique is used with per-pixel evaluation of texture coordinates any bijective parametrization of BRDFs can be used.

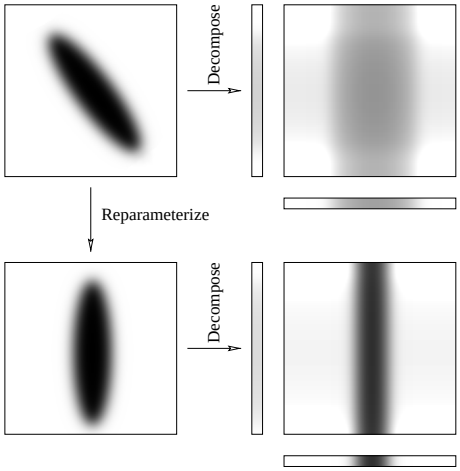


Fig. 1. *Reparameterization can improve the performance of decomposition. In this case the SVD would result in a perfect single-term reconstruction. The ND algorithm used in this figure results in a blurrier reconstruction.*

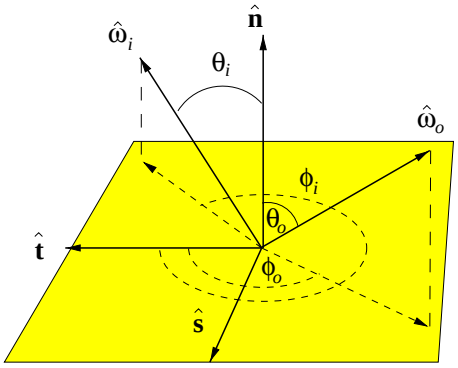
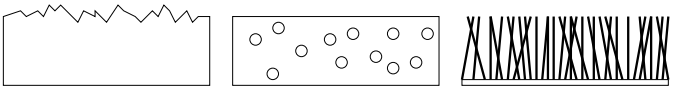


Fig. 2. *Surface coordinate system used to parameterize a BRDF. The surface normal is* $\hat{\mathbf{n}}$*, the primary surface tangent is* $\hat{\mathbf{t}}$*, and* $\hat{\mathbf{s}}$ *is the secondary surface tangent perpendicular to* $\hat{\mathbf{n}}$ *and* $\hat{\mathbf{t}}$*.*

We have not found a single parameterization that works well for all BRDFs, although we have found parameterizations that work well for broad categories of BRDFs. This is to be expected due to the different surface phenomena that contribute to variation in reflectance, as shown in Figure 3. Each of these phenomenon aligns the features the BRDF along different axes. We will show examples of these tradeoffs in Figure 5.



A: Microfacets	B: Subsurface Scattering	C: Deep Microgeometry
Example: rough metal	Example: human skin	Example: velvet

Fig. 3. *Surface phenomena that contribute to BRDFs.*

5.1 BRDF Parameterization

The standard parameterization of a BRDF is with respect to the incident direction $\hat{\omega}_i$ and viewing direction $\hat{\omega}_o$ relative to a local surface frame at $\underline{\mathbf{x}}$. A unit length vector $\hat{\mathbf{a}}$ can be expressed in spherical coordinates $(\theta(\hat{\mathbf{a}}), \phi(\hat{\mathbf{a}}))$ relative to the local surface frame $\{\hat{\mathbf{n}}, \hat{\mathbf{t}}, \hat{\mathbf{s}}\}$ (see Figure 2) as follows:

$$\begin{aligned} \cos\theta(\hat{\mathbf{a}}) &= \hat{\mathbf{n}} \cdot \hat{\mathbf{a}}, \\ \tan\phi(\hat{\mathbf{a}}) &= (\hat{\mathbf{s}} \cdot \hat{\mathbf{a}})/(\hat{\mathbf{t}} \cdot \hat{\mathbf{a}}). \end{aligned}$$

Let $\theta_i = \theta(\hat{\omega}_i)$, $\phi_i = \phi(\hat{\omega}_i)$, $\theta_o = \theta(\hat{\omega}_o)$ and $\phi_o = \phi(\hat{\omega}_o)$.

Many BRDFs are not especially separable with respect to the standard incident/view, or $(\theta_i, \phi_i) \times (\theta_o, \phi_o)$ parameterization.

Very good results can usually be obtained using an elevation/azimuth, or $(\theta_i, \theta_o) \times (\phi_i, \phi_o)$ parameterization, which is unfortunately not directly compatible with hardware texture mapping; see Section 5.3.

Rusinkiewicz [22] has parameterized the BRDF in terms of the halfway vector $\hat{\mathbf{h}}$ (the vector halfway between the incident and outgoing ray) and a "difference vector" $\hat{\mathbf{d}}$:

$$\begin{aligned} \hat{\mathbf{h}} &= \text{norm}(\hat{\omega}_i + \hat{\omega}_o), \\ \hat{\mathbf{d}} &= (\text{Rot}\{\hat{\mathbf{s}}, -\theta(\hat{\mathbf{h}})\} \circ \text{Rot}\{\hat{\mathbf{n}}, -\phi(\hat{\mathbf{h}})\})\, \hat{\omega}_i. \end{aligned}$$

The two rotations are chosen to rotate $\hat{\mathbf{h}}$ to the pole of a new coordinate system, which is visualized in Figure 4; $\hat{\mathbf{d}}$ is in fact $\hat{\omega}_i$, but parameterized with respect to this transformed frame. The Rusinkiewicz reparameterization can be interpreted as a change of basis with the new basis found by a Gram-Schmidt orthonormalization of $\{\hat{\mathbf{h}}, -\hat{\mathbf{n}}, \hat{\mathbf{n}} \times \hat{\mathbf{h}}\}$:

$$\begin{aligned} \hat{\mathbf{h}} &= \text{norm}(\hat{\omega}_i + \hat{\omega}_o), & \hat{\mathbf{u}} &= -\text{norm}(\hat{\mathbf{n}} - (\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})\hat{\mathbf{h}}), \\ \hat{\mathbf{v}} &= \hat{\mathbf{h}} \times \hat{\mathbf{u}}, & \hat{\mathbf{d}} &= [\hat{\omega}_i \cdot \hat{\mathbf{h}}, \hat{\omega}_i \cdot \hat{\mathbf{u}}, \hat{\omega}_i \cdot \hat{\mathbf{v}}]^T. \end{aligned}$$

Although the Rusinkiewicz parameterization makes many BRDFs much more separable, it is numerically unstable if $\hat{\mathbf{h}} \approx \hat{\mathbf{n}}$. This makes it unsuitable for hardware interpolation; see Section 5.3.

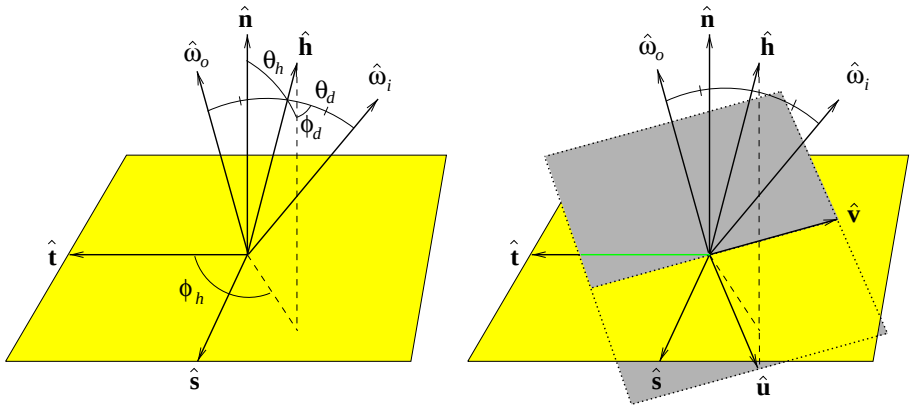


Fig. 4. *Two different views of the Rusinkiewicz parameterization. View 1 (left): The angles (θ_d, ϕ_d) of the "difference" vector $\hat{\mathbf{d}}$ are relative to $\hat{\mathbf{h}}$ and the plane containing $\hat{\mathbf{h}}$ and $\hat{\mathbf{n}}$. View 2 (right): The Rusinkiewicz parameterization generates a new orthonormal frame consisting of $\hat{\mathbf{h}}$, a vector $\hat{\mathbf{u}}$ perpendicular to $\hat{\mathbf{h}}$ and in the same plane as both $\hat{\mathbf{h}}$ and $\hat{\mathbf{n}}$, and a third vector $\hat{\mathbf{v}}$ perpendicular to both $\hat{\mathbf{h}}$ and $\hat{\mathbf{u}}$. The vector $\hat{\omega}_i$ is analyzed against this new frame to obtain the difference vector coordinates.*

To avoid the numerical instability, we can instead apply Gram-Schmidt orthonormalization to $\{\hat{\mathbf{h}}, \hat{\mathbf{t}}, \hat{\mathbf{s}}\}$:

$$\begin{aligned} \hat{\mathbf{h}} &= \text{norm}(\hat{\omega}_i + \hat{\omega}_o), & \hat{\mathbf{t}}' &= \text{norm}(\hat{\mathbf{t}} - (\hat{\mathbf{t}} \cdot \hat{\mathbf{h}})\hat{\mathbf{h}}), \\ \hat{\mathbf{s}}' &= \hat{\mathbf{h}} \times \hat{\mathbf{t}}', & \hat{\mathbf{d}} &= [\hat{\omega}_i \cdot \hat{\mathbf{h}}, \hat{\omega}_i \cdot \hat{\mathbf{s}}', \hat{\omega}_i \cdot \hat{\mathbf{t}}']^T. \end{aligned} \tag{4}$$

Note that $\hat{\mathbf{h}}$ cannot equal $\hat{\mathbf{t}}$ for any true surface frame. For vertex frames that are not exactly aligned with the surface, as used in computer graphics, it is theoretically possible for $\hat{\mathbf{h}}$ to be close to $\hat{\mathbf{t}}$ but only for glancing retroreflection (for which the majority of BRDFs are 0), and even then for only one view direction.

This parameterization may not align anisotropic features of the BRDF and does not have the same symmetries as the Rusinkiewicz parameterization, but in practical applications it is numerically stable and it is compatible with hardware interpolation.

5.2 Error Analysis

To compare these parameterizations, we tested a number of analytic and measured BRDFs [10]. Each BRDF was sampled 32 times along each dimension and a separable decomposition was computed using the SVD. To obtain a consistent parameterization-independent comparison the RMS error of the outgoing radiance was estimated using 8000 Monte Carlo samples evenly distributed over both the incident and view hemispheres. Each sample was computed by multiplying the value of the approximated BRDF with the cosine of the incident elevation angle (i.e. θ_i). This choice was made because outgoing radiance corresponds to what is actually perceived by the eye.

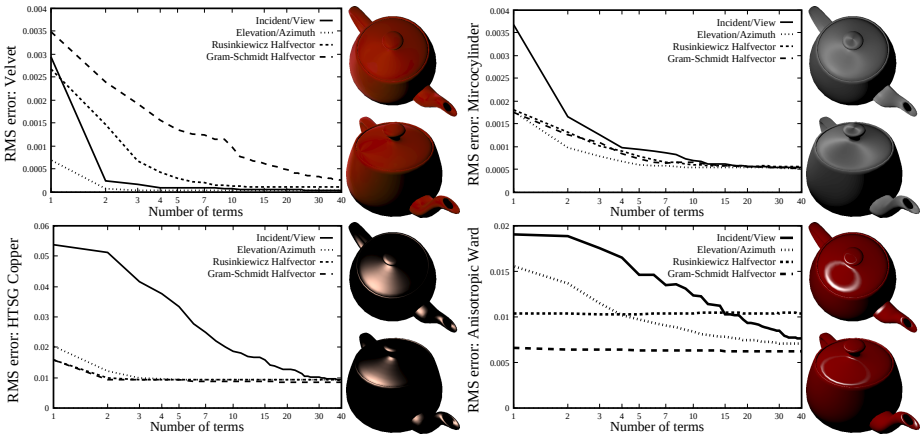


Fig. 5. *Cosine weighted RMS luminance error for all parameterizations and BRDFs for the velvet, HTSG copper, Poulin-Fournier microcylinder brushed metal model and Ward's model, as a function of the number of terms in an SVD. The non-zero asymptotic error is due to sampling and the bilinear interpolation used to reconstruct a continuous version of each factor.*

Results are shown in Figure 5. The non-zero asymptotic error is due to the bilinear interpolation used to reconstruct a continuous version of each factor. The asymptotic errors shown in Figure 5 are also highly dependent on the size of the texture maps used and the average value of each BRDF. Figure 6 shows the dependency between texture map size and RMS error. Note that the asymptotic error may not decrease monotonically, depending on how the BRDF interacts with the sampling grid, and how antialiasing is performed during BRDF sampling.

RMS error is not a totally appropriate error metric, as it tends to overweight the peaks while underweighting the diffuse colour. Future research should consider appropriate error metrics for BRDF approximations, perhaps by analyzing shape-from-

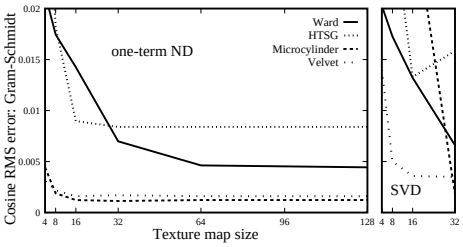


Fig. 6. *Cosine weighted RMS error with respect to the original BRDF for single-term ND and single-term SVD approximations with Gram-Schmidt parameterization using varying texture map sizes.*

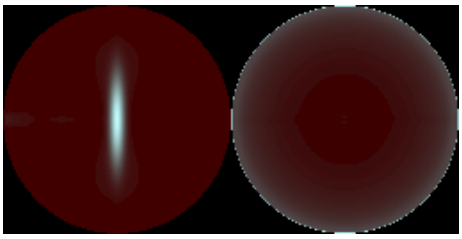


Fig. 7. *The resulting texture maps (as XY hemisphere maps) after decomposing Ward's model in the Gram-Schmidt parameterization with the ND algorithm. See Figure 10 for an image rendered with these texture maps.*

shading algorithms in the context of the nonlinear intensity sensitivity of the human visual system.

With respect to RMS error, we found that no single parameterization was the best for all the BRDFs we tested. BRDFs that are dominated by specular or near-specular reflection off randomly oriented microfacets (Figure 3A), such as HTSG models of rough metal or Ward's model, were usually best approximated by halfvector parameterizations. Matte materials in which "thick" microgeometry and self-shadowing were present and that had colour shifts at normal viewing angles, such as matte translucent paints or velvet (Figure 3B&C), had BRDFs that usually were most separable using the standard incident/view parameterization. Certain BRDFs, such as glossy iridescent paints, may be best approximated using a sum of two terms with different parameterizations. The elevation/azimuth parameterization was surprisingly effective in most cases.

5.3 Factor Parameterization

Now we have to find a suitable representation of the functions in a separable expansion that allows for a good interpolation of the parameters. Several possible representations [9, 14, 17] permit bilinear interpolation in the (u, v)-space of texture maps to approximate spherical interpolation of unit vectors.

Mapping a spherical coordinate (θ, ϕ) representation directly onto the coordinates of a texture map does not give the desired result, since interpolation does not correctly wrap around in ϕ using the shortest path over the hemisphere.

Hemisphere maps are a better solution to angular interpolation. The XY hemisphere map just projects a given unit vector $\hat{\mathbf{a}}$ onto the tangent vectors:

$$a_x = \hat{\mathbf{a}} \cdot \hat{\mathbf{t}}, \qquad a_y = \hat{\mathbf{a}} \cdot \hat{\mathbf{s}}.$$

Now we have to map a_x and a_y into the usual range of texture coordinates:

$$\mathfrak{a}_u = \frac{1}{2}(a_x + 1), \qquad \mathfrak{a}_v = \frac{1}{2}(a_y + 1). \tag{5}$$

If problems with backfacing vertex frames are encountered it is better to use parabolic maps [9], which automatically take care of negative $a_z = \hat{\mathbf{a}} \cdot \hat{\mathbf{n}}$ and scale the other

coordinates so they lie outside the unit circle and extend out to infinity smoothly:

$$\mathfrak{a}_u = \frac{1}{2}\left(\frac{a_x}{1+a_z}+1\right), \qquad \mathfrak{a}_v = \frac{1}{2}\left(\frac{a_y}{1+a_z}+1\right).$$

This computation can be accomplished with a projective transformation and normalization (required anyways for correct texture coordinate interpolation) so the net cost is just another dot product to compute a_z.

The Rusinkiewicz parameterization does not work well with hemisphere map representations. Our approach calculates texture coordinates only at vertices; linear interpolation of texture coordinates is used within polygons. The difference vectors at the vertices of polygons covered by a highlight tend to be parameterized with wildly different values (i.e., vary heavily in ϕ_d) in the Rusinkiewicz parameterization. Bilinear interpolation then calculates incorrect texture coordinates, which introduces very visible errors.

The elevation/azimuth $(\theta_i, \theta_o) \times (\phi_i, \phi_o)$ parameterization is also not compatible with hemisphere maps. When interpolating azimuth angles the shortest direction of interpolation needs to be chosen, taking into account the periodicity of both ϕ_i and ϕ_o. Neither hemisphere maps nor periodic textures satisfy this requirement.

The incident/view $(\theta_i, \phi_i) \times (\theta_o, \phi_o)$ parameterization works with hemisphere maps, and certain "deep microgeometry" BRDFs (such as velvet) are more separable with respect to this parameterization.

The new Gram-Schmidt halfvector parameterization combined with an XY hemisphere map representation of the functions g_k and h_k has proven to be a good combination for many near-specular BRDFs and is given in full by the following:

$$\begin{aligned}
\hat{\mathbf{h}} &= \mathrm{norm}(\hat{\omega}_i + \hat{\omega}_o), & \hat{\mathbf{t}}' &= \mathrm{norm}(\hat{\mathbf{t}} - (\hat{\mathbf{h}}\cdot\hat{\mathbf{t}})\hat{\mathbf{h}}), \\
& & \hat{\mathbf{s}}' &= \hat{\mathbf{h}} \times \hat{\mathbf{t}}', \\
\mathfrak{h}_u &= (\hat{\mathbf{h}}\cdot\hat{\mathbf{t}}+1)/2, & \mathfrak{d}_u &= (\hat{\omega}_i\cdot\hat{\mathbf{t}}'+1)/2, \\
\mathfrak{h}_v &= (\hat{\mathbf{h}}\cdot\hat{\mathbf{s}}+1)/2, & \mathfrak{d}_v &= (\hat{\omega}_i\cdot\hat{\mathbf{s}}'+1)/2,
\end{aligned} \tag{6}$$

The bottom left image of Figure 10 shows Newell's teapot rendered with a single-term decomposition of Ward's model using the ND algorithm and the above parameterization. Figure 7 shows the corresponding texture maps that were used to render the single-term approximation.

6 Rendering

Compositing and texturing operations, which are already available in current graphics hardware through OpenGL or Direct3D, can be used for interactive reconstruction of a separably decomposed BRDF. Frame buffer and texture map arithmetic is used to reconstruct the outgoing radiance (eq. 1) in parallel for each pixel.

For each term, first render the scene with one diffusely illuminated texture map factor. Without clearing the colour or depth buffers, render the scene again with constant illumination and the second texture map. Texture coordinates have to be recalculated at each vertex whenever the view or relative light source position changes. For the second pass, set the depth test to "equality" and set the compositing operation to "multiply". The accumulation buffer can then be used to sum multiple terms. With multitexturing,

1. Calculate texture coordinates $\mathfrak{h}$ and $\mathfrak{d}$ for each vertex (see Equation 6).
2. Clear the colour and depth buffers.
3. Set up a simple Lambertian lighting model.
4. Render the scene using $g_k(\mathfrak{h})$ as a texture map on the diffuse reflectance. The result in the colour buffer is $g_k(\mathfrak{h}) I \cos_+ \theta_i / r^2$.
5. Set the z-test to ''equality''.
6. Set the compositing operator to *multiply* colours.
7. Turn off hardware shading.
8. Render the scene using $h_k(\mathfrak{d})$ as a texture map. The result in the colour buffer is $g_k(\mathfrak{h}) h_k(\mathfrak{d}) I \cos_+ \theta_i / r^2$.

Fig. 8. *Pseudocode for rendering using contemporary graphics hardware.*

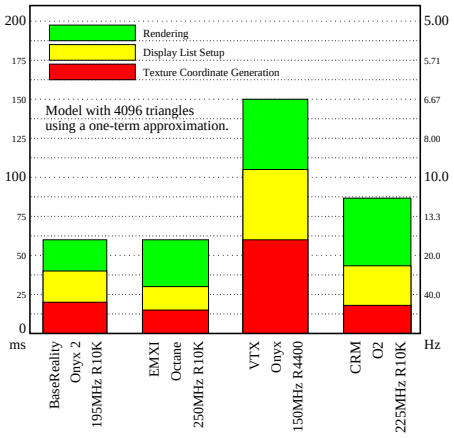


Fig. 9. *Rendering times and rates on different platforms using a one-term, two-factor approximation. A breakdown into stages is shown; the lower two stages (texture coordinate generation and display list regeneration) must currently be done by the host CPU.*

the product of two texture maps and the diffuse lighting can be formed in a single pass. See Figure 8 for pseudo code for this algorithm.

For a single-term, two-factor decomposition and a model with 2048 quads, we achieved rates of 6.7-17Hz; see Figure 9. Performance was strongly dependent on bus and CPU performance, as it was necessary to perform tangent vector transformation and texture coordinate generation on the host.

Texture mapping a diffuse term can reintroduce a dependence on $\underline{\mathbf{x}}$ that can add visual complexity; see Figure 13. As well, MIP-mapping can be used to avoid highlight aliasing, by generating a pyramid of factorizations at different resolution levels and prefiltering the BRDF for each level. Alternatively, at some loss in accuracy, the factors of a single high-resolution decomposition can be filtered and downsampled independently.

6.1 Hardware Rendering Issues

Our implementation needed to address the limitations of current graphics hardware, specifically lack of dynamic range and precision, limited texture coordinate generation capabilities, and lack of negative numbers and signed arithmetic. In this section we only sketch how to overcome these limitations; for more detail see [10]. Some of these limitations had a moderate impact on performance, but it would be relatively easy to address them in new hardware designs.

Cosine Term: We use hardware lighting to multiply the term $g_k h_k$ by $\cos_+ \theta_i$. It could be built into the reflectance function, but this might affect the separability.

Signed Arithmetic: Multiterm approximations will have negative values in the expansion, which are incompatible with contemporary graphics hardware. We can use biasing to convert the expansion to a form that uses only multiplication of positive values. Three additional rendering passes are then required to correct for

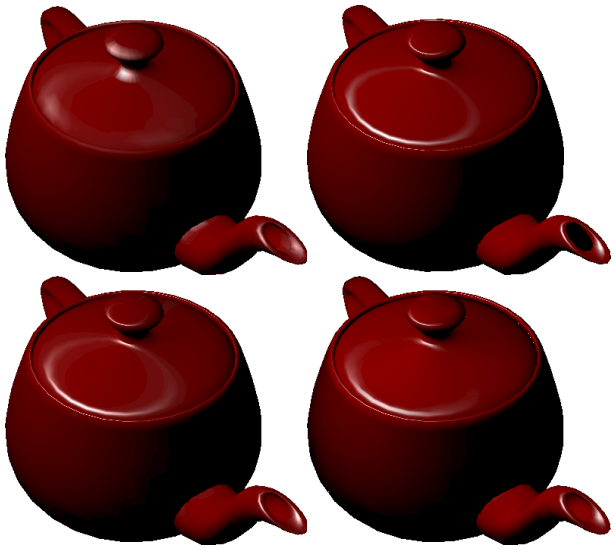


Fig. 10. *Comparison of rendering modes using Ward's anisotropic BRDF model [25], using* $k_d = (1.0, 0.0, 0.0)$*,* $(\alpha_x, \alpha_y) = (0.21, 0.048)$*, and* $k_s = 0.3$*. Top left: Gouraud interpolation of the BRDF evaluated at the vertices. Top right: the BRDF evaluated in a raytracer at every pixel, with interpolation of the frame vectors from the vertices. Bottom left: ND with one term (*128×128 *pixels), rendered with hardware acceleration. Bottom right: SVD with five terms (*32×32 *pixels), rendered with hardware acceleration.*

the bias [10]. Note that all passes use the same set of texture coordinates. See Figure 11 for an example of a multi-term rendering.

Precision: Frame buffers often only have 8 bits of precision. While this is sufficient for some BRDFs, deeper frame buffers increase quality. If multitexturing is used, only the multitexturing unit needs to have high precision.

Dynamic Range: Frame buffers usually clamp intermediate colour components to the range $[0, 1]$; texture maps can likewise only store values in the range $[0, 1]$. Unfortunately, BRDFs are defined over the range $[0, \infty)$. To avoid clamping we multiply the textures by scale factors chosen to ensure that no possible pointwise product will exceed the limits of the frame buffer, while maximizing the available precision. In a final rendering step, the result must be scaled by a product of the reciprocals of these scale factors. To avoid precision issues in the frame buffer, we limit the range of the original BRDFs to $[0, 5]$.

Texture Map Resolution: A texture map resolution of 128×128 has proven to be adequate for all the BRDFs we tested. Usually 64×64 and sometimes even less is enough; see Figure 6.

While the algorithm runs now at interactive rates on current hardware (Figure 9), some relatively simple extensions to the graphics hardware (and/or the graphics API and drivers), most importantly some new texture coordinate generation modes, would permit much better performance and ease of use [16].

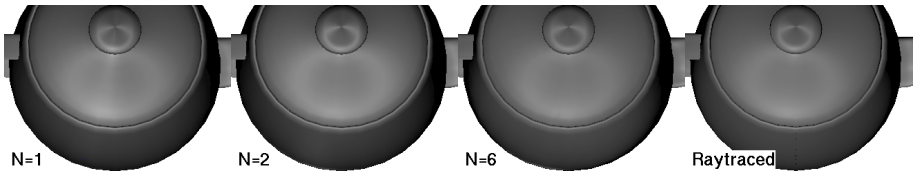


Fig. 11. *The microcylinder model rendered with different numbers of terms using OpenGL. The incorrect extra highlight that appears using one term disappears with two or more terms.*

6.2 Software Rendering

So far we have presented what has to be done to use a separable representation in hardware accelerated rendering. A separable representation can easily be used in software rendering: We do not have to even limit ourselves to parameterizations that are compatible with linear texture coordinate interpolation if texture coordinates are evaluated at each pixel.

To evaluate a BRDF at a certain position, one has to compute a sum of products at that position (eq. 3). As we have seen, a few terms are often sufficient to visually approximate a given BRDF. Thus the evaluation boils down to a few texture lookups, multiplications and additions. Implementation in a renderer or in a shading language is straightforward.

7 Conclusions

We have investigated different parameterizations and decomposition algorithms to improve the visual convergence of the separable approximation to BRDFs, and have shown how reconstruction from this compressed representation can be accomplished at interactive rates using current graphics hardware.

A modification of the Rusinkiewicz half-vector/difference vector parameterization [22] combined with a hemisphere map parameterization of the factors stored in texture maps gave excellent results for “microfacet” BRDFs.

The proposed representation of BRDFs can also be used for software renderers. Separable decompositions are a good choice for a compact general-purpose data format to store compressed BRDFs, as they allow for fast and accurate evaluation of BRDFs, including those reconstructed from measured data.

Acknowledgements

Holly Rushmeier and Greg Ward Larson were extremely helpful in retrieving data from Ward’s imaging gonioreflectometer, used here for the vinyl BRDF. The velvet and feather BRDFs were reconstructed from data provided by the group at Columbia-Utretch. The HTSG analytic model was implemented by Glenn Evans. The implementation of the Poulin-Fournier model was provided by Szymon Rusinkiewicz. Some additional BRDF data was provided by Heinz Mayer of the Graz University of Technology.

This research was funded by a grant from the National Sciences and Engineering Research Council of Canada.

References

1. H. Andrews and Hunt. *Digital Image Restoration*. Prentice-Hall, 1977.
2. D. Banks. Illumination in Diverse Codimensions. In *Proc. SIGGRAPH*, pages 327–334, July 1994.
3. B. Cabral, N. Max, and R. Springmeyer. Bidirectional Reflection Functions from Surface Bump Maps. In *Proc. SIGGRAPH*, pages 273–281, July 1987.
4. J. DeYoung and A. Fournier. Properties of Tabulated Bidirectional Reflectance Distribution Functions. In *Proc. Graphics Interface*, pages 47–55, May 1997.
5. A. Fournier. Separating Reflection Functions for Linear Radiosity. In *Eurographics Rendering Workshop*, pages 383–392, June 1995.
6. J. Gondek, G. Meyer, and J. Newman. Wavelength Dependent Reflectance Functions. In *Proc. SIGGRAPH*, pages 213–220, July 1994.
7. X. He, K. Torrance, F. Sillion, and D. Greenberg. A Comprehensive Physical Model for Light Reflection. In *Proc. SIGGRAPH*, pages 175–186, July 1991.
8. W. Heidrich and H.-P. Seidel. Efficient Rendering of Anisotropic Surfaces Using Computer Graphics Hardware. In *Image and Multi-dimensional DSP Workshop (IMDSP)*, 1998.
9. W. Heidrich and H.-P. Seidel. Realistic, Hardware-accelerated Shading and Lighting. In *Proc. SIGGRAPH*, August 1999.
10. J. Kautz. Hardware Rendering with Bidirectional Reflectances. Technical Report CS-99-02, University of Waterloo, 1999.
11. A. Keller. Instant Radiosity. In *Proc. SIGGRAPH*, pages 49–56, August 1997.
12. J. Koenderink, A. van Doorn, and M. Stavridi. Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes. In *European Conference on Computer Vision*, pages 28–39, 1996.
13. E. Lafortune, S.-C. Foo, K. Torrance, and D. Greenberg. Non-linear Approximation of Reflectance Functions. In *Proc. SIGGRAPH*, pages 117–126, August 1997.
14. E. Lafortune and Y. Willems. Using the Modified Phong Reflectance Model for Physically Based Rendering. Technical Report CW197, Dept. Comp. Sci., K.U. Leuven, 1994.
15. R. Lewis. Making Shaders More Physically Plausible. In *Eurographics Workshop on Rendering*, pages 47–62, June 1993.
16. M. McCool and W. Heidrich. Texture Shaders. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, August 1999. See also Technical Report CS-99-11, University of Waterloo.
17. L. Neumann and A. Neumann. Photosimulation: Interreflection with Arbitrary Reflectance Models and Illuminations. *Computer Graphics Forum*, 8(1):21–34, March 1989.
18. M. Olano and A. Lastra. A Shading Language on Graphics Hardware: The PixelFlow Shading System. In *Proc. SIGGRAPH*, pages 159–168, July 1998.
19. B.-T. Phong. Illumination for Computer Generated Pictures. *Comm. ACM*, 18(6):311–317, June 1975.
20. P. Poulin and A. Fournier. A Model for Anisotropic Reflection. In *Proc. SIGGRAPH*, pages 273–282, August 1990.
21. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, 1992.
22. S. Rusinkiewicz. A New Change of Variables for Efficient BRDF Representation. In *Eurographics Workshop on Rendering*, pages 11–23, June 1998.
23. P. Schröder and W. Sweldens. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Proc. SIGGRAPH*, pages 161–172, August 1995.
24. B. Walter, G. Alppay, E. Lafortune, S. Fernandez, and D. Greenberg. Fitting Virtual Lights for Non-Diffuse Walkthroughs. In *Proc. SIGGRAPH*, pages 45–48, August 1997.
25. G. Ward. Measuring and Modeling Anisotropic Reflection. In *Proc. SIGGRAPH*, pages 265–272, July 1992.
26. S. Westin, J. Arvo, and K. Torrance. Predicting Reflectance Functions from Complex Surfaces. In *Proc. SIGGRAPH*, pages 255–264, July 1992.

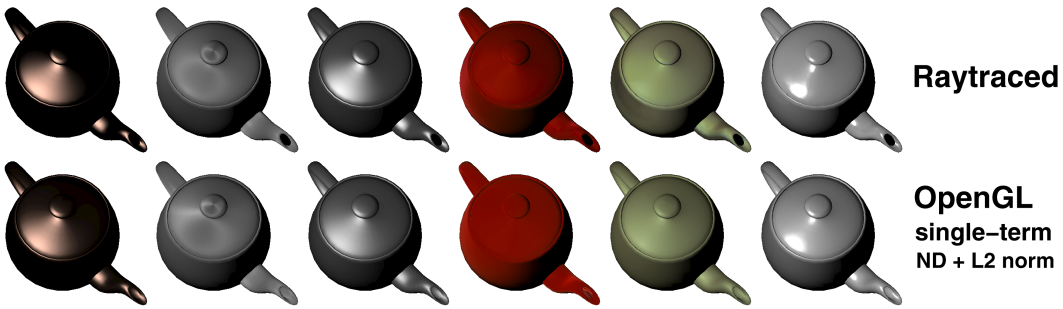


Figure 12: *HTSG copper, Poulin/Fournier's brushed metal, Lafortune/Willems' modified Phong, measured velvet, measured peacock feather, and measured grey vinyl.*

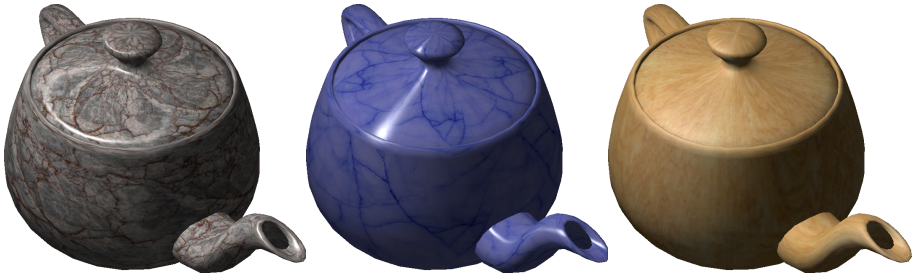


Figure 13: *Diffuse texture maps can be added to single-term separable decompositions for the specular highlight. Left to right: Ward's anisotropic BRDF; Ward's anisotropic BRDF (oriented orthogonally to the first example); (measured) varnished wood.*