Efficient Rendering of
Spatial Bi-directional Reflectance Distribution Functions

David K. McAllister,[1] Anselmo Lastra,[1] and Wolfgang Heidrich[2]

[1] Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, USA
[2] Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada

Abstract

We propose texture maps that contain at each texel all the parameters of a Lafortune representation BRDF as a compact, but quite general surface appearance representation. We describe a method for rendering such surfaces rapidly on current graphics hardware and demonstrate the method with real, measured surfaces and hand-painted surfaces.

We also propose a method of rendering such spatial bi-directional reflectance distribution functions using prefiltered environment maps. Only one set of maps is required for rendering the different BRDFs stored at each texel over the surface.

Categories and Subject Descriptors: I.3.7: Three-Dimensional Graphics and Realism

Keywords: Graphics Hardware, Reflectance & Shading Models, Rendering Hardware, Texture Mapping

1. Introduction

Many real surfaces have detailed reflectance variation both spatially over the surface and directionally over the incident and exitant hemispheres at each spatial location. We call this variation the Spatial Bi-directional Reflectance Distribution Function. The SBRDF is a six-dimensional function parameterized over incoming and outgoing light directions, and surface position. We represent an SBRDF as a texture map with all the parameters of a Lafortune BRDF (Lafortune, Foo et al. 1997) at each texel. This representation is general in that the BRDF at each texel may be completely unique, and within each texel, the Lafortune BRDF consists of a series of basis lobes that approximate the BRDF to any selected level of generality.

We present two techniques for real-time rendering of spatial bi-directional reflectance distribution functions that use the Lafortune representation. The first is a multipass method suitable for rendering on existing graphics hardware with standard point or directional lights. Our technique is not based on factoring and preevaluating a BRDF, but rather directly evaluating the BRDF equation as each fragment is shaded. The rendering algorithm is independent of the number of different materials in the scene. All SBRDFs can be rendered in the same set of rendering passes.

The second rendering method addresses global illumination of SBRDFs using environment maps. Since each texel has a unique BRDF we cannot preconvolve the environment map with the BRDF. Instead, we preconvolve with just the radially symmetric sharpness function (specular lobe) of the BRDF for several different sharpness values and store these as a 3D environment map. Then while rendering, the radiance computation includes environment map lookups at the lobe peak direction for each lobe of the BRDF. Because of the properties of the Lafortune representation, this enables anisotropic peak direction, anisotropic lobe shape (using a sum of lobes), the increasing reflectance of Fresnel reflection, forward scattering, back scattering, off-specular scattering, and spatial variation of gloss level.

2. Previous Work

Practical representations of BRDFs for hardware rendering fall into two major categories: those for surfaces illuminated by discrete lights, and those for surfaces illuminated by environment maps. Additionally, a growing body of work addresses spatial variation of the BRDF over a surface.

2.1. Spatial Variation of Reflectance

Varying the BRDF, or portions thereof, over a surface has been done for decades for offline rendering using hand-

made maps (Cook 1984), (Hanrahan and Haeberli 1990). Acquisition of such surface variation often uses constant specularity across a polygon (Yu, Debevec et al. 1999), (Sato, Wheeler et al. 1997), (Marschner, Westin et al. 1999) or limited angular resolution (Debevec, Taylor et al. 1996).

(Debevec, Hawkins et al. 2000) acquire a reflection model for human skin. Specular and diffuse parameters vary sharply across the surface, but other parameters are constant. (Lensch, Kautz et al. 2001) compute a shift-variant isotropic BRDF of an object with known geometry, focusing on finding a few basis BRDFs and representing each texel as a linear combination of these bases, with small per-texel detail.

(Dana, Ginneken et al. 1999) acquire a reasonably dense sampling of a planar surface and compute directionally varying image histogram statistics (the Bi-directional Texture Function) and the aggregate isotropic BRDF of a surface. (Liu, Shum et al. 2001) registered some samples from this database using image correlation and constructed statistically similar view dependent texture maps for rendering.

Polynomial Texture Maps (PTMs) (Malzbender, Gelb et al. 2001) represent a four-dimensional subspace of the SBRDF by holding the exitant direction constant (approximately in the normal direction) and varying the incident light direction over the hemisphere. PTMs may be evaluated efficiently in graphics hardware. For approximately the same texture size as a PTM, an SBRDF allows the variation in exitant direction required to render surfaces with arbitrary spatial BRDF variation on arbitrary geometry.

(Kautz and Seidel 2000) propose a flexible paradigm for evaluating spatially varying BRDFs in graphics hardware in which the BRDF equation is divided into linear operations such as dot products that can be evaluated by the graphics hardware, and nonlinear operations that are precomputed over their domain and stored in lookup tables as texture maps. Transformed BRDF parameters are stored per texel in texture maps and used for hardware shading. The SBRDF rendering method that we propose follows this paradigm since it directly stores BRDF parameters per texel. We will compare below our specialization of the paradigm to the models explored by Kautz.

2.2. BRDFs With Hardware Lights

(Fournier 1995) and (Heidrich and Seidel 1999) proposed factoring BRDF equations into simpler functions that could be precomputed into texture maps and rendered using compositing or multitexturing arithmetic in a constant number of rendering passes per light. (Kautz and McCool 1999) perform this factorization numerically using singular value decomposition. (McCool, Ang et al. 2001) solve problems of previous factorization approaches, with an elegant factorization that includes only positive factors. This method also works directly with arbitrary scattered bi-directional reflectance samples. By performing the factorization in log space, the highlights tend to be smoother, avoiding aliasing (though having broader peaks).

These methods all apply to surfaces with a uniform BRDF over the surface, whereas our method and that of (Kautz and Seidel 2000) store BRDF parameters over the surface, rather than evaluating the BRDF at regular parametric intervals and storing the results in texture maps. This is the key characteristic that enables total variation of the BRDF over the surface.

However, some spatial variation can be achieved in factorization methods by combining the BRDF with a standard diffuse texture, or using a weighted blend of several BRDFs. This is done by rendering passes for each BRDF for each light and accumulating the weighted results. In practice, this only works well for a small number of BRDFs because of the rendering cost of the layers and the storage space for the weight textures. We compare this to our method in Section 5.

One advantage of the BRDF factorization methods is that very flexible BRDFs may be represented, since each texel is essentially a coefficient of a BRDF representation. This means that these methods typically do not require a number of rendering passes proportional to the complexity of the BRDF, whereas the number of passes for the Lafortune representation used in SBRDFs depends on the number of lobes. BRDFs with a single lobe that has a depressed center, such as velvet, may require several Lafortune lobes to adequately approximate.

2.3. Environment Map Convolution

The original use of environment maps in (Blinn 1976) enabled mirror reflections. More recent work by (Cabral, Olano et al. 1999) and (Heidrich and Seidel 1999) enable glossy reflection of environment maps by convolving the environment map *a priori* with portions of the BRDF equation. These methods are thus only suitable for a single BRDF per set of maps. (Voorhies and Foran 1994) use environment maps to store the precomputed highlights from directional lights convolved with a Phong BRDF. (Bastos, Hoff et al. 1999) represent a sampling of the scene geometry using images with depth, and preconvolve these with spatially uniform portions of the BRDF for specific reflecting surfaces. This method allows reflections of local geometry rather than just light at infinity. (Kautz and McCool 2000) create a set of prefiltered environment maps for a given BRDF by fitting lobes that are fixed in orientation relative to a selected exitant polar angle. Many such angles are used, creating a 3D environment map. This 3D map is indexed based on the exitant direction and exitant polar angle. When summing multiple lobes each lobe uses a separate 3D environment map. This method has the important advantage of handling increasing lobe sharpness at grazing angles. (Kautz, Vázquez et al. 2000) provide faster map convolution, enabling dynamically changing glossy environment maps, and anisotropic environment maps constrained to an approximate Banks model.

These techniques do not address different BRDFs at each point on a surface, except in simple ways like using a textured diffuse term, or potentially modulating the sampled environment map with a specular albedo.

3. Representation

A spatial bi-directional reflectance distribution function can be acquired from measured data or painted by an artist. See the conclusion for a discussion of the considerations involved in the SBRDF paint program that we have under development.

In other work (McAllister 2002), we describe an SBRDF measurement and fitting system. We place a planar sample of a material on a pan-tilt-roll unit, and using this and a moving light, sample the incident and exitant hemispheres at a given angular density using a stationary camera. This provides a dense sampling of the entire six-dimensional SBRDF – between 300 and 8000 poses. We rectify these high dynamic range photographs and compute a reflectance image from each – typically on the order of 512 × 512 resolution. Our sampled SBRDFs measure up to 8 GB in size.

The reflectance values from the corresponding pixel of all rectified images are used as input to a nonlinear solver to compute the RMS best fit Lafortune BRDF parameters at that pixel, for a specified number of lobes. We use either Levenberg-Marquardt or a more efficient custom data fitting method. The resulting SBRDF files are less than 10 MB in size.

3.1. The Lafortune Representation

One can approximate the BRDF by projection into general basis functions (Westin, Arvo et al. 1992), (Lalonde and Fournier 1997), (Lafortune, Foo et al. 1997). Many models such as (Phong 1975) and (Ward 1992) only yield highlights in the reflection direction and thus cannot be used as a sum of bases for arbitrary BRDFs. The Lafortune representation is well suited for the shape that BRDFs typically have, is compact and is capable of representing interesting BRDF properties such as the increasing reflectance of Fresnel reflection, off-specular peaks and retro-reflection. The Lafortune representation consists of a sum of terms:

$$f_r(\omega_i \to \omega_r) = \rho_d + \sum_j \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \qquad (1)$$

where ρ_d is the diffuse reflectance. The terms in the summation are specular lobes. Each lobe j has an albedo, $\rho_{s,j}$, and a lobe shape, s_j. The lobe shape is a generalized Phong lobe:

$$s(\omega_i, \omega_r) = \left(\begin{bmatrix} \omega_{r,x} \\ \omega_{r,y} \\ \omega_{r,z} \end{bmatrix}^T \cdot \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{i,x} \\ \omega_{i,y} \\ \omega_{i,z} \end{bmatrix} \right)^n \qquad (2)$$

Figure 1: *Texture maps used in Figure 5(d): diffuse albedo (ρ_d), lobe albedo (ρ_s), lobe shape (C), and lobe exponent (n). Lobe shape maps -1..1 to 0..1. The lobe exponent is stored in the alpha channel of the lobe shape texture.*

The Lafortune representation is evaluated in local surface coordinates. The X and Y axes are the principal directions of anisotropy and Z is the normal. The matrix C is defined as C_x=*-1*, C_y=*-1*, C_z=*1* to cause ω_i to reflect about the normal, yielding a standard Phong lobe. But each lobe is significantly more general than a Phong lobe: the C coefficients may also take on other values to shear the specular lobe in ways that represent real surface scattering behavior but still enforce reciprocity and conservation of energy. The lobe's peak will be in the direction $C \cdot \omega_i$. For isotropic BRDFs, $C_x = C_y$. For off-specular reflection, $|C_z| < |C_x|$, pulling the lobe toward the tangent plane. For retroreflection, $C_x > 0$ and $C_y > 0$. When C_x and C_y have opposite sign, a lobe will forward scatter when parallel to the principal direction of anisotropy, but back scatter when perpendicular to it, as arises with parallel cylinder microgeometries (Westin, Arvo et al. 1992). The Lafortune representation's great flexibility to aim and scale each scattering lobe is key in using few lobes to approximate BRDFs. This property also enables our glossy environment mapping technique.

We represent the ρ albedo values as RGB channels but share the C_x, C_y, C_z, and n values between channels. For anisotropic materials we may also store the direction of anisotropy at each pixel, shared for all lobes.

4. Hardware Rendering

Surfaces with SBRDFs may be rendered using current graphics hardware such as an Nvidia Geforce 4 or an ATI Radeon 8500. This section provides the details of our formulation, texture representation, and rendering implementation, which we have implemented within a typical interactive rendering engine.

We first store the SBDRF parameters in texture maps. We store ρ_d in a single map, with the alpha channel being used for transparency. For each lobe j, two additional texture maps are defined. A three-channel map contains ρ_s and a four-channel map contains the lobe shape C_x, C_y, C_z, and n.

The value of a BRDF ranges from 0 to ∞, so it is possible for ρ_s to be greater than 1. Since seven parameters are used per lobe but the lobe only contains six degrees of freedom, ρ_s at each pixel may be scaled by an arbitrary factor e, while C is scaled by $\sqrt[n]{e}$ to preserve precision where necessary, since on current hardware all values will

be represented in 8-bit fixed point. Also, C may be mapped using the hardware's -2..2 mapping to preserve C values greater than 1.

4.1. Shader Formulation

The surface reflectance equation is:

$$L_r(\omega_r) = \int_{\Omega_i} f_r(\omega_i \to \omega_r) L_i(\omega_i)(\vec{N} \cdot \omega_i) d\omega_i \quad (3)$$

where ω_i is the incident direction, ω_r is the exitant direction, Ω_i is the incident hemisphere domain, $\vec{N}$ is the surface normal, f_r is the BRDF, and $L(\omega)$ is the radiance in the direction ω.

To implement Equation (3) in graphics hardware the equation must be expressed entirely using discrete arithmetic and split into portions corresponding to hardware operations. Let us first substitute the Lafortune BRDF representation into Equation (3) and replace the integral over the incident hemisphere by a discrete sum over the hardware lights:

$$L_r(\omega_r) = \sum_l \left(\rho_d + \sum_{j=1}^{k} \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i)(\vec{N} \cdot \omega_i) \quad (4)$$

We then move ρ_d within the sum over the BRDF lobes by using a delta function to cause ρ_d to only be added once per light:

$$L_r(\omega_r) = \sum_l \sum_{j=1}^{k} \left(\rho_d \delta(j-1) + \rho_{s,j} s_j(\omega_i, \omega_r) \right) L_i(\omega_i)(\vec{N} \cdot \omega_i) \quad (5)$$

On current graphics hardware we evaluate Equation (5) in two rendering passes per lobe per light. The pair of passes corresponds to the term within the double sum. These exitant radiance terms are summed in the frame buffer to yield the total exitant radiance.

The exponentiation within $s(\omega_i, \omega_r)$ may not be performed directly in current graphics hardware, so, following (Kautz and Seidel 2000), we create a 256 × 256 lookup table storing $f(x,n) = x^n$, and store this in a texture. However, we are unable to sample this map using sub-texel precision, as will be explained below. Because of this it is important to avoid sharp discontinuities in this map. In particular, for large n, the highlight falloff is very steep as x diminishes from 1. We thus remap x as: $x' = x^2$. Also, visual difference between highlight size is much greater for small specular exponents than for large, with specular exponents ranging from 0 to ∞. The lobe sharpness portion of the Lafortune representation could be replaced with an arbitrary sharpness function of the generalized dot product, such as a roughness parameter that varied from 0 to 1. However, our measured data uses the standard specular exponent formulation, so we remap n as: $n' = 255\ (n/255)^2$ to give more precision for smaller exponents.

4.2. First Pass

The first of each pair of passes is rendered into a p-buffer – a portion of video card memory that can store an off-screen image. P-buffers may be rendered to like a screen window and read from as texture maps. This first pass computes the lobe shape, $s(\omega_i, \omega_r)$, except for the exponentiation. The application sends vertex positions, normals, tangents, and texture coordinates for all objects with SBRDF surfaces. A vertex program transforms the camera and light vectors (for either point or directional lights) by the $\vec{T}$, $\vec{B}$, $\vec{N}$ frame into local surface coordinates. When normalized, this yields ω_r and ω_i. These are stored as texture coordinates and rasterized along with the standard surface texture coordinates. These vectors must be renormalized at each pixel. This is performed in the texture shader using a normalization cube map (Kilgard 1999).

Figure 2: *Hardware rendered result using our method. The couch, tan chair, blue chairs, table cloth, leaves, brass table, and gift wrap have measured SBRDFs. The cherry wood and floor wood are painted SBRDFs. Surfaces have 1 or 2 lobes per texel. Three hardware lights are used. Average frame rate for this scene is 18 fps.*

For each rasterized fragment, the texture shader also reads the lobe shape parameters C_x, C_y, C_z and n'. The first register combiner stage component-wise multiplies ω_i by $\langle C_x, C_y, C_z \rangle$. The second combiner stage computes the dot product of this product with ω_r, yielding x, the result of the generalized dot product portion of $s(\omega_i, \omega_r)$. This value is squared in the final combiner to yield x', then converted to eight-bit fixed point and replicated across the red, green, and blue components of the output; n' is bound to the alpha output, and the resulting fragment is written to the p-buffer.

4.3. Second Pass

The second of the two passes performed per lobe per light computes the rest of one term of Equation (5). The vertex program of pass 2 transforms the vertex to screen space, passing the result as texture coordinates. Interpolation of these texture coordinates must not use perspective correction so that the interpolated coordinates will precisely index the p-buffer pixel corresponding to the fragment now being rasterized. To disable perspective correct texture interpolation we multiply the texture coordinates by the vertex's w coordinate (Akeley 1992). The vertex program also passes the standard surface texture coordinates, then

computes the irradiance due to light *l*. The light direction is computed as in pass 1. The dot product with the vertex normal is computed in world space. Negative values of the dot product represent a light behind the polygon and are thus clamped to 0. This value is multiplied by the radiance of the light, optionally including the inverse squared falloff, and the result is stored as a color.

The p-buffer created in pass 1 is indexed by the screen coordinates to fetch the value written to this pixel in pass 1. This is *x', n',* which are used to perform a dependent texture read $s_j(\omega_i,\omega_r) = map(x',n')$. The result is stored in all three channels of the output color sent to the register combiners. The lobe albedo and diffuse albedo at the given point are also sampled and passed to the register combiners.

The first register combiner stage component-wise multiplies $\rho_{s,j}$ by $s_j(\omega_i,\omega_r)$ and to input A of the second stage. The interpolated irradiance is mapped to inputs B and D. For the first lobe ($j=1$) the combiner maps ρ_d to C and otherwise maps 0 to C. The output AB+CD is the exitant radiance for this term of Equation (5), and is sent to the frame buffer, with the alpha component of ρ_d being sent as the transparency of the fragment.

5. Hardware Rendering Results

Figure 5 and Figure 2 show rendered results for a variety of measured and painted surfaces. These were hardware rendered using an Nvidia Geforce 4 card.

The model of Figure 2 consists of 221,925 vertices and 261,549 triangles. Some surfaces have simple texture maps, but most surfaces use SBRDFs accounting for 188,833 vertices and 213,828 triangles. Twelve different SBRDFs are used, consuming 26 MB of texture memory. On a 2500 frame walkthrough path that renders an average of approximately 25% of the geometry per frame, rendering performance is shown in Table 1.

Especially for a single lobe, the rendering performance is very acceptable. One lobe is usually sufficient for interactive applications, with multiple lobes being more suitable for CAD applications where users typically study a surface more closely. Also, a single lobe is more acceptable within the SBRDF regime than for a spatially uniform surface since the high surface frequency can mask detailed highlight shape, as with our fabric samples. The white fabric of the lobby model uses two lobes. All other surfaces use one lobe.

	SBRDF			No light, no tex	Light, no tex	No light, tex	Light, tex
# lights	# lobes						
	1	2	3				
1	50.7	29.5	20.8	176.5	156.8	109.7	95.7
2	29.4	16.2	11.1		140.6		93.1
3	20.8	11.1	7.57		122.1		77.0

Table 1: *Frame rate for SBRDF shader, for the Figure 2 scene. 800×600, 2× AA. Although the model contains SBRDFs with 1 or 2 lobes, for timing tests, all surfaces were forced to the stated number of lobes. SBRDF results are compared against simple one-pass approaches.*

The major source of visual artifacts stems from using the lookup table. The rounding to eight bits as *x'* and *n'* are stored in the p-buffer causes adjacent pixels with slightly different *x'* values to index the same table entry, yielding artifacts at highlight tails.

Another artifact results from clamping *x'* to 1 before writing it to the p-buffer in the first pass. The Lafortune representation depends on dot product values $x > 1$ to represent the increasing reflectance at grazing angles of Fresnel reflection. If $n \geq 1$, for $x > 1$ we have $x^n > x$, but for $x < 1$ we have $x^n < x$. With x being clamped to 1 it is impossible to yield a bi-directional reflectance value x^n greater than 1. Using the exponentiation lookup table the lookup result will instead exactly equal 1. This value is then multiplied by the irradiance, which approaches zero as the light direction approaches the grazing angle, yielding a final exitant radiance usually less than 1. This artifact will be alleviated by increased frame buffer dynamic range.

5.1. Comparison to BRDF Factorization

With BRDF factorization methods, for a given surface to have different BRDFs at different points, each unique BRDF must be stored in a set of texture maps, and a number of rendering passes must be performed for each BRDF for each light, as mentioned in (McCool, Ang et al. 2001). These results are modulated by per-texel weights for each BRDF in order to composite the per-BRDF renderings into a final result. For a very small number of BRDFs, this representation is compact, since it is essentially a palette with per-texel weights, and the rendering method works quite well.

However, an advantage of storing BRDF parameters per texel is that the number of required rendering passes does not depend on the number of different BRDFs. Table 2 compares the number of rendering passes required on Nvidia Geforce 4 hardware for a varying number of lights, and for varying surface complexity. For SBRDFs, each texel has a unique BRDF, but the number of lobes is fixed for the surface (though not for a whole scene). For McCool's method, the total number of BRDFs in the surface is fixed, and each BRDF is evaluated at each texel.

Figure 5(d) shows a single SBRDF made by combining many measured and synthetic SBRDFs. This SBRDF consists of over one million unique BRDFs since it is a 1024 × 1024 image. However, many texels are obviously very similar to others. We have not exploited this similarity, but a method such as (Lensch, Kautz et al. 2001) would do so. By manually analyzing this SBRDF, we estimate that 48 significantly different basis BRDFs appear in the SBRDF. Assuming that factorization methods such as McCool's could be extended to handle the per-texel deviation from the bases and still render in N+1 passes per BRDF for N lights, the final column of Table 2 shows the

	SBRDF			McCool			
# lights	**# lobes**			**# BRDFs**			
	1	**2**	**3**	**1**	**2**	**3**	**48**
1	2	4	6	2	4	6	96
2	4	8	12	3	6	9	144
3	6	12	18	4	8	12	192
4	8	16	24	5	10	15	240

Table 2: *Number of passes required for our method vs. the factorization method of McCool for varying number of lights and varying surface complexity on an Nvidia GeForce 4.*

estimated number of passes to render Figure 5(d) using McCool's method. The values are comparable for other BRDF factorization methods. This shows that rendering surfaces with BRDF parameters per texel is more efficient than rendering basis BRDFs for surfaces with a great deal of spatial BRDF variation. On more programmable hardware, these numbers could be reduced to a constant cost by storing BRDF indices rather than weights per texel, and evaluating only the indexed BRDFs at each pixel.

5.2. Comparison to Kautz-Seidel

The SBRDF rendering method that we propose fits the paradigm of (Kautz and Seidel 2000) in that linear operations are used to compute texture indices. Nonlinear operations are stored in texture maps, and the results are used in additional linear operations, completing the BRDF evaluation. Kautz & Seidel demonstrated their method with a novel anisotropic Blinn-Phong model. The potential flexibility of the paradigm was shown using other models demonstrated with software simulations. Our method shares some precision and clamping artifacts that Kautz & Seidel encountered. We have successfully applied this paradigm despite dependent texture reads not being as general as predicted by Kautz and Seidel.

But the major difference we propose that applies to current graphics hardware is to use the Lafortune representation, which has several properties not possessed by the models explored by Kautz & Seidel. In particular, the lobe directions of the Lafortune representation can aim in arbitrary directions relative to the incident direction, which makes a sum of lobes much more effective than with Phong (Phong 1975) or Ward (Ward 1992) bases.

The Lafortune representation is also particularly well suited to this paradigm. Our method requires two passes per lobe per light on a GeForce 4, but could most likely be done in one pass on hardware that allows register combiner results to be used as dependent texture coordinates within the same pass. On a GeForce 4, the anisotropic Ward model would require at least ten texture reads and four rendering passes per lobe per light, but likely only three passes on a Radeon 8500. The Banks model could be evaluated in two passes per light.

6. Global Illumination of SBRDFs

Beyond illuminating SBRDFs with discrete point or directional lights, this section discusses illuminating SBRDFs with incident light from all directions using environment maps. Environment maps represent only the incident direction of the radiance, as if all illumination comes from infinity. Each point in the scene receives the same illumination, invariant of its location.

Previous methods for glossy reflection of environment maps in graphics hardware share the constraint that the environment map must be preconvolved with (a portion of) each different BRDF. Thus, a different set of environment maps is required for each different BRDF. Although burdensome, this is at least possible for scenes consisting of a fairly small number of objects, each with one or a few discrete BRDFs. However, in the SBRDF regime, every point on every surface is treated as having a unique BRDF, so the existing methods do not directly apply. We present a method for rendering from preconvolved environment maps that allows each point on each surface to have a completely different BRDF.

6.1. Environment Map Convolution Formulation

The formulation begins with a modified form of Equation (3), applied to the Lafortune representation, with separate diffuse and specular terms:

$$L_r(\omega_r) = \rho_d \int_{\Omega_i} L_i(\omega_i)\left(\vec{N}\cdot\omega_i\right)d\omega_i + \sum_j \rho_{s,j} \int_{\Omega_i} s_j(\omega_i,\omega_r) L_i(\omega_i)\left(\vec{N}\cdot\omega_i\right)d\omega_i \tag{6}$$

The incident radiance $L_i(\omega_i)$ is stored in the environment map, indexed by the incident direction. The diffuse term can be easily encoded in an environment map indexed simply by $\vec{N}$:

$$D\left(\vec{N}\right) = \int_{\Omega_i} \left(\vec{N}\cdot\omega_i\right) L_i(\omega_i)\, d\omega_i \tag{7}$$

$D\left(\vec{N}\right)$ is independent of the BRDF, so it is precomputed once for all objects that are to reflect the environment map $L_i(\omega_i)$. This was done by (Miller and Hoffman 1984) and (Green 1986). The specular terms also take advantage of precomputed maps. Just as $\vec{N}$ is used to index the preconvolved diffuse map, a function of the view direction will index a preconvolved specular environment map:

$$p_j(\omega_r) = \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{r,x} \\ \omega_{r,y} \\ \omega_{r,z} \end{bmatrix} \tag{8}$$

$p_j(\omega_r)$ is the peak vector of the lobe-shaped sampling kernel – the incident direction of maximum influence on the exitant radiance toward ω_r due to lobe j. Equation (6) becomes

$$L_r(\omega_r) = \rho_d D(\vec{N}) + \sum_j \rho_{s,j} \int_{\Omega_i} (p(\omega_r)\cdot\omega_i)^{n_j} L_i(\omega_i)(\vec{N}\cdot\omega_i)\,d\omega_i \quad (9)$$

We define the specular environment map as:

$$S(\omega_p, n) = \int_{\Omega_i} \left(\frac{\omega_p}{\|\omega_p\|} \cdot \omega_i \right)^n L_i(\omega_i)\,d\omega_i \quad (10)$$

As discussed below, this map is parameterized both on the incident kernel peak direction ω_p and on the exponent n. The exitant radiance formulation used in hardware rendering becomes:

$$L_r(\omega_r) \approx \rho_d D(\vec{N}) + \sum_j \rho_{s,j} S(p(\omega_r), n_j) \|p(\omega_r)\|^{n_j} (\vec{N}\cdot p(\omega_r)) \quad (11)$$

The $\|p(\omega_r)\|^{n_j}$ factor arises because S is computed with a normalized ω_p, so the incident radiance must still be scaled by the magnitude of the lobe. This equation is only an approximation since the irradiance falloff $\vec{N}\cdot\omega_i$ must be computed inside the integral over ω_i, but this could not then be stored in an environment map since it would be parameterized by both $p(\omega_r)$ and $\vec{N}$. We instead weight all incident directions equally within the integral but weight $S(\omega_p,n)$ by $\vec{N}\cdot p(\omega_r)$. This problem and resolution were explained by (Kautz and McCool 2000). This is a high quality approximation, and the quality improves for increasing values of n. A problem our method shares with many preconvolution methods is that by removing $\vec{N}\cdot\omega_i$ from the integral, some light from below the surface is included in $L(\omega_r)$. This has not presented a practical problem.

Two key facts allow environment maps to illuminate spatial BRDFs. First, the Lafortune representation uses a specular exponent, which creates radially symmetric lobes. In general, only radially symmetric lobes may be used to preconvolve an environment map because otherwise the map would only be correct for a single normal direction. Second, the Lafortune representation directly computes a general lobe peak vector. Thus, all BRDF properties that the Lafortune representation expresses using the peak vector – anisotropic peak direction, anisotropic lobe shape (by summing radially symmetric lobes), the increasing reflectance of Fresnel reflection, and forward reflective, retroreflective, and off-specular peaks – are independent of the environment map.

6.2. Implementation for Graphics Hardware

Several representation alternatives arise for the $S(\omega_p,n)$ map. The possibilities are constrained by the graphics hardware's small choice of texture representations. One possibility is to represent S in a single cube map, with the most specular value of n being stored in the finest MIP level, and successively diffuse n being stored in coarser levels. The decreasing resolution of MIP levels corresponds nicely to the decreasing high frequency content of environment maps convolved with wider specular lobes. This property allows the most compact representation of S. Another approach is to store a small set of cube maps for discrete values of n, and render with each of these, weighting each by basis functions evaluated at the pixel's n, yielding a result interpolated to the pixel's n from maps for nearby values of n. A third possibility is to use a 3D texture, with the s and t map dimensions mapping to a parabolic map (Heidrich and Seidel 1999) and the r dimension mapping to n. This is similar to the representation by (Kautz and McCool 2000), except their r dimension mapped to exitant polar angle.

Figure 3: *Software rendering simulating our prefiltered environment map technique. Specular exponents range from 1 to 5 for the couch, 5 to 15 for the grain of the wood, and 75 to 150 for the wood foreground. The wood is increasingly reflective at grazing angles.*

We believe glossy environment reflection of SBRDFs is impractical for graphics hardware with register combiner fragment shading. Today's hardware does not allow choosing MIP levels based on a value sampled from a texture map, so there is no way to choose a gloss level based on the pixel's n. This prevents use of the single MIP-map representation. The 3D texture representation could be used, except it would be difficult to compute the indices into this map for each fragment being shaded. If the C_x, C_y, and C_z values were constant over the polygon, the s and t coordinates could easily be computed in a vertex program with a per-fragment n value possibly mapped to r. This would yield spatially varying gloss level, but would not enable spatially varying lobe direction effects. The representation with several cube maps could be rendered with one pass per lobe per map level by using the bump map vector perturbation circuit to compute $p(\omega_r)$ at each fragment. However, the computation of fragment weights based on the fragment n and the current map's n would be difficult and may require different weight maps for each

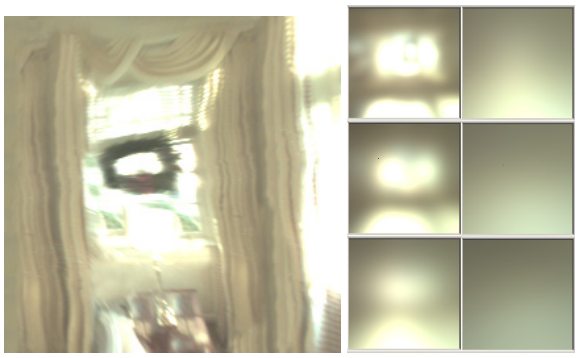

Figure 4: *Left: One face of high dynamic range cube map used in figure 3. Right: Prefiltered maps for specular exponents (top to bottom, left to right) 256, 64, 16, 4, 1, 0 (diffuse). All prefiltered maps are 128 × 128 pixels.*

environment map *n*. Alternatively, *n* could be held constant here, yielding a modified form of bump mapped environment mapping.

For future hardware, we believe all three representations will be practical as fragment (pixel) shading becoming more general. Using a software renderer, we have implemented both cube map approaches. The shader directly follows Equation (11), with the note that the view vector is rasterized in local surface coordinates, transformed by C to yield $p(\omega_r)$, and then transformed by the local frame to world coordinates for sampling the environment map. This requires rasterizing and renormalizing $\vec{N}$ and $\vec{T}$. As well all linear interpolation and renormalization of vectors, this can cause artifacts in regions of high curvature.

For the set of discrete cube maps, we used linear interpolation between values of *n* for specular exponents 1, 4, 16, 64, and 256. The blending artifacts were minimal, but the large number of texture accesses makes this method less suitable for real-time rendering.

The MIP-mapped cube map representation works well. This method is the most space efficient, requiring a single map, the most bandwidth efficient, requiring a single trilinear sample, and finally, it is the most computationally efficient since the texture coordinate computation is performed in a dedicated cube mapping circuit, rather than in staged fragment shader computations. Cube map MIP levels generated via environment map convolution with different specular exponents are not equivalent to general MIP level computation. However, it is very visually similar and has not presented a problem.

We have not yet implemented the 3D texture representation with parabolic maps. Although the parabolic mapping would need to be computed in the fragment shader, rather than a dedicated circuit, we believe the implementation should be straightforward and provide good results.

Figure 3 shows a rendered result that employs the high dynamic range environment map of Figure 4. Note the synthetic texture of wood. The grain, which has a specular exponent of about 10, yields much lower gloss highlights than the wood foreground, with a specular exponent of about 150. The measured white upholstery fabric is represented with two lobes. At most texels, these have specular exponents of about 1 and 5 for the silky thread and about 1 and 2 for the cotton thread. For most silk texels the high exponent lobe is anisotropic – forward scattering when parallel to the threads, and back scattering when perpendicular to them.

Figure 6 shows a rendered result with a brushed metal and fabric SBRDF, illuminated by the environment map of Figure 4. For the metal texels of the SBRDF, only the direction of anisotropy varies. The metal BRDF uses three anisotropic lobes with specular exponents of 18, 56, and 184.

7. Conclusion and Discussion

Rendering surfaces with completely different BRDFs at each texel is practical on today's graphics hardware using the representation and method described in this paper. The method builds upon existing work by using a flexible, compact Lafortune BRDF representation that yields quite convincing renderings of many measured or synthetic surfaces at satisfactory frame rates. For near future hardware, this paper offers an additional rendering method that again takes advantage of the flexibility of the BRDF representation to yield spatially varying environment mapped illumination.

To conclude, we would like to discuss hardware-related issues that arose during this work, and discuss some future directions. First, we observe that, although lookup tables are the key enabler of many advanced shading techniques today, such as BRDF factorizations, normalization maps, the nonlinear function maps of (Kautz and Seidel 2000), and our exponent table, this is probably not the most viable approach in the long term. This is because memory bandwidth has been growing much more slowly than on-chip computation power, so it makes sense to reserve memory bandwidth for content that cannot be derived by computation. This calls for efficient, general per-fragment shading.

Texture sampling within graphics hardware assumes that texel values are linear in the exitant radiance of the pixel, so linear interpolation of texels is acceptable. However, when storing input parameters to nonlinear functions in texture maps, linear interpolation is inappropriate. We haven't noticed artifacts due to interpolating C values. We believe the main artifact would be a depressed highlight similar to failing to renormalize an interpolated normal, but that would only occur when two adjacent texels had very different C values.

One approach to this problem is to allow the fragment shader the flexibility of receiving all sampled texels for this texture lookup unfiltered, together with their filter weights, and processing them as desired in the fragment shader. For

minification filtering, MIP levels may be created with parameters that will yield shader results that approximate shading followed by filtering.

Regarding future work, the *C* matrix consists of simply the diagonal elements C_x, C_y, C_z and can perturb a vector arbitrarily within the local coordinate frame. However, it is not as general as bump mapping. For rendering SBRDFs with bump maps one could use two three-channel maps to represent the entire (symmetric) matrix C, including the normal perturbation for bump mapping, rotation to the direction of anisotropy, and the shear by C_x, C_y, C_z. At each fragment, ω_i is simply transformed by this 3×3 matrix.

We believe it may be possible to render two lights per pair of passes in our current rendering regime. The first pass would compute *x'* for each light, and store *x'* and *n'* for each in the frame buffer. The second pass would employ dependent texture reads for each light.

For creating and editing SBRDFs, we have in progress a paint program that natively supports SBRDFs. Many common painting operations treat pixel values simply as data to be copied. These are simple to apply to BRDF pixels. Other paint operations require interpolation of pixel values. For nonlinear BRDF representations such as Lafortune, the most accurate interpolation method is to sample the BRDFs over their domain, interpolate between the samples, and fit a new BRDF to the interpolated sample vector. This always yields a physically plausible BRDF (McAllister 2002).

Acknowledgements

We appreciate the talents of Ben Cloward of Vicious Cycle, Inc. for creating the Carolina Inn lobby model. We appreciate Alexander Stevenson who helped port the rendering code to use vertex programs. We appreciate helpful discussions with Steve Molnar, Chris Wynn, and others at Nvidia.

References

Akeley, K. (1992). Reality Engine Graphics. Proc. of SIGGRAPH '92, Chicago, IL.

Bastos, R., K. Hoff, et al. (1999). Increased Photorealism for Interactive Walkthroughs. Proc. of Symposium on Interactive 3D Graphics.

Blinn, J. F. (1976). "Texture and Reflection in Computer Generated Images." Communications of the ACM **19**(10): 542-546.

Cabral, B., M. Olano, et al. (1999). Reflection Space Image Based Rendering. Proc. of SIGGRAPH '99, Los Angeles, CA.

Cook, R. L. (1984). Shade Trees. Proc. of SIGGRAPH '84.

Dana, K. J., B. v. Ginneken, et al. (1999). "Reflectance and texture of real-world surfaces." ACM Transactions on Graphics **18**(1): 1-34.

Debevec, P., T. Hawkins, et al. (2000). Acquiring the Reflectance Field of a Human Face. Proc. of SIGGRAPH '00.

Debevec, P. E., C. J. Taylor, et al. (1996). Modeling and Rendering Architecture from Photographs. Proc. of SIGGRAPH '96, New Orleans, LA.

Fournier, A. (1995). Separating Reflection Functions for Linear Radiosity. Rendering Techniques '95 (Proc. of Eurographics Workshop on Rendering), Springer.

Green, N. (1986). "Environment Mapping and Other Applications of World Projections." Computer Graphics and Applications **6**(11): 21-29.

Hanrahan, P. and P. Haeberli (1990). Direct WYSIWYG Painting and Texturing on 3D Shapes. Proc. of SIGGRAPH '90, Dallas, TX.

Heidrich, W. and H.-P. Seidel (1999). Realistic, Hardware-accelerated Shading and Lighting. Proc. of SIGGRAPH '99, Los Angeles, CA.

Kautz, J. and M. D. McCool (1999). Interactive Rendering with Arbitrary BRDFs using Separable Approximations. Rendering Techniques '99 (Proc. of Eurographics Workshop on Rendering), Granada, Spain.

Kautz, J. and M. D. McCool (2000). Approximation of Glossy Reflection with Prefiltered Environment Maps. Graphics Interface '00.

Kautz, J. and H.-P. Seidel (2000). Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware.

Kautz, J., P.-P. Vázquez, et al. (2000). A Unified Approach to Prefiltered Environment Maps. Rendering Techniques '00 (Proc. of Eurographics Workshop on Rendering), Springer.

Kilgard, M. J. (1999). NVIDIA OpenGL Cube Map Texturing.

Lafortune, E. P. F., S.-C. Foo, et al. (1997). Non-Linear Approximation of Reflectance Functions. Proc. of SIGGRAPH '97.

Lalonde, P. and A. Fournier (1997). "A Wavelet Representation of Reflectance Functions." IEEE Transactions on Visualization and Computer Graphics **3**(4): 329-336.

Lensch, H., J. Kautz, et al. (2001). Image-Based Reconstruction of Spatially Varying Materials. Rendering Techniques '01 (Proc. of Eurographics Workshop on Rendering), London, England.

Liu, X., H.-Y. Shum, et al. (2001). Synthesizing Bidirectional Texture Functions for Real-World Surfaces. Proc. of SIGGRAPH '01, Los Angeles, CA.

Malzbender, T., D. Gelb, et al. (2001). Polynomial Texture Maps. Proc. of SIGGRAPH '01, Los Angeles, CA.

Marschner, S. R., S. H. Westin, et al. (1999). Image-based BRDF Measurement Including Human Skin. Rendering Techniques '99 (Proc. of Eurographics Workshop on Rendering), Granada, Spain.

McAllister, D. K. (2002). A generalized Representation of Surface Appearance. Department of Computer Science. Chapel Hill, North Carolina, University of North Carolina at Chapel Hill**:** 103.

McCool, M., J. Ang, et al. (2001). Homomorphic Factorization of BRDFs for High-Performance Rendering. Proc. of SIGGRAPH '01, Los Angeles, CA.
Miller, G. and R. Hoffman (1984). Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments. SIGGRAPH '84 Course Notes - Advanced Computer Graphics Animation.
Phong, B. T. (1975). "Illumination for Computer Generated Pictures." Communications of the ACM **18**: 311-317.
Sato, Y., M. D. Wheeler, et al. (1997). Object Shape and Reflectance Modeling From Observation. Proc. of SIGGRAPH '97, Los Angeles, FL.
Voorhies, D. and J. Foran (1994). Reflection Vector Shading Hardware. Proc. of SIGGRAPH '94.
Ward, G. (1992). Measuring and Modeling Anisotropic Reflection. Proc. of SIGGRAPH '92, Chicago, IL.
Westin, S. H., J. R. Arvo, et al. (1992). Predicting Reflectance Functions from Complex Surfaces. Proc. of SIGGRAPH '92, Chicago, IL.
Yu, Y., P. Debevec, et al. (1999). Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs. Proc. of SIGGRAPH '99, Los Angeles, CA.

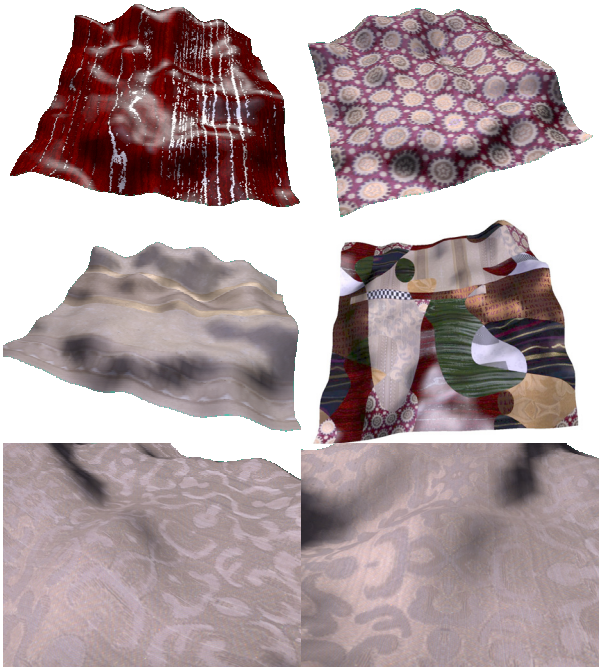

Figure 5: *Hardware rendered results using the method of this paper. a) measured gilded wall paper, b) hand painted cherry wood, c) measured gift wrap, d) hand-composite map with 10 measured and synthetic materials. e,f) Measured, anisotropic upholstery fabric with two lobes per texel. The foreground and background threads change relative brightness under 30 ˚ rotation.*

Figure 6: *Anisotropic brushed metal teapot with spatially varying direction of anisotropy, with white fabric lettering. One SBRDF with three lobes is used for the entire surface. Illumination comes from the prefiltered high dynamic range lobby environment map.*