Real-Time BRDF Editing in Complex Lighting

Aner Ben-Artzi
Columbia University

Ryan Overbeck
Columbia University

Ravi Ramamoorthi
Columbia University

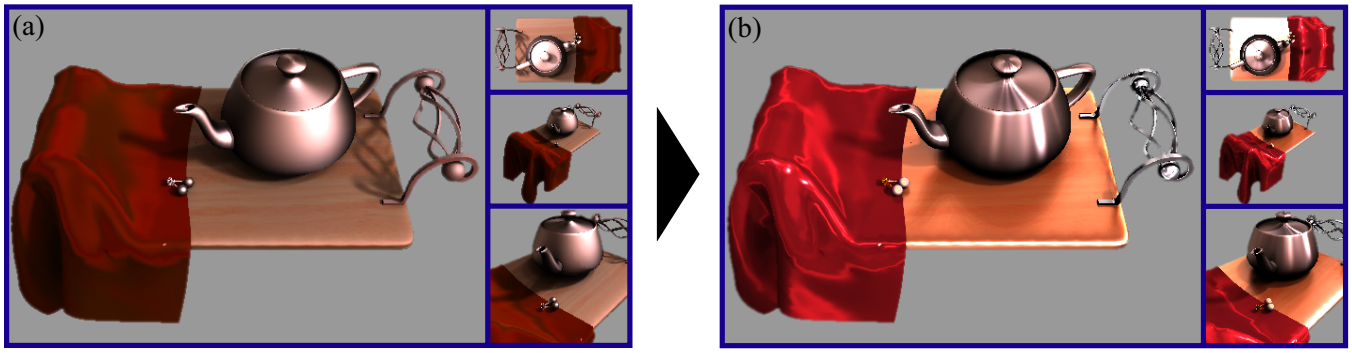


Figure 1. Editing Session. Our system was used to make real-time edits to all of the BRDFs in this scene, illuminated by 4,000 lights. The cloth and handles use measured BRDFs, and the other objects use various analytic models. Besides adjusting analytic parameters to make the teapot more anisotropic, and the tray more specular, freehand edits of the measured materials were used to create novel BRDFs. A small number of fixed views show the user the view-dependent effects of their edits. Notice how the dark reflection of the teapot in (b) appears at a different location in each view, and that the detailed shadow of the handle is diminished. Details of similar edits are shown in Figures 2 and 6.

Abstract

Current systems for editing BRDFs typically allow users to adjust analytic parameters while visualizing the results in a simplified setting (e.g. unshadowed point light). This paper describes a real-time rendering system that enables interactive edits of BRDFs, as rendered in their final placement on objects in a static scene, lit by direct, complex illumination. All-frequency effects (ranging from near-mirror reflections and hard shadows to diffuse shading and soft shadows) are rendered using a precomputation-based approach. Inspired by real-time relighting methods, we create a linear system that fixes lighting and view to allow real-time BRDF manipulation. In order to linearize the image's response to BRDF parameters, we develop an intermediate curve-based representation, which also reduces the rendering and precomputation operations to 1D while maintaining accuracy for a very general class of BRDFs. Our system can be used to edit complex analytic BRDFs (including anisotropic models), as well as measured reflectance data. We improve on the standard precomputed radiance transfer (PRT) rendering computation by introducing an incremental rendering algorithm that takes advantage of frame-to-frame coherence. We show that it is possible to render reference-quality images while only updating 10% of the data at each frame, sustaining frame-rates of 25-30fps.

1 Introduction

Computer graphics develops tools for skilled artists and engineers to create images based on the building blocks of geometry, material properties, and lighting. Interactive design of these scene components allows users not only to converge on a goal more quickly, but also to explore the design space and discover new effects. While final-quality, real-time *relighting* is an active field of research [Pellacini et al. 2005], almost no work exists on modifying *BRDF*s in their final scene placement. Indeed, the parameters of analytic BRDF models are typically chosen without any interactive feedback of how editing them affects the scene appearance when using final lighting and accurate shadows. Even less work addresses editing tabulated BRDF data [Ashikhmin et al. 2000b; Lawrence et al. 2006], and these are also restricted to simple lighting.

We present a real-time rendering system for editing the parameters of analytic BRDFs, and manipulating measured reflectance data. The visual effects of such edits are tightly coupled with the scene geometry and lighting, and we render the BRDFs using their final placement within a scene using complex illumination and cast shadows. As seen in Figures 1, 2, and 6, we can edit the parameters of common models such as Cook-Torrance [1982] and Ashikhmin-Shirley [2000a], as well as edit measured materials from the Dana et al. [1999] (CURET) and Matusik et al. [2003] databases. Section 4 describes how to use our method for these and other BRDFs.

Algorithmically, our method is closest to recent precomputed radiance transfer (PRT) methods for relighting [Sloan et al. 2002, 2003; Ng et al. 2003]. All-Frequency effects such as mirror-like reflections, glossy and diffuse BRDFs, and soft and sharp shadows are important when editing BRDFs, and we are inspired by all-frequency relighting approaches for static scenes [Ng et al. 2003, 2004; Wang et al. 2004; Liu et al. 2004]. Those methods fix the BRDF while allowing real-time illumination editing. By contrast, we fix the lighting and allow real-time BRDF editing. Like Ng et al. [2003], we fix the view, but extend our system to support multiple static views simultaneously, as seen in Figure 1.

As discussed in Section 3, the adaptation is not straightforward, and we describe a number of technical contributions to make final-placement BRDF editing tractable and accurate. We use an intermediate curve-based representation (Section 4) that linearizes the effect of (often non-linear) BRDF parameters. Our method does not render interreflection effects since these break the linear response of the image to the BRDFs of scene objects. Our representation also exposes only one dimension of the space of possible edits to the rendering system at a time; simplifying rendering and precomputation calculations to 1D. This representation is very similar to distributions used by analytic functions, and common factors of measured data. By exposing this intermediate representation to the user, these 1D curves can be used to directly manipulate the BRDF. We also discuss several modifications to the precomputation step which make it faster and more robust (Section 5). Finally, we describe an improvement to the standard PRT rendering algorithms (Section 6). We leverage the coherence between frames in an editing session to incrementally update only the part of the BRDF that has changed. This allows us to often retain an exact solution at the same cost as standard non-linear approximations. In addition, because we use the full representation of both lighting and BRDFs, we render all-frequency effects in both (see Figure 4).

email: {aner, rso, ravir}@cs.columbia.edu

2 Previous Work

Real-Time Rendering: Current hardware rendering methods (excluding PRT, discussed below) typically cannot simultaneously handle both complex lighting and cast shadows. Ramamoorthi and Hanrahan [2002] render general complex materials in environment lighting with little precomputation, using a spherical harmonic approximation. It might be possible to extend their method (or related work on rendering Lafortune lobes by [McAllister et al. 2002]) to some types of interactive BRDF editing, but they still ignore cast shadows, which substantially affect the final appearance of the scene. Modern soft-shadowing methods [Assarsson et al. 2003] give impressive results, but typically assume simple or fixed BRDFs. Such methods usually rely on GPU shaders which often lose their real-time capabilities as the BRDF becomes more sophisticated and more than a few lights are used.

Precomputed Radiance Transfer: The linearity of light transport has been used in a large body of work on relighting. Beginning with Nimeroff et al. [1994], lighting could be edited by representing the user's input as a linear combination of lighting functions, that can be combined to generate a relit image. More recently, precomputed radiance transfer (PRT) for static scenes [Sloan et al. 2002] has rekindled an interest in basis projections of editable lighting. This, and subsequent work later enabled changing view [Sloan et al. 2003] and dynamic geometry [Sloan et al. 2005; Zhou et al. 2005], but are all limited to low-frequency effects due to the choice of spherical harmonics as the basis for lighting. Editing within a restricted family of BRDFs is possible with [Sloan et al. 2002], but only for low-frequency, radially symmetric Phong-like models.

Since it is important to preserve the full richness of effects in the BRDF and lighting while editing, we focus instead in this paper on all-frequency wavelet-based approaches [Ng et al. 2003]. While recent advances allow for changing view as well as lighting [Sloan et al. 2004; Wang et al. 2004; Ng et al. 2004], they require a precomputed factorization or tabulation of the BRDF that can neither be edited continuously, nor recomputed in real-time.

It is also important to note some differences that make BRDF editing a fundamentally harder problem, and underlie our assumptions. The BRDF lobe at a given pixel is affected by all of the: lighting, view, and geometry (surface normal). This is why we must fix these quantities, while relighting methods can sometimes factor shadowing effects from material properties [Ng et al. 2004].

BRDF Representations: We build on existing BRDF representations including analytic models, and factored BRDFs. Parametric BRDFs have long been common in computer graphics, beginning with Phong [1975], Blinn-Phong [1977], and their extensions: Ashikhmin-Shirley [2000a], Cook-Torrance [1982] and Lafortune [1997]. More recently, there has been a focus on measured reflectance, which can often be more realistic. Hybrid approaches use a parametric-based form, with a measured or user-specified distribution for components like the half-angle dependence [Ashikhmin et al. 2000b; Ngan et al. 2005]. Among purely data-driven methods, factored BRDF models [Kautz and McCool 1999; McCool et al. 2001] are accurate, compact and suitable for hardware rendering. Matusik et al. [2003] acquired a database of BRDFs, and demonstrated some (offline) editing with a non-linear representation.

BRDF Editing: Parametric models can be easily edited (under point source lighting) by simply moving sliders for their parameters, and such software is available for research [Rusinkiewicz 1998a] and production (Maya, 3DMax).

For specific applications like car paint design, specialized systems [Ershov et al. 2001] have been developed to allow edits of a custom designed BRDF model. They quote interactive frame-rates (1-6fps), but like GPU approaches suffer from degraded speed as the BRDF becomes more complex, and do not demonstrate general complex illumination. To our knowledge commercial systems allow users to specify weights for a linear combination of pre-defined materials [PEARL], while viewing the results in a static scene. However, a few pre-defined materials do not suffice for many applications [Matusik et al. 2003].

Previous work has not focused much on real-time *editing* of data-driven reflectance. Ashikhmin et al. [2000b] and Jaroskiewicz and McCool [2003] render based on 2D maps, which can be edited as an image using standard image-editing software. However, this approach is not always intuitive for deriving new physically-based BRDFs, and these methods are only amenable to point lighting. Lawrence et al. [2006] demonstrate how 1D BRDF factors based on the half-angle/difference-angle parameterization can be used to edit measured data, but still using a point source.

We make use of existing BRDF editing techniques, and extend them to work with complex lighting and shadows. To do so, we develop an intermediate BRDF representation for rendering that supports both parametric and measured BRDFs in a unified way. This representation is similar to Ashikhmin et al.'s [2000b] distribution functions, but works for a wider class of analytic BRDFs. It is also similar to the 1D factors of [Lawrence et al. 2006], but with extensions that preserve perfect accuracy in the factorization step, and admit the use of more general parameterizations.

3 Overview

We use a precomputed approach to calculate all of the static scene-data and factor out the user-editable parts for render-time multiplication. This amounts to expressing the outgoing radiance at a point, in the view direction, $R(x,\omega_o)$, as a dot-product, which we arrive at in Equation 6. We begin with the reflection equation:

$$R(x,\omega_o)=\int_{\Omega_{4\pi}} L(x,\omega_i)V(x,\omega_i)\rho(\omega_i,\omega_o)S(\omega_i,\omega_o)d\omega_i, \qquad (1)$$

where L is the lighting[1], V is the binary visibility function, and ρ is the BRDF, all expressed in the local coordinate-frame. In this frame, the BRDF is the same at each point, allowing edits of a single BRDF to be used over the whole surface. For now, S is the cosine-falloff term, $\max(\cos\theta_i,0)$, but later it will be used to express arbitrary (static) functions of the light and view directions.

In the rest of this paper, we consider a single term of a BRDF. A sum of terms (such as diffuse-like and specular lobes) can be handled separately and summed[2]. Spatial variation is currently handled with a texture for any term that simply multiplies the corresponding image pixels. Editing spatial weights of the terms through texture painting is a natural extension, but we do not address it here. Different objects can have different BRDFs, and are treated independently. Finally, every operation is performed 3 times, once for each color channel. Our final scenes show a variety of materials, with complex multi-term BRDFs, color variations, and texturing.

We now make the important assumption that any BRDF can be expressed as a linear combination of J basis functions, b_j:

$$\rho(\omega_i,\omega_o)=\sum_{j=1}^{J} c_j b_j(\omega_i,\omega_o). \qquad (2)$$

While it is always the case that such basis functions[3] can be found, our task will be to discover b_j's that are intuitive and expressive for editing, accurate for general BRDFs with a small number of terms, and allow for efficient precomputation and rendering.

As in standard precomputation techniques, we define the equivalent of a transport function, L', as the product of all the fixed data. Instead of transport, this quantity captures 'local irradiance':

$$L'(x,\omega_i,\omega_o)=L(x,\omega_i)V(x,\omega_i)S(\omega_i,\omega_o). \qquad (3)$$

[1] We use environment maps as a convenient source of complex lighting for our sample scenes, but the mathematical development and our rendering system can handle arbitrary local lighting as well.

[2] Since this is simply a summation, they can also be edited simultaneously — such as modifying their relative weights to conserve energy.

[3] We use the term 'basis functions' loosely. We only need a set of functions whose linear combination approximates the BRDF. Specifically, orthogonality is not required by our 'basis functions'.

We use the name L' to suggest that it is a modified version of the lighting (specific to each location's visibility, normal, and static function). Substituting Equations 2 and 3 in Equation 1, and pulling values not dependent on ω_i (i.e. c_j) outside the integral,

$$R(x,\omega_o)=\sum_j c_j \int_{\Omega_{4\pi}} L'(x,\omega_i,\omega_o) b_j(\omega_i,\omega_o) d\omega_i. \quad (4)$$

Using the fixed viewpoint assumption ($\omega_o=\omega_o(x)$), we define the light transport coefficients $T_j(x)$, to be

$$T_j(x)=\int_{\Omega_{4\pi}} L'(x,\omega_i,\omega_o(x)) b_j(\omega_i,\omega_o(x)) d\omega_i. \quad (5)$$

Substituting Equation 5 into Equation 4 allows us to express the outgoing radiance at a point x, in the direction $\omega_o(x)$, as the following dot-product:

$$R(x,\omega_o(x))=\sum_j c_j T_j(x)=\mathbf{c}\bullet\mathbf{T}(x). \quad (6)$$

Equation 6 has the same form as precomputed relighting methods, except that we use BRDF coefficients c_j, instead of lighting coefficients. The user will manipulate the BRDF, either by editing curves (section 4) which directly specify **c**, or by varying analytic parameters, which the system uses to compute the coefficient vector, **c**. The dot-product in Equation 6 is then performed at each location x, corresponding to each pixel in the image.

We solve three important challenges to successfully and efficiently implement this basic method:

Representation: While it is theoretically possible to represent BRDFs using a basis like spherical harmonics or wavelets [Lalonde and Fournier 1997], many terms are needed since BRDFs are 3D or 4D functions. Moreover, the coefficients in these representations are not intuitively editable — the same applies to weights of measured BRDFs [Matusik et al. 2003].

It may seem like parametric BRDFs are easier to deal with. Indeed, we allow the user to edit analytic BRDFs in the standard way by adjusting parameters of the model. However, these parameters cannot be used directly in the rendering system due to their complex and non-linear influence, while the linearity in the dot-product of Equation 6 is critical for real-time rendering in complex lighting.

In section 4, we introduce our use of a curve-based representation that is general and accurate for a wide class of BRDFs, including most parametric models as well as measured/factored BRDF representations. Moreover, since the space of edits for one of these curves is 1D, editing is intuitive and the computations are efficient.

Precomputation: Assume we use Equation 5 directly for precomputing **T**. At each pixel, there are J coefficients, T_j, and M lights, so the time complexity will be $O(JM)$. In section 5, we develop an efficient $O(M)$ algorithm (which is the same order as rendering a single image.) Moreover, we introduce robust methods to consider the area of sampled regions in an environment map, as opposed to simply concentrating the directionally distributed energy into point sources, enabling true all-frequency effects.

Rendering: Standard wavelet methods for computing Equation 6 make aggressive approximations that can impact quality. In an editing session, typically only local changes are made to the BRDF at each frame. We exploit this by developing an incremental wavelet rendering algorithm in section 6 that is high-quality and efficient. By exploiting the similarity between frames, we continuously improve on the previous frame, instead of making an independent approximation at each time-step.

4 BRDF Representation

It has long been known that re-parameterizing a BRDF can be useful for accurate representation, such as with the half-difference parameterization of Rusinkiewicz [1998b]. For example, the (isotropic) specular highlight is best represented as a 1D function of the half-angle, θ_h, while the Fresnel effect is best characterized as a 1D function of the difference-angle, θ_d. As most interesting BRDF effects are one-dimensional (for the correct parameterization), we propose the following representation for a BRDF, ρ:

	model	γ_A	γ_B	θ_{out}	Eq.	Analytic / Measured
CT	Cook-Torrance	θ_h	θ_d	no	10	A
AS	Ashikhmin-Shirley	γ_α	γ_β	no	15-16	A
HF	McCool et al. 2001	θ_h	θ_{in}	yes	17	M
LV	Light-View	θ_{in}	NA	yes	18	A/M
HD	Half-Diff	θ_h	θ_d	no	18	M

Table 1: The Models. Each model is shown with the abbreviations for it as used throughout the paper. The parameterizations (Equation 9) used to edit the model are also listed. In addition, any model can have an explicit dependence on $f(\theta_o)$, and we indicate which ones use this ability.

$$\rho(\omega_i,\omega_o)=\rho_q(\omega_i,\omega_o) f(\gamma(\omega_i,\omega_o)). \quad (7)$$

Here, $\gamma(\omega_i,\omega_o)$ is a 1D parameterization of the light and view directions, $f(\gamma)$ is the editable 1D factor (curve), and ρ_q is the 4D quotient BRDF — the static part that remains after dividing out the curve $f(\gamma)$. Equation 7 is an equality, due to the arbitrary complexity allowed in the (fixed) quotient BRDF, ρ_q. Figure 2 shows examples of the $f(\gamma)$ curves for various parameterizations.

We place the static factor ρ_q, in $S(\omega_i,\omega_o)$ from Equation 1, thus rewriting Equation 2 as:

$$f(\gamma)=\sum_{j=1}^{J} c_j b_j(\gamma). \quad (8)$$

As in previous work on all-frequency relighting, we choose (now 1D) wavelets as basis functions $b_j(\gamma)$. Unlike for relighting, the BRDF is visualized directly whenever small lights are present. Therefore, we use a smoother Daubechies 4 [1988] basis (see Appendix B), instead of Haar. An immediate advantage of the curve representation is that we only need to represent a 1D function $f(\gamma)$, and therefore do not need many basis functions. We often use J=256, which gives us a resolution of a fraction of a degree, and is enough to accurately represent very sharp specularities. Hence, we have written Equation 8 as an equality, not an approximation, since the discretization error is negligible.

As a simple example of how our representation can be applied to an analytic model, consider the original Phong model, $\cos^n(\theta_r)$. The parameterization $\gamma(\omega_i,\omega_o)=\theta_r$ is the angle between the reflected light and the view direction. The curve $f(\gamma)=\cos^n\gamma$, is controlled by the 'parameter' n. To avoid confusion with the curve parameterization, we will refer to an analytic parameter, such as the Phong exponent, as a *user-controlled variable* or UCV.

For analytic BRDFs like the Phong model, our 1D curves can be seen as an intermediate representation between editing non-linear parameters (e.g. n), and rendering with linear basis functions. For editing, the user adjusts a slider or otherwise specifies n in the standard way. Our system computes the curve $f(\gamma)=\cos^n\gamma$, every time the user edits the UCV, and then projects it onto the wavelet basis, for use by the rendering system. While the curve need never be directly exposed to the user for analytic BRDFs, it provides an intuitive interface when more control is desired. For example, if the user wants to change the falloff of the highlight from a cosine lobe to some other form (like Gaussian), she can edit $f(\gamma)$ directly. For measured BRDFs, there are no UCVs to expose to the user, and the 1D curves form an intuitive representation for direct editing.

Sometimes, a single 1D parameterization γ is not enough to capture all the effects one wants to edit, such as editing both the Fresnel term, and the specular highlight. In section 5.3, we will show how our system can accept BRDFs with two editable factors,

$$\rho(\omega_i,\omega_o)=\rho_q(\omega_i,\omega_o) f_A(\gamma_A(\omega_i,\omega_o)) f_B(\gamma_B(\omega_i,\omega_o)), \quad (9)$$

each with a different parameterization.

The rest of this section, and Table 1, deal with specific analytic and measured/factored BRDFs, showing how they can be expressed in the form of Equation 9.

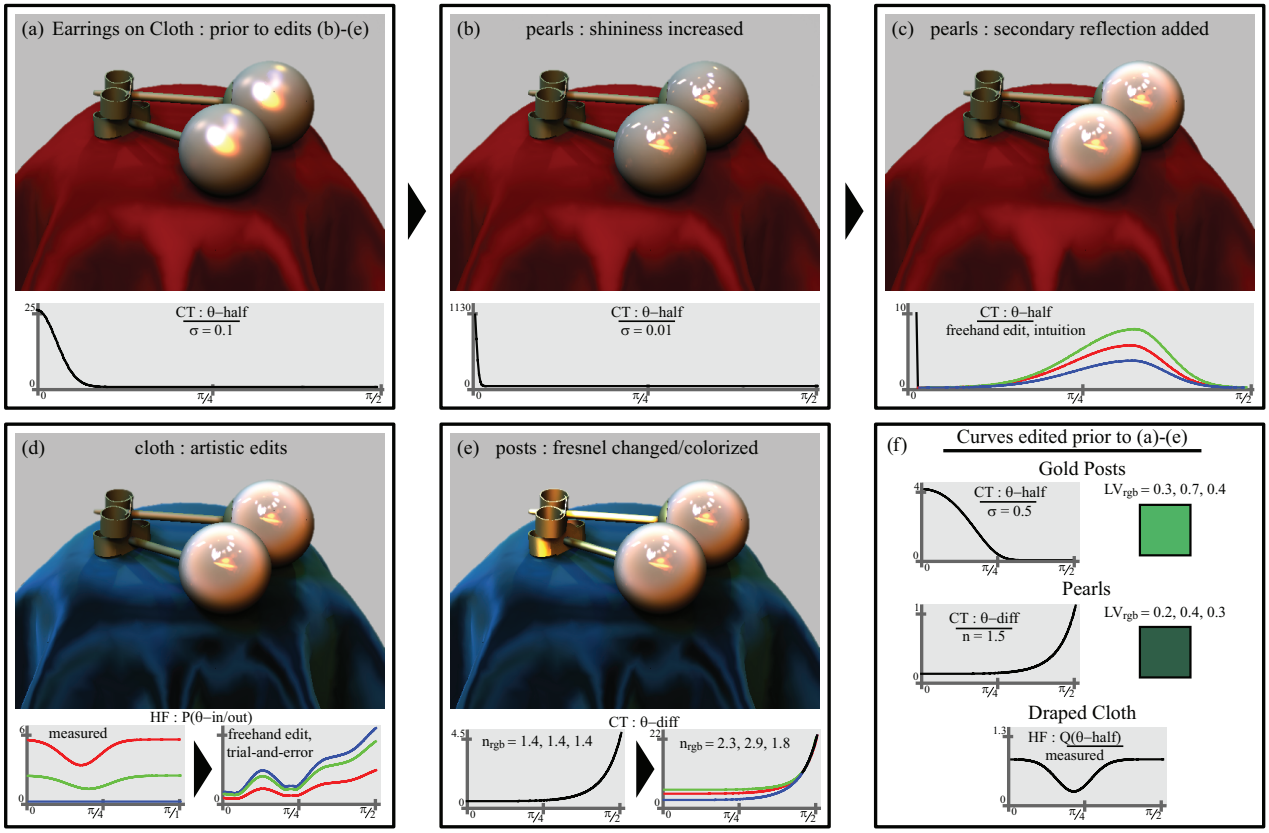


Figure 2. Earrings. A sample editing session shows before (a) and after (e) of a scene with pearl earrings on a cloth draped over a pedestal, as illuminated in Grace Cathedral. The pearls and posts use a Cook-Torrance specular term + LV diffuse term, and the cloth uses homomorphic factorization. The session begins by setting some initial values for the UCVs (f), and loading data for red velvet, based on the factorization presented in [McCool et al. 2001]. First (b), the pearls are given sharper reflections by decreasing σ, which in turn defines the $f(\theta_h)$ curve. The reader is encouraged to compare this with a real pearl and notice that indeed, the reflections are near mirror-like. The 'hazy' effect of pearls comes from a secondary reflection, and this is added in (c) by adjusting the curve with a freehand edit. In (d), the user plays around with the shape of the $P(\theta_i / \theta_o)$ function, and arrives at a desirable blue cloth material. Finally (e), the index-of-refraction for the Fresnel term of the posts is set to give them a metallic gold appearance.

4.1 Cook-Torrance: an analytic BRDF example

Like many analytic BRDFs, the Cook-Torrance model [1982] is already in the desired form presented in Equation 9. The specular term of CT is:

$$\rho_{CT} = \frac{F_{n,e}(\theta_d)G(\omega_i,\omega_o)D_\sigma(\theta_h)}{4\pi(\omega_i \cdot N)(\omega_o \cdot N)}, \qquad (10)$$

where N is the surface normal. The user-controlled variables are n (index of refraction), e (extinction coefficient), and σ (mean slope distribution). The first two UCVs control the shape of the Fresnel term, $F(\theta_d)$, while the last controls the shape of the slope distribution function, $D(\theta_h)$, often taken to be the Beckmann distribution. The geometric attenuation factor, $G(\omega_i,\omega_o)$, and the denominator, are fixed for all values of the UCVs, and form the quotient brdf, ρ_q, while $f_A = D(\theta_h)$ and $f_B = F(\theta_d)$. The corresponding parameterizations are: $\gamma_A = \theta_h(\omega_i,\omega_o)$ and $\gamma_B = \theta_d(\omega_i,\omega_o)$.

Figure 2(a,b) shows how a user's choice of σ changes the BRDF of the pearls. At each frame, a slider (not shown) is used to specify the value for σ, which is then used to generate the displayed curve, $f_A(\theta_h)$ (via explicit evaluation of the Beckmann function at 256 points along the curve). As shown in Figure 2(c), one may also wish to perform a freehand edit on the curve to explicitly control the appearance, such as the specular behavior beyond the initial peak.

When the user changes the index-of-refraction (n) or the extinction coefficient (e), the new value is used to generate $f_B(\theta_d)$ (by evaluating the Fresnel equation). Figure 2(e) shows an example of changing the index-of-refraction in the Fresnel term of the posts.

4.2 Ashikhmin-Shirley: an anisotropic analytic BRDF

Our framework can be applied to anisotropic analytic models, such as Ashikhmin-Shirley [2000]. We will use this model to demonstrate a step-by-step separation of an analytic BRDF into the form required by Equation 9.

We begin with the specular component of the AS BRDF, as it appears in their paper, with minor notational adjustments:

$$\rho_{AS} = \frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi} \frac{(\cos\theta_h)^{n_u \cos^2\phi_h + n_v \sin^2\phi_h}}{\theta_d \max(\cos\theta_i, \cos\theta_v)} F(\theta_d). \qquad (11)$$

The first step is to identify the UCVs of the model: n_u and n_v. These are similar to the exponent in the Phong model, but having two controls allows for anisotropic highlights. Next, we find the smallest subexpression that contains all instances of all UCVs:

$$\sqrt{n_u+1}\sqrt{n_v+1}(\cos\theta_h)^{\cos^2\phi_h n_u + \sin^2\phi_h n_v}. \qquad (12)$$

The rest of the BRDF becomes ρ_q. Then, we try to factor this expression into f_A and f_B, each of which is defined for some 1D parameterizations, γ_A and γ_B. Note that once we factor Equation 12, the same UCV cannot appear in both factors. A simple identity for exponents allows us to do exactly that (we name them α and β):

$$\alpha = f_A = \sqrt{(n_u+1)}(\cos(\theta_h))^{n_u \cos^2\phi_h} \qquad (13)$$

$$\beta = f_B = \sqrt{(n_v+1)}(\cos(\theta_h))^{n_v \sin^2\phi_h}. \qquad (14)$$

The final step is to identify the 1D parameterization of all angle-dependent values. We must find an expression that does not involve the user-controlled variables, so we eventually obtain

$$\gamma_\alpha = \cos^{-1}\left((\cos\theta_h)^{\cos^2\phi_h}\right);\ \alpha(\gamma_\alpha) = \sqrt{n_u+1}\cos^{n_u}\gamma_\alpha \qquad (15)$$

$$\gamma_\beta = \cos^{-1}\left((\cos\theta_h)^{\sin^2\phi_h}\right);\ \beta(\gamma_\beta) = \sqrt{n_v+1}\cos^{n_v}\gamma_\beta \qquad (16)$$

The inverse-cosine was added because it is useful to think of the γ's as angles that range from 0 to $\pi/2$, but is not strictly necessary. The teapot in Figures 1 and 6 uses this Ashikhmin-Shirley BRDF.

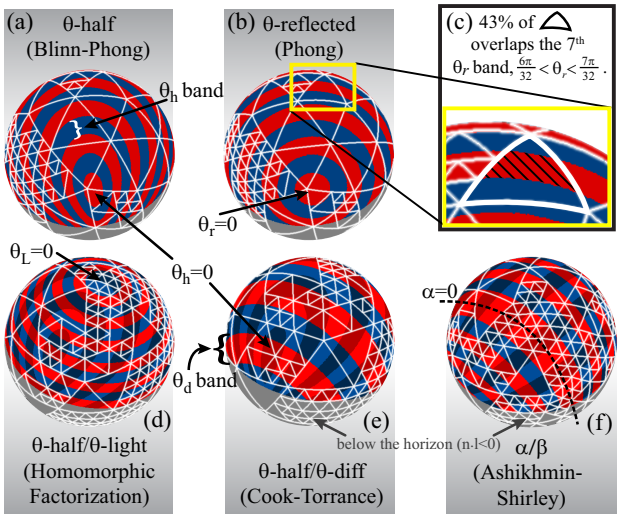


Figure 3. Bands and Overlapping Area Lights.. The box functions of the BRDF parameterization create bands when visualized on the sphere of incoming directions. The triangular lights that approximate the environment are also shown to illustrate the overlaps that must be computed in Equations 21 and 22. The orientation of the bands depends on the image pixel (surface normal and view).

4.3 Other Analytic BRDFs

The anisotropic Ward model [1992] can be handled in essentially the same way as AS, above. For most analytic BRDFs, the factors, parameterizations, and UCVs can be found by inspection. Examples with half angle and difference angle, $f_A(\theta_h)$ and $f_B(\theta_d)$, are Beard-Maxwell [Maxwell et al. 1973] and Schlick [1994].

In the Oren-Nayar model [1994], the dependence of terms on the canonical directions, ω_i and ω_o is static, though quite complicated. The UCVs serve to define a scalar multiplier for these static terms. While our system would not be needed for a model such as Oren-Nayar, it serves to demonstrate that per-light/per-pixel evaluation of a BRDF can be arbitrarily expensive. An advantage of our method, is the decoupling of the analytic complexity from the rendering algorithm. In addition, secondary 'freehand' operations could be applied directly to curves representing an O-N BRDF.

In rare cases where the analytic form is not amenable to curve-based decomposition, we can edit a factored form of the BRDF, as discussed for measured data, below.

4.4 Measured/Factored BRDFs

Besides analytic BRDF models, we seek to edit measured reflectance data. Isotropic factored BRDFs can be manipulated, by editing curves corresponding to each of the factors. We consider editing the homomorphic factorization (HF) of McCool et al. [2001]. Specifically, the factorization is:

$$\rho_{HF} \simeq P(\omega_i)Q(\omega_h)P(\omega_o) = \rho_q P(\theta_i)Q(\theta_h)P(\theta_o), \qquad (17)$$

where for isotropic BRDFs, we are really editing 1D curves P and Q. (If the 2D terms are not symmetric because of noise or mild anisotropy, we can absorb that in the quotient, ρ_q). The outgoing angle is fixed at each pixel and is not part of the integrand in Equation 1. Therefore, $P(\theta_o)$ simply requires a per-pixel multiplication of the final result. As indicated in Table 1, any model can add this $f(\theta_o)$ dependence. Hence, we have the form required by Equation 9, with $f_A(\gamma_A)=Q(\theta_h)$ and $f_B(\gamma_B)=P(\theta_i)$. Figure 2 uses [McCool et al. 2001]'s data for red velvet (factored from the CURET database [Dana et al. 1999]), and shows an edit of the P term (part d).

Finally, we consider measured reflectance, as in the database of Matusik et al. [2003]. One possible factorization is the above approach. Since most BRDFs contain diffuse-like and specular-like lobes, we use the following alternative factorization

$$\rho = \rho_d + \rho_s \;;\; \rho_d = \rho_{qd} G_d(\theta_i) H_d(\theta_o) \;;\; \rho_s = \rho_{qs} G_s(\theta_d) H_s(\theta_h), \qquad (18)$$

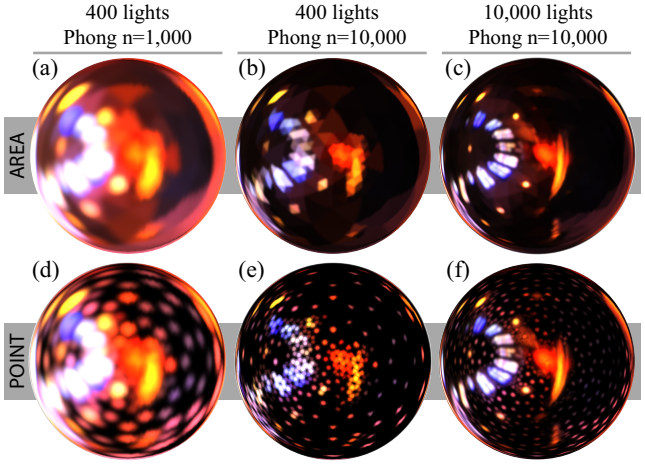


Figure 4. Point vs. Area Lights. A comparison of reducing the environment to point lights (d-f), versus preserving the solid angles and representing the environment as triangular area lights (a-c).

where the diffuse-like lobe, ρ_d, is expressed in the incident-outgoing parameterization, and the specular-like lobe, ρ_s, uses the half-angle and difference-angle. The net BRDF is the sum of the two terms, each treated independently. The quotient BRDF is used to make up for the error of factoring. There are a several ways to compute these factorizations, and we use a non-negative factorization as in [Lawrence et al. 2004, 2006]. The handles in Figure 6 are an example of captured nickel from the Matusik database.

5 Precomputation

We now consider the precomputation step, described by Equation 5. It will be convenient to treat each pixel separately in what follows, so we can drop the dependence on spatial location x. To emphasize this, we write $L'(x,\omega_i,\omega_o(x))=L'(\omega_i)$, where both x and $\omega_o(x)$ are fixed for a given pixel. We now make use of the 1D BRDF curve representation from Equations 7 and 8, and rewrite 5:

$$T_j = \int_{\Omega_{4\pi}} L'(\omega_i) b_j(\gamma(\omega_i)) d\omega_i. \qquad (19)$$

Equation 19 takes the form of a projection. The lighting function, L', is being projected onto the jth basis function. The 1D wavelet basis functions b_j, that are used for rendering, are better replaced with box functions for precomputation, since this makes the projection sparser and more efficient. It is a simple matter of using the fast wavelet transform to convert box coefficients to Daubechies4 coefficients at the end of precomputation, before rendering. Figure 3 shows examples of the bands formed by these box functions, when visualized on the sphere of incoming directions.

5.1 Initial Approach

We now describe two simple ideas which seem like a straightforward solution, but turn out to be inappropriate for BRDF editing.

In standard PRT, Equation 19 simply involves a spherical harmonic or wavelet transform, where projection onto the basis is a well-studied operation. In our case, the box functions b_j use a different parameterization than the lighting, with γ implicitly depending on the local view direction, and therefore *changing* for each pixel. As seen in Figure 3, the box functions form highly irregular projections onto the sphere of lighting directions. To address this issue, one might imagine making a change of variables so that the lighting and the integral are also specified over the domain of γ, where b_j is a well-behaved box-function:

$$T_j = \int_0^1 L'(\gamma^{-1}(\omega_i)) b_j(\gamma) \left| Jac(\gamma^{-1}) \right| d\gamma, \qquad \text{(Initial.1)}$$

where Jac is the Jacobian necessary for a multi-dimensional change-of-variables. It turns out that most parameterizations, γ, are not invertible, so Initial.1 cannot be evaluated analytically. Moreover, we do not want be restricted to a subclass of mappings that are

invertible, and whose Jacobian can be written explicitly. Hence, we discretize Equation 19 directly, without reparameterizing.

The straightforward approach for discretization is to reduce the continuous environment map into point samples. We can select a number of incoming directions, such as with structured importance sampling [Agarwal et al. 2003], and replace the environment map with a light for each of these directions. The radiance of the solid angle around each light is preintegrated, and attributed to the directional light, thus allowing it to act as proxy for the illumination of its neighboring directions. Equation 19 now becomes:

$$T_j = \sum_{m=1}^{M} L'(\omega_m)\Delta_{m,j}, \tag{21}$$

where the environment is transformed into M discrete light sources at locations ω_m, and $\Delta_{m,j}$ is defined here, but redefined in 22:

$$\Delta_{m,j} = b_j(\gamma(\omega_m)), \tag{Initial.2}$$

meaning that the energy of light m contributes to T_j if its location (center) falls inside band j.

Figure 4 shows the shortcoming of (Initial.2). The difference between area-based illumination, and point samples is obvious for highly specular materials, such as the Phong sphere in Figure 4(d-f). Furthermore, the concentration of energy from large areas results in abnormally bright points where the environment is relatively dim (compare Figure 4(a vs. d)). A proposed solution is simply to increase the number of samples, but as seen in Figure 4(f), even 10,000 lights results in noticeable artifacts. We conclude that point lights are only viable for BRDFs that blur the environment with a kernel larger than the distance between any two such points.

5.2 Our Algorithm

The problem with point-sampled environment maps for the lighting is that, in editing BRDFs, we care about the angular distribution of energy as much as its strength. In order to retain this information, we use a basis of spherical triangles to approximate the lighting. Similar to the scheme used in Spherical Q2-Trees [Wan et al. 2005], we hierarchically subdivide the sphere into polygonal domains, breaking each one into four children as required by the metric in [Agarwal et al. 2003]. However, we use triangles instead of rectangles, because they are more efficient, and also amenable to the optimized overlap test described in Appendix A.

Figure 3 shows how these triangle lights overlap several common parameterizations. We still evaluate the visibility, V, and quotient BRDF, S, at the light centers, as in 21. However, we use a more sophisticated technique for calculating the overlap (projection) between a given light and the bands created by the box functions of γ, as shown in Figure 3(c). Hence, we redefine $\Delta_{m,j}$ from 21:

$$\Delta_{m,j} = \frac{1}{\Omega_m}\int_{\Omega_m} b_j(\omega)d\omega, \tag{22}$$

where Ω_m is the solid angle preintegrated for light m, and the integral in Equation 22 sums to 1 when light m falls entirely within band j, and a fraction thereof otherwise. This overlap can be computed in constant time, as detailed in Appendix A.

By accurately computing angular overlaps, we are able to use fewer light samples to speed up precomputation. The final images are accurate up to the variation lost in averaging the radiance over the area of the triangle, as seen in Figure 4(b vs. c). This should be contrasted with the precomputation in standard PRT relighting, which must consider all (usually about 25,000) lighting directions.

Efficiency: A direct implementation of Equation 21 at each pixel loops over all lights m, for each j. The cost is therefore $O(JM)$, as it would be for most PRT relighting algorithms. We can reduce this to time linear in the number of light samples by recognizing that a triangle light m, overlaps (on average) a small number of box functions j. Therefore, we reorder the computation to first loop over lights m, and then over coefficients j that overlap m. (This inner loop now has essentially constant complexity). Moreover, we now need only compute visibility (by ray-tracing) once in the outer loop. Pseudocode for the optimized algorithm is shown in Code 1. This reduces the complexity to $O(M)$ at each pixel.

```
procedure computeT(pixel p, parameterization γ):
for m = 1 to M do {                    //loop over all lights
    V = raytrace(p, ω_m);
    if (V == blocked) continue;
    S = ρ_q(p, ω_m)*(N•ω_m);    //quotient BRDF *cos θ_i
    L' = V*S*preintegratedRadiance(m);          (3)
    minBand = firstOverlap(ω_m,γ, p);
    maxBand = lastOverlap(ω_m,γ, p);
    //loop over relevant bands (usually only 2 or 3)
    for j = minBand to maxBand do {
        Δ_{m,j} = fractionalOverlap(m, j, γ, p);  (22)
        T[j] += L'*Δ_{m,j};                       (21)
    }
}
```

Code 1. Precomputation Algorithm. We find accurate overlaps between triangle lights and bands in O(M) time.

5.3 Two Curve BRDFs

We now detail the extension to two different curves and parameterizations for editing. We use the fact that a user can only edit one curve at a time, making the other curve temporarily 'dormant'. Each curve or factor is approximated with a set of box functions. Using a 2-factor form of Equation 2 through the derivations in Section 3, we arrive at a quadratic form of Equation 6,

$$R = \sum_j c_j^A \sum_k c_k^B U_{jk} = (\mathbf{c}^B)^T \mathbf{U}\mathbf{c}^A, \tag{23}$$

where $\mathbf{c}^A$ and $\mathbf{c}^B$ are the coefficient vectors for the two curves, f_A and f_B , as in Equation 9. The vector $\mathbf{T}$ has been replaced with the matrix $\mathbf{U}$, defined by the 2D form of Equation 19:

$$U_{jk} = \int_{\Omega_{4\pi}} L'(\omega_i) b_j(\gamma_A(\omega_i)) b_k(\gamma_B(\omega_i)) d\omega_i. \tag{24}$$

Recalling that Equation 24 takes the form of a projection, we note that $\mathbf{U}$ is a projection of the lighting onto the (non-orthogonal) axes of γ_A and γ_B[4]. The discrete form is analogous to Equation 21,

$$U_{jk} = \sum_{m=1}^{M} L'(\omega_m)(\Delta_{m,j}\Delta_{m,k}) \tag{25}$$

As in the one-curve case, we loop first over all the lights, and then find the overlapping bands for each parameterization.

The quadratic form in Equation 23 cannot currently be evaluated directly in real-time. Instead, either the left or the right matrix-vector multiplication is evaluated in a run-time precomputation step we call *curve switching*. Let us say the user chooses curve A (Figure 2(a):Pearls θ_h) . Using the (temporarily fixed) values of $\mathbf{c}^B$ for the 'dormant' curve (Figure 2(f):Pearls θ_d), we compute

$$\mathbf{T}_A = (\mathbf{c}^B)^T \mathbf{U} \quad ; \quad R = \mathbf{T}_A \bullet \mathbf{c}^A, \tag{26}$$

where $\mathbf{T}_A$ is the transport coefficient vector for standard 1D curve editing of curve A, given the fixed value of curve B. While not real-time, it takes only a few seconds to compute $\mathbf{T}_A$ (shown in Table 3), making it a viable part of an interactive editing session. The second part of Equation 26 to compute the actual image is the standard rendering dot-product, and one can now edit the first curve with real-time feedback. At some point, the user decides to *switch curves,* fixing $\mathbf{c}^A$ while editing $\mathbf{c}^B$. The system now computes $\mathbf{T}_B = \mathbf{U}\mathbf{c}^A$, which again takes only a few seconds. (See Appendix B for proper wavelet transformations of $\mathbf{U}$ and the $\mathbf{c}$'s.)

6 Incremental Wavelet Rendering

We now know how to obtain the transport vector $\mathbf{T}$, and vector of function coefficients $\mathbf{c}$, required by Equation 6. In this section, we introduce a new rendering algorithm that is generally applicable to any precomputation method that takes the form of Equation 6.

[4] For the case of $\gamma_A = \theta_i$ and $\gamma_B = \phi_i$, $\mathbf{U}$ is the lighting, modulated by V and S.

T =(3, 5, 2)	Incremental Non-Linear						Non-Linear	
t	$c_{perfect}$	R_{perf}	c_{diff}	$\tilde{c}_{diff}$	$T \bullet \tilde{c}_{diff}$	R_{effec}	$c_{effective}$	$\tilde{c}_{perf}$	$T \bullet \tilde{c}_{perf}$
0	0 0 0	0	0 0 0	0 0 0	0	0	0 0 0	0 0 0	0
1	4 2 1	24	4 2 1	4 0 0	12	12	4 0 0	4 0 0	12
2	4 3 1	29	0 3 1	0 3 0	15	27	4 3 0	4 0 0	12
3	4 2 0	22	0 -1 0	0 -1 0	-5	22	4 2 0	4 0 0	12

Table 2: Didactic Example of Incremental vs. Non-Linear. In this example, the monochromatic color of a single pixel (with **T** shown in the upper left) is computed at 4 consecutive frames. For instructive purposes, we use the original box functions, so that the coefficients shown can be used to directly envision the curve. We assume that only 1 multiplication is possible per frame, though in real scenarios, 20-30 wavelet terms can be multiplied at each frame. First, we initialize all values to **0**. At t=1, the user loads in a curve with geometric decay: (4, 2, 1). Next (t=2), the user amplifies the center of the curve. Finally (t=3), the user decides to shift all but the left part of the curve down by 1 unit. At each frame, Incremental is able to move closer to the perfect result by concentrating all of the computational budget on the **difference** in the user's input. By contrast, the standard Non-Linear approximation always concentrates on the largest coefficient.

The number of elements in the dot-product $\mathbf{T}\bullet\mathbf{c}$, can be quite large, and when performed at each pixel, modern CPUs may not be able to render this in real-time. A common solution is to transform both vectors into a wavelet basis, and then use a non-linear wavelet approximation that makes these vectors sparse,

$$R^i_{perfect} = \mathbf{T}\bullet\mathbf{c}^i_{perfect} \quad (27)$$

$$R^i_{effective} = \mathbf{T}\bullet\mathbf{c}^i_{effective}, \quad (28)$$

where $\mathbf{c}^i_{effective}$ is the representation of $\mathbf{c}^i_{perfect}$ after approximation, and $R^i_{effective}$ is the effective (approximate) image displayed.

This method has two drawbacks for us. First, we cannot compress as aggressively as relighting methods, which never directly visualize the lighting[5]. We need to keep many more terms in $\mathbf{c}^i_{effective}$, since both lighting and BRDF retain all frequencies. Second, the standard non-linear wavelet approximation, by definition ignores small coefficients, often giving the user no feedback when small adjustments are made to the BRDF.

We note that $\mathbf{c}$ tends to change by small amounts between frames, since the user is continuously editing the 1D curve that defines it. We take advantage of this by rendering the current frame i, using the change $\mathbf{c}^i_{diff}$ relative to the previous frame i–1,

$$\mathbf{c}^i_{diff} = \mathbf{c}^i_{perfect} - \mathbf{c}^{i-1}_{effective}. \quad (29)$$

The $\mathbf{c}$'s of consecutive frames tend to be more similar when the b_j's of Equation 8 are a wavelet basis.

In interactive editing, one is usually making local modifications to the curve, so $\mathbf{c}^i_{diff}$ has only a few nonzero wavelet coefficients, even when $\mathbf{c}^i_{perfect}$ would need many terms for accurate approximation. We must also consider occasional cases where $\mathbf{c}^i_{diff}$ has more nonzero wavelet terms than we can afford to use in a single frame. This happens when the user makes drastic changes that affect the entire curve (such as a uniform scaling). To handle these situations within our framework, we use a standard non-linear wavelet approximation, $\tilde{\mathbf{c}}^i_{diff}$, of $\mathbf{c}^i_{diff}$. For most frames, $\tilde{\mathbf{c}}^i_{diff}$ will be equal to $\mathbf{c}^i_{diff}$, but allowing for approximation makes our system robust when large changes occur (or if the wavelet budget is very small). Our final equations for incremental rendering then become:

$$R^i_{effective} = R^{i-1}_{effective} + \mathbf{T}\bullet\tilde{\mathbf{c}}^i_{diff} \quad (30)$$

$$\mathbf{c}^i_{effective} = \mathbf{c}^{i-1}_{effective} + \tilde{\mathbf{c}}^i_{diff}, \quad (31)$$

where we make explicit that the final radiance, $R^i_{effective}$ may not be exact. Equation 31 shows how we keep track of the $\mathbf{c}^i_{effective}$, so that c^{i+1}_{diff} can be calculated in the next frame, per Equation 29. Note that the expensive operation here is the (per-pixel) dot-product ($\mathbf{T}\bullet\tilde{\mathbf{c}}_{diff}$) just as in standard (non-incremental) PRT rendering.

Even though we never explicitly calculate the dot-product in Equation 28, our incremental updates in Equation 30 *effectively* give the same result. The main difference is that our $\mathbf{c}_{effective}$ are not approximated with a predetermined budget, but rather can get arbitrarily close (and often equal) to $\mathbf{c}_{perfect}$. Table 2 uses Equations 29-31 to compute successive frames of a didactic editing session.

Incremental wavelet rendering addresses the issues mentioned at the start of this section. Since work is focused only in the local region where the user is changing the BRDF, our system almost always renders reference-quality images. In rare cases, when the region of change is large, and cannot be exactly captured in a single incremental update, our system automatically continues to refine the image using Equations 30 and 31 until the $\mathbf{c}_{effective}$ equals $\mathbf{c}_{perfect}$.

Figure 5(a) compares standard and incremental curve approximations for a typical BRDF edit. Figure 5(b,c) compares the image quality. The user's input in Figure 5(a) is better approximated by our method because the energy in the peak on the left (more easily seen in Figure 2b) is static, leaving a full budget of wavelet coefficients to approximate the secondary rise. Note that Standard Non-Linear is negative in some regions, leading to a darkening of the diffuse term near the center of the pearls, as seen in Figure 5(c).

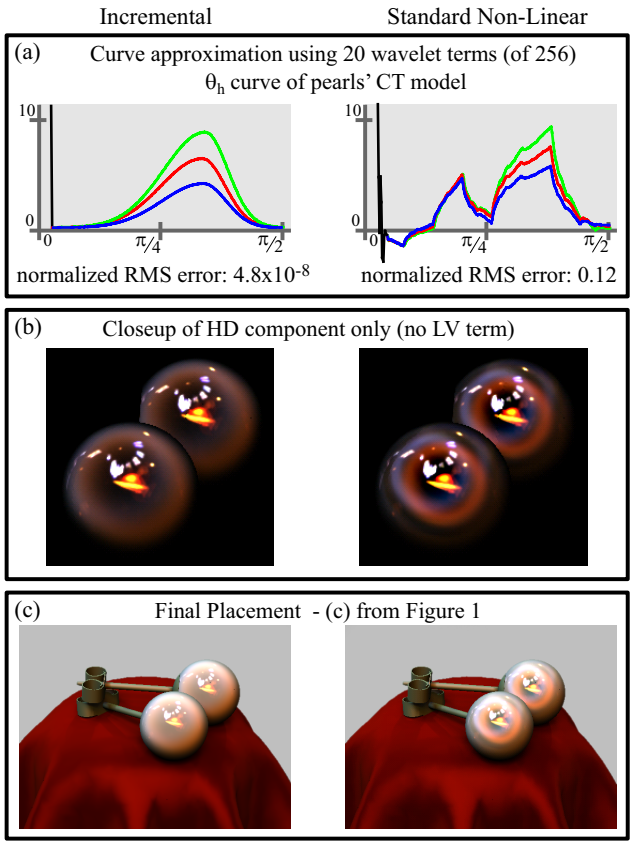


Figure 5. Incremental Wavelet Rendering. Our incremental method is compared to standard non-linear wavelet approximation. Non-linear uses most of its wavelet budget for the large peak on the left (visible more clearly in Figure 2b), and poorly approximates the colored rise on the right. The incremental images are nearly identical to the reference image.

[5] Relighting implicitly assumes the BRDF is lower frequency than the lighting (no near-mirror), thus being more forgiving of lighting approximations.

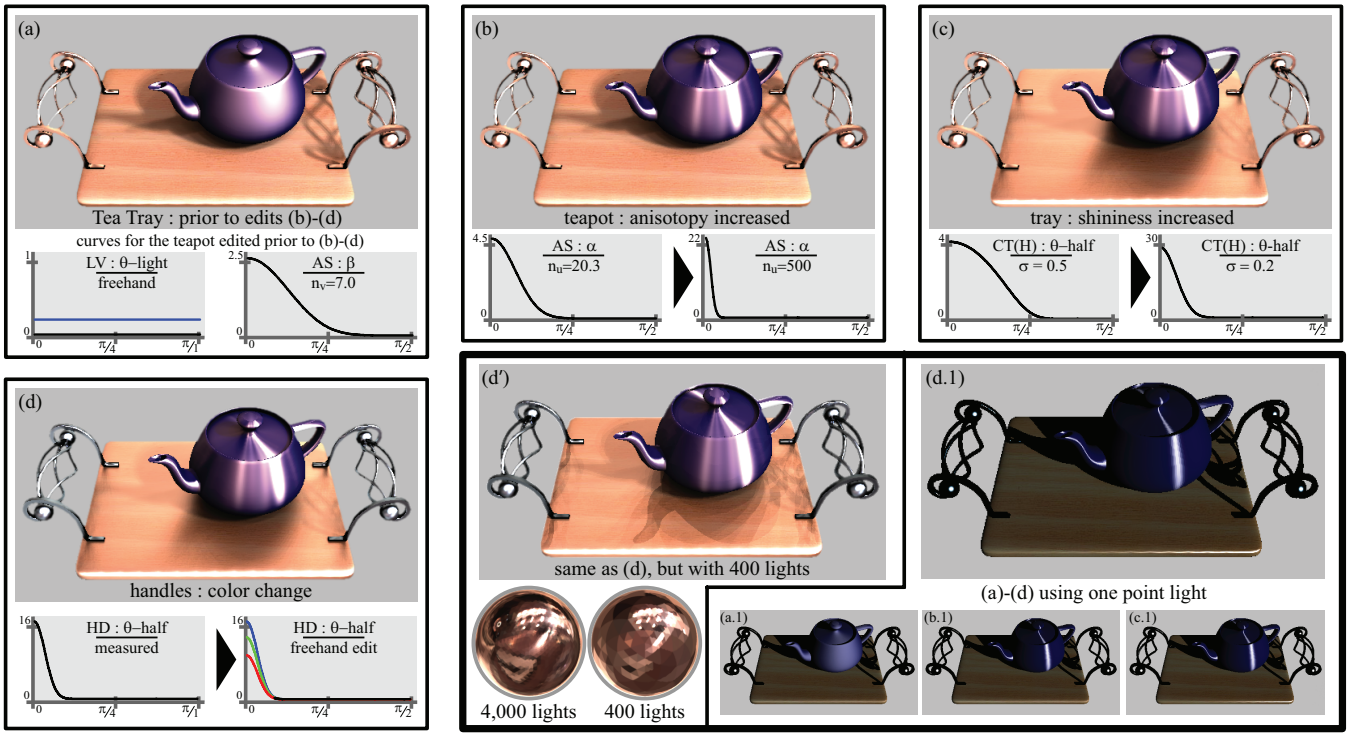


Figure 6. Teatray. The Utah teapot on a wooden tray *before* (a) and *after* (d) an editing session with our system. The scene starts with a teapot rendered using an Ashikhmin-Shirley specular term + LV diffuse term on a tray with a Cook-Torrance specular term + LV with wood-grain texture term. The handles use the HD parameterization with nickel data from the Matusik database. The illumination is a 4,000 light approximation of the Galileo HDR probe. After setting initial values for the teapot (a), the user switches from editing the β curve, to editing the α curve of the AS model. The anisotropy is increased (b), giving a brushed look. Next (c), the tray's specular term is made sharper. For a planar surface, the most noticeable effect of this edit is a change in the shadows. Now that the diffuse term is less prominent (as evidenced by the washed out texture), the shadows caused by lights in the reflection direction are more visible than those in the incident direction. Finally, the user notices that even though the data loaded for the handles is that of nickel, they appear golden due to the overall color of the environment. To regain the feel of a silvery metal, the user edits the left portion of the $f(\theta_h)$ curve with a freehand adjustment (d) to offset the coloration given by the environment. Each of these operations is compared under a point source (a.1-d.1), and it is clear that the feel of the final materials cannot be understood by making edits in such a different light setting. The strength and position of the point light were set by hand to match the most noticeable aspects of the environment. Finally, we compare (d) with the same image rendered using only 400 area lights (d′). The overall look and feel are preserved, with the most noticeable difference being a loss of softness in the shadows.

7 Results

We first show some BRDF edits that are possible in real-time, with complex lighting and shadows. We then evaluate the performance of our precomputation and rendering methods quantitatively.

7.1 Real-time Edits

While this paper focuses on the rendering algorithms, rather than specific editing techniques, our experience has been that one can quickly create a variety of effects and precisely choose material properties with our system — tasks that would often be difficult without final-placement, real-time feedback. Figures 2 and 6 show a number of edits using our system, that indicate some of the richness of effects that can be produced. We will refer to these as *Earrings* and *Teatray*, respectively.

Specular highlight adjustment: One basic edit, shown in Earrings(a-b) and in Teatray(c), is adjusting the width of a specular highlight, by modifying analytic parameters like surface roughness (σ), which in turn specifies the half-angle distribution $f(\theta_h)$. Compare the appearance of the specular BRDF in complex lighting for the handles of the Teatray(d) to the same BRDF in simple lighting (d.1). Complex lighting adds the richness of encompassing illumination that is lost under a point source. Note how the two main light sources blend together in Earrings(a), and are made distinguishable in Earrings(b). Finding the right σ for a desired level of interaction between different parts of the environment can only be accomplished with interactive feedback.

Anisotropic highlights: We can also edit anisotropic BRDFs like Ashikhmin-Shirley (see Teatray). When comparing the teapot in Teatray(b) to Teatray(b.1), we see that the point-lit teapot is much darker. This is not due to a dimmer light, but rather to the fact that the point light interacts only with surfaces oriented in the specular direction, relative to the viewer. By contrast, the full complex illumination affects the entire surface of the teapot. If we attempted to set n_u under a point source, we would be misled to believe that a larger value creates a darker appearance for the teapot.

Retroreflection and the Fresnel effect: Earrings(e) shows how changing the index of refraction at each color channel generates a new set of curves for $f(\theta_d)$, according to the Fresnel Equation. Values of $f(\theta_d)$ near θ_d=0 control retroreflection, since the view and light directions are close to each other. As seen in the separate RGB curves, the color distinction is large near this part of the curve, and dissipates as θ_d grows. This will give the desired Fresnel effect of desaturating the color of lights near grazing angles.

Interaction of Materials and Shadows: Shadows can often depend on the material properties, since a diffuse BRDF draws lighting energy from a different area than a glossy BRDF. In Teatray(c), we edit the tray's material from diffuse to glossy, which also allows us to create different shadowing effects. Notice that occluding geometry in the reflection direction is now much more important to the creation of shadows than in the diffuse case, Teatray(b).

Artistic edits: We can also go beyond physically-based BRDFs, to create flexible artistic effects. Earrings(c) shows how intuition about a secondary reflection can be used to edit a portion of a curve otherwise specified by the user's choice of σ. In Earrings(d), we used trial-and-error to manipulate a curve loaded from factored data in order to design a desirable material. A simpler type of artistic license is used on the handles in Teatray(d), where we compensate for the color of the environment to force the silver appearance.

object	pixels	model	bands	curve switching	storage (MB) raw	storage (MB) compressed	precompute(*minutes*) *4000 lts*	precompute(*minutes*) *400 lts*	visibility(*minutes*) *4000 lts*	visibility(*minutes*) *400 lts*	
Tray	40,529	LV	256	NA	121	36.3 (0.3)	1.5	0.2	*opt*	*standard*	
		CT(H)	256	NA	243	121.5 (0.5)	1.6	0.3	1.2	6.6	0.7
Teapot	17,709	LV	256	NA	53	15.9 (0.3)	0.7	0.1	0.5	2.8	0.3
		AS	128/128	2.1 *sec*	3,400	408 (0.1)	1.7	0.2			
Handles	8,629	HD	256/32	0.7 *sec*	330	60.2 (0.2)	0.8	0.1	0.4	1.4	0.2
Figure 6	**66,867**				4,147	**641.9 (0.15)**	**6.3** ***min***	0.9	**2.1** ***min***	11	1.2
Pearls	22,505	LV	256	NA	4	2.3 (0.6)	0.9	0.1	0.7	3.9	0.5
		CT	256/64	2.3 *sec*	1,464	248.9 (0.2)	2.6	0.3			
Posts	7,626	LV	256	NA	2	0.9 (0.5)	0.3	0.1	0.4	1.3	0.2
		CT	256/64	0.3 *sec*	92	18.4 (0.2)	0.8	0.2			
Cloth	98,890	HF	128/128	1.8 *sec*	1,186	260.9 (0.2)	6.3	0.8	3.2	20	2.1
Figure 2	**129,021**				2,748	**531.4 (0.19)**	**11** ***min***	1.5	**4.3** ***min***	24	2.8

Table 3. Timings and Memory. The precomputation time and memory overhead for each object in the two scenes, Earrings and Teatray. Images are rendered at 512x512, with a fixed view. Note that the compression ratios are somewhat low because we cannot afford to compress aggressively, as mentioned in Section 7.2. Also note that even though the memory overhead grows geometrically with the dimensionality (1D vs. 2D), the precomputation time only grows linearly. CT(H) refers to a fixed Fresnel factor for the Tray. The 4000 light visibilities were performed with a standard ray-tracer, and with optimizations as in [Ben-Artzi et al. 2006].

7.2 Precomputation

Table 3 analyzes the time and memory usage for the precomputation phase, for the Earring and Teatray scenes, itemized for each object. Since each pixel is treated separately, the totals are proportional to the number of pixels, and would be comparable for any scene with the same (512x512) image resolution. These tests were run on an Intel Xeon 3.2Ghz 64bit processor with 8GB of RAM.

We proceed from left to right in Table 3. For each object, we show the BRDF model used. In many cases we use a sum of diffuse and specular lobes, such as LV + CT for the posts, and LV + AS for the teapot. We typically use 128 or 256 bands to accurately represent specularities. In some cases, lower resolutions of 32 or 64 are used for θ_d in HD or CT representations. For two-curve models, the time to switch curves for editing is only one or two seconds.

We compress the transport coefficient vectors and matrices, by zeroing out values below a small threshold (10^{-6}) and not storing them explicitly. Unlike relighting, we must compress conservatively since the BRDF can be visualized directly. We achieve average compression rates of only 5:1 – 6:1 (sometimes only 2:1). Even so, our final storage requirements, while large (100s of MB), are comparable with all-frequency relighting PRT methods. Most of the storage is for two-curve BRDF models, where we must store transport matrices $\mathbf{U}$ at each pixel (for example, AS in teapot, CT in pearls, and HF in cloth). The storage for transport vectors from one-curve BRDF models like LV is usually much smaller.

We separate the precomputation time into visibility and our precomputation. The visibility is computed once for the scene, using a ray-tracer. For the 4,000 light approximations, we show timings with and without the acceleration provided by [Ben-Artzi et al. 2006]. The band-light overlap and fixed BRDF shading computations are performed as per Code 1. The time for both computations is approximately linear in the number of lights and pixels. For a reasonable 400 triangular light approximation, we require only 2-4 minutes, making quick previewing of a variety of different scenes possible. As shown in Figure 6(d′), our area-preserving method for precomputing with triangular lights allows a small number of lights to be used with only a small loss in the quality of rendered images. If precomputation time is not a bottleneck, higher quality approximations can be obtained in 10-15 minutes, with thousands of lights, as done for the figures in this paper.

7.3 Rendering

Rendering is real-time, at about 25fps, and with rapid interactive switching between curves. We typically use 30 wavelet coefficients for each frame. Our incremental rendering algorithm ensures essentially reference-quality images while editing, as seen in Figure 5. As with other precomputation methods, rendering time is independent of scene complexity. In addition, our method decouples rendering time from the lighting and shading complexity. Our incremental rendering technique further breaks dependence on even the BRDF resolution (number of bands).

8 Conclusions and Future Work

In this paper, we have addressed the novel problem of editing BRDFs in their final scene placement, with complex lighting and shadows. We have shown how to linearize analytic models to prepare them for use by the rendering engine, while preserving their intuitive nature for editing. This same curve-based representation is also used to render and edit measured BRDF data. We also describe a fast precomputation algorithm that uses triangular area lights which are more accurate than point lights (and necessary for highly specular BRDFs). Finally, we introduce a new mode of render-time computation that makes use of frame-to-frame coherence with minimal overhead. It is likely that many of these ideas can also be leveraged into standard PRT for relighting.

In the future, we would like to investigate inclusion of global illumination effects. In this case, the final image is no longer linear in the BRDF because of multiple bounces. Even though linearity is the cornerstone of all PRT algorithms, we are optimistic that one bounce interreflections can be handled if we allow some restrictions on the exit radiance used for secondary reflection. We also plan to investigate trading off the ability to edit 2 curves for the ability to edit lighting and 1 curve, in a similar formulation. Another interesting area of research is to investigate semantically meaningful editing operations for measured reflectance. Higher level abstractions such as those presented in [Matusik et al. 2003] would be useful if they can be map onto the curve-based BRDFs.

Historically, computer graphics users have had the ability to interactively modify the lighting and viewpoint in real-time for complex scenes. This paper has closed a major gap, allowing real-time *editing of BRDFs* in complex lighting, and we predict many future developments in BRDF editing and interactive material design.

Acknowledgments: We thank Eitan Grinspun for his general support, advice on wavelets, and assistance with proofs in Appendices A and B. Thanks to Jason Lawrence for contributing factored versions of measured BRDFs. This research was funded by a Sloan Research Fellowship and NSF grants #0305322 and #0446916. The cloth and handles were created, and the scenes were composed using Maya.

References

AGARWAL, S., RAMAMOORTHI, R., BELONGIE, S., JENSEN, H. 2003. Structured Importance Sampling of Environment Maps. *ACM Transactions on Graphics (SIGGRAPH)*, 22, 3, 605–612.

ASHIKHMIN, M., SHIRLEY, P. 2000a. An Anisotropic Phong BRDF Model. *Journal of Graphics Tools*, 5, 2, 25-32.

ASHIKHMIN, M., PREMOZE, S., SHIRLEY, P. 2000b. A Microfacet-based BRDF generator. Proceedings of *ACM SIGGRAPH 2000*, 65-74.

ASSARSSON, U., DOUGHERTY, M., MOUNIER, M., MOLLER, T. 2003, A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware. *ACM Transactions on Graphics (SIGGRAPH)*, 22, 3, 511-520.

BEN-ARTZI, A., RAMAMOORTHI, R., AGARWALA, M. 2006. Efficient Shadows for Sampled Environment Maps. *Journal of Graphics Tools*, 11, 1, 13-36.

BLINN, J. F., 1977. Models of Light Reflection for Computer Synthesized Pictures. *Computer Graphics and Interactive Techniques*.192-198.

COOK, R.L, TORRANCE, K.E., 1982. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1, 1, 7-24.

DAUBECHIES, I. 1988. Orthonormal bases of compactly supported wavelets, *Comm. Pure & Appl. Math.*, 41, 909-996.

DANA, K. J., B. VAN GINNEKEN, S. K. NAYAR, J. J. KOENDERINK: 1999, Reflectance and Texture of Real World Surfaces. *ACM Transactions on Graphics*, 18, 1, 1–34.

ERSHOV, S., KOLCHIN, K., MYSZKOWSKI, K., 2001. Rendering Pearlescent Appearance Based on Paint-Composition Modelling. *Eurographics*, 20, 3.

JAROSZKIEWICZ, R., MCCOOL, M. 2003. Fast Extraction of BRDFs and Material Maps from Images. *Graphics Interface 2003*. 1-10.

KAUTZ, J., MCCOOL, M. 1999. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. *Proceedings of the 10th Eurographics Workshop on Rendering*. 281-292.

LALONDE, P., FOURNIER, A. 1997. A Wavelet Representation of Reflectance Functions. *IEEE Trans. on Visualization & Comp. Graphics,* 3(4) 329-336.

LAFORTUNE, E., FOO, S., TORRANCE, K., GREENBERG, D. 1997. Non-Linear Approximation of Reflectance Functions. *ACM SIGGRAPH 97*, 117-126.

LAWRENCE, J., RUSINKIEWICZ, S., RAMAMOORTHI, R. 2004. Efficient BRDF Importance Sampling Using a Factored Representation. *ACM Transactions on Graphics (SIGGRAPH)*, 23, 3, 496-505.

LAWRENCE, J., BEN-ARTZI, A., DECORO, C., MATUSIK, W., PFISTER, H., RAMAMOORTHI, R., AND RUSINKIEWICZ, S. 2006. Inverse Shade Trees for Non-Parametric Material Representation and Editing. *ACM Transactions on Graphics (SIGGRAPH)*, 25, 3.

LIU, X., SLOAN, P., SHUM, H., SNYDER, J. 2004. All-Frequency Precomputed Radiance Transfer for Glossy Objects. *Eurographics Symposium on Rendering*. 337-344.

MATUSIK, W., PFISTER, H., BRAND, M., MCMILLAN, L. 2003. A Data-Driven Reflectance Model. *ACM Transactions on Graphics (SIGGRAPH)*, 223, 3.

MAXWELL, J.R., BEARD, J., WEINER, S., LADD, D. 1973. Bidirectional reflectance model validation and utilization. *Technical Report AFAL–TR–73–303*, Environmental Research Institute of Michigan (ERIM), October.

MCALLISTER, D., LASTRA, AL, HEIDRICH, W. 2002. Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. *Eurographics Workshop on Graphics Hardware*, 79-88.

MCCOOL, M., ANG,J., AHMAD,A. 2001.Homomorphic Factorization of BRDFs for High-Performance Rendering. Proceedings of *ACM SIGGRAPH 2001*, 171-178.

NG, R., RAMAMOORTHI, R., HANRAHAN, P. 2003. All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (SIGGRAPH)*, 22, 3, 376-381.

NG, R., RAMAMOORTHI, R., HANRAHAN, P. 2004. Triple product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics (SIGGRAPH)*, 23, 3, 475-485.

NGAN, A., DURAND, F., MATUSIK, W. 2005. Experimental Analysis of BRDF Models. *Eurographics Symposium on Rendering*. 117-126.

NIMEROFF, J., SIMONCELLI,E., DORSEY,J. 1994. Efficient Rerendering of Naturally Illuminated Environments. *Eurographics Rendering Workshop 94*, 359-373.

OREN, M., NAYAR, S. 1994. Generalization of Lambert's Reflectance Model. Proceedings of *ACM SIGGRAPH 94*. 239-246.

PEARL – A paint design tool. *www.integra.jp/eng/products/pearl*.

PELLACINI, F., VIDIMCE, K., LEFOHN, A.E., MOHR, A., LEONE, M., AND WARREN, J. 2005. Lpics: A Hybrid Hardware-Accelerated Relighting Engine for Computer Cinematography. *ACM Transactions on Graphics (SIGGRAPH)*. 24, 3, 464-470.

PHONG, BUI TUONG. 1975. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18, 6, 311-317.

RAMAMOORTHI, R., AND HANRAHAN, P. 2002. Frequency Space Environment Map Rendering. *ACM Transactions on Graphics (SIGGRAPH)*, 21, 3.

RUSINKIEWICZ, S. 1998a. bv: *graphics.stanford.edu/~smr/brdf/bv*.

RUSINKIEWICZ, S. 1998b. A New Change of Variables for Efficient BRDF Representation. *Eurographics Rendering Workshop 98*, 11-22.

SCHLICK, C. 1994. An inexpensive BRDF Model for Physically-Based Rendering. *Computer Graphics Forum (Eurographics)*, 13, 3, 233-246.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics (SIGGRAPH)*, 21, 3, 527-536.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2003. Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics (SIGGRAPH)*, 24, 3. 382-391.

SLOAN, P., LUNA, B., AND SNYDER, J. 2005. Local, Deformable Precomputed Radiance Transfer. *ACM Transactions on Graphics (SIGGRAPH)*, 24, 3. 1216-1224.

WAN, L., WONG, T., AND LEUNG, C. 2005. Spherical Q2-tree for Sampling Dynamic Environment Sequences. *Eurographics Symposium on Rendering*, 21-30.

WANG, R., TRAN, J., AND LUEBKE, D. 2004. All-Frequency Relighting of Non-Diffuse Objects using Separable BRDF Approximation. *Eurographics Symposium on Rendering*, 345-354

WARD, G. 1992. Measuring and Modeling Anisotropic Reflection, Proceeding of *ACM SIGGRAPH 92*. 265-272.

ZHOU, K., JU, Y., LIN, S., GUO, B., AND SHUM, H. 2005. Precomputed Shadow Fields for Dynamic Scenes. *ACM Transactions on Graphics (SIGGRAPH)*, 25, 3. 1196-1201.

Appendix A: Computing Overlaps

This appendix details how we compute the overlaps $\Delta_{m,j}$ between the triangle light m and BRDF band j, as introduced in Equation 22. See Figure 3(c) for an example using $\gamma=\theta_r$.

First, we consider the overlap between a planar triangle and a region bordered by horizontal lines in the plane. In this case, the vertical or y-coordinate is simply the relevant parameter (e.g. θ_r). The key idea is that we only need this parameter at each vertex, which is known, given that a light's vertex represents a particular value of ω_i.

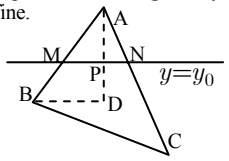
The fraction of the area of a ΔABC above a line $y=y_0$ is given by:

$$\frac{\Delta AMN}{\Delta ABC}=\frac{\sin A|AM||AN|}{\sin A|AB||AC|}=\frac{|AM||AN|}{|AB||AC|}$$

We construct ΔABD so that $y_D=y_B$. By similar triangles, and an analogous construction for the right half, we find that the fraction depends only on the y values (γ values) of the vertices:

$$\frac{AM}{AB}=\frac{AP}{AD}=\frac{y_A-y_0}{y_A-y_B}\ ;\ \frac{\Delta AMN}{\Delta ABC}=\frac{(y_A-y_0)^2}{(y_A-y_B)(y_A-y_C)}.$$

We compute the percentage above and below the boundary lines of a band to find the overlap within the band. It is important to be able to calculate the overlaps based on a single coordinate because our parameterizations are 1D, and there is no notion of 'x' values for the vertices of triangular lights. Even in the simple Phong parameterization, $\gamma=\theta_r$, a perpendicular axis (possibly ϕ_r) does not exist, and would be ambiguous to define.



The remaining question is what to do when the assumption of local linearity in the parameterization breaks down (as it will for example, near the pole of the parameterization, causing the lines of constant γ to be curved, relative to the triangle.) To account for this, we simply subdivide a triangle into four smaller triangles. If the fractions of the area, added up from these four, agree with the coarser calculation to some tolerance, we consider the parameterization locally linear and stop. Otherwise, we subdivide again recursively. Figure 3 shows the original subdivision of the lighting environment for one image pixel, and the reader may wish to find large triangles that would require subdivision, and smaller ones that would not.

Appendix B: Use of Daubechies 4 Wavelets

The low-pass and high-pass filter coefficients for Daubechies 4 [1988] wavelets we used are shown on the right. Like the Haar basis, Daub4 wavelets are orthonormal. Unlike Haar, they are designed for C1, not just C0 continuity.

low-pass=	[d1 d2 d3 d4]
high-pass=	[d4 –d3 d2 –d1]
d1=	0.4829629131445341
d2=	0.8365163037378077
d3=	0.2241438680420134
d4=	-0.1294095225512603

To obtain the Daub4 coefficients of the user's input curve, we first discretize it using the box basis functions. It is important to integrate over the domain of each box function (as opposed to taking point samples) in order to avoid popping artifacts during edits. So at each frame, the following operation happens once for each edited curve:

$$c_j=\int_{\frac{\pi(j-1)}{2J}}^{\frac{\pi j}{2J}} b_j(\gamma)f(\gamma)d\gamma,\ \text{where we've assumed } \gamma\in[0,\tfrac{\pi}{2}], \text{ as it usually is.}$$

Using column vectors, we use the synthesis matrix for Daub4, $\mathbf{S}_{D4}$, to write the wavelet transformed versions of $\mathbf{c}^A$ and $\mathbf{c}^B$:

$$\mathbf{c}^A_{D4}=\mathbf{S}_{D4}\mathbf{c}^A\ ;\ \mathbf{c}^B_{D4}=\mathbf{S}_{D4}\mathbf{c}^B.$$

To find the properly transformed $\mathbf{U}, \mathbf{U}_{D4}$, we refer to Equation 23:

$$R=(\mathbf{c}^B_{D4})^T\mathbf{U}_{D4}\mathbf{c}^A_{D4}\ ;\ \mathbf{U}_{D4}=\mathbf{S}^{-T}_{D4}\mathbf{U}\mathbf{S}^{-1}_{D4}=\mathbf{S}_{D4}\mathbf{U}\mathbf{S}^T_{D4},$$

where the last equality uses the fact that the Daubechies 4 synthesis matrix is orthonormal. Therefore, we must use the standard 2D transform of the matrix $\mathbf{U}$, which first uses the 1D transform to convert the rows, and then uses it to convert the columns (or vice-versa). Though the non-standard 2D transform often provides better compression, it is not applicable for our use.