A Precomputed Polynomial Representation for Interactive BRDF Editing with Global Illumination

Aner Ben-Artzi   Kevin Egan   Frédo Durand   Ravi Ramamoorthi

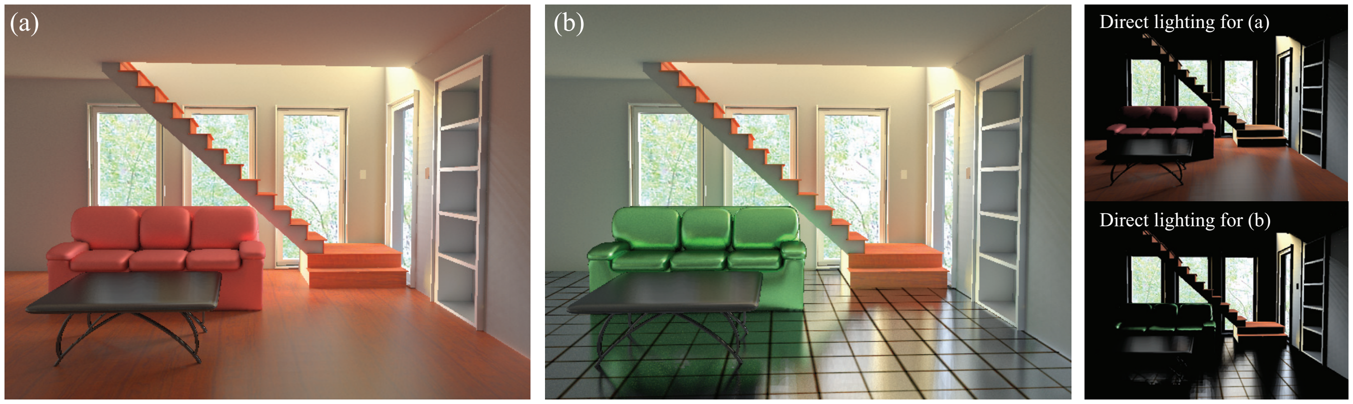

Figure 1: *We simulate an interior design session in which we edit the BRDFs of the couch and floor. The couch's red fabric in (a) is loaded from measured data, and edited to a more specular green material in (b). The floor is re-textured and made very glossy in (b). The reflections of objects in the near-mirror floor, and color bleeding from the couch to the staircase can be seen here as well as in closeups in Fig. 10. The scene is lit only from the large windows, using the top half of an exterior environment map (campus). We can see in the comparisons to direct lighting (far right) that most of the room's lighting is a result of indirect illumination.*

Abstract

The ability to interactively edit BRDFs in their final placement within a computer graphics scene is vital to making informed choices for material properties. We significantly extend previous work on BRDF editing for static scenes (with fixed lighting and view), by developing a precomputed polynomial representation that enables interactive BRDF editing with global illumination. Unlike previous precomputation-based rendering techniques, the image is *not linear* in the BRDF when considering interreflections. We introduce a framework for precomputing a multi-bounce tensor of polynomial coefficients, that encapsulates the nonlinear nature of the task. Significant reductions in complexity are achieved by leveraging the low-frequency nature of indirect light. We use a high-quality representation for the BRDFs at the first bounce from the eye, and lower-frequency (often diffuse) versions for further bounces. This approximation correctly captures the general global illumination in a scene, including color-bleeding, near-field object reflections, and even caustics. We adapt Monte Carlo path tracing for precomputing the tensor of coefficients for BRDF basis functions. At runtime, the high-dimensional tensors can be reduced to a simple dot product at each pixel for rendering. We present a number of examples of editing BRDFs in complex scenes, with interactive feedback rendered with global illumination.

1 Introduction

Recent advances in real-time rendering have improved the ability of designers to interactively specify lighting and materials in computer graphics scenes. While relighting systems have long provided feedback with global illumination in complex scenes [Dorsey et al. 1995], BRDF editing has been limited to simplified settings such as point lights.

Recently, Ben-Artzi et al. [2006] have introduced the ability to edit BRDFs under natural illumination, albeit only with direct lighting. This is a significant limitation since indirect illumination and glossy reflection are essential to the realism of today's renderers, and are often critical to correctly perceive and choose material properties.

In this paper, we develop a precomputation-based method for interactive BRDF editing with global illumination (see results in Fig. 1). The main challenge arises from the fact that final scene radiance (an image) is *not even linear* in the objects' BRDFs. It is well known that albedo, or more generally a BRDF, has a non-linear effect because it multiplies the light at each bounce. We develop a higher-order representation of the image as a function of the scene's BRDFs. We precompute a tensor at each pixel, fixing the lighting and view for a static scene, but leaving the BRDFs unspecified until runtime, when they can be edited.

Our first contribution, in Sec. 3, is a general theoretical framework for BRDF editing, based on a bilinear formulation of the reflection operator, that extends the linear operator formulation of rendering [Arvo et al. 1994]. We show how the precomputed matrix of previous methods must be extended to a multi-bounce tensor of polynomial coefficients.

The full multi-bounce tensor is a complete representation of the image as a function of scene BRDFs, but is computationally too expensive to treat in full generality. We consider frequency characteristics, developing a tractable approximation that preserves most important perceptual effects (Sec. 4). Specifically, the first bounce from the eye usually uses the full BRDF, to capture glossy reflections, while subsequent bounces use a lower-frequency approximation to capture overall shading effects. Within this general framework, we show two possibilities—where further bounces (up to the fourth-bounce) are treated as purely diffuse (Figs. 1, 6 and 9), and where additionally, the second bounce from the eye uses a lower-frequency approximation to achieve accurate indirect reflections in glossy surfaces (Fig. 4), or even intricate effects like caustics (Fig. 11).

For precomputation (Sec. 5.1), we show how Monte Carlo path tracing can be extended to precompute the multi-bounce tensors needed. For rendering (Sec. 5.2), since only one object's BRDF is edited at a time, we show that the tensor can be reduced to a few vector dot products at each pixel, whose coefficients can be computed in a very fast runtime preprocess. Our results show a variety of BRDF edits, with interactive feedback rendered with global illumination.

2 Previous Work

Precomputed Radiance Transfer (PRT): We build on PRT ideas for static scenes [Sloan et al. 2002; Ng et al. 2003]. While those methods focus on lighting design with fixed BRDFs, we focus on BRDF editing, with fixed lighting. We are inspired by a body of recent work that underscores the importance of interreflections in relighting [Hasan et al. 2006; Wang et al. 2006; Kontkanen et al. 2006]. All these approaches exploit the linearity of relighting with respect to light intensity, even when global illumination is taken into account. In contrast, BRDF editing is fundamentally non-linear when global illumination is considered.

Most previous PRT methods precompute a linear light transport vector at each image location, taking advantage of the linearity of light. We extend this concept to a general tensor of coefficients for a high-dimensional polynomial. The idea of going from linear to quadratic or cubic precomputed models has also begun to be explored in physical simulation [Barbic and James 2005] but in the context of differential equations and model dimensionality reduction. In the context of real-time rendering, [Sun and Mukherjee 2006] precompute with a larger product of functions, leading to an N-part multiplication at runtime. Each function is still precomputed independently, and the runtime calculations are still linear in any of the individual functions.

While some PRT methods allow for all-frequency effects and view changes [Wang et al. 2006], BRDF editing cannot take advantage of such factorization-based approaches, since the BRDF lobe at a pixel is defined over both the dimensions of lighting (ω_i) and view (ω_o) simultaneously, and also depends on the surface normal. Therefore, we fix the lighting, view, and geometry, but could in principle allow a small number of pre-defined views or lighting conditions to be updated simultaneously (as in [Ben-Artzi et al. 2006]).

Global Illumination: Our precomputation method is inspired by offline global illumination algorithms, such as Monte Carlo path tracing [Kajiya 1986]. We have also been able to adapt finite element radiosity [Cohen and Wallace 1993], although we found meshing and complexity issues difficult to deal with for our complex scenes, and do not discuss it further. Global illumination techniques usually require the BRDF to be fixed, and use it for importance sampling or hierarchy construction—we develop extensions that are independent of the BRDF, and allow real-time editing. In effect, we precompute a *symbolic representation* of the image, which can be evaluated at runtime with polynomials in the user-specified BRDF values, to obtain the final intensity.

Séquin and Smyrl [1989] also precomputes a symbolic representation of the final image for recursive ray tracing—but not full global illumination. Phong shading can be evaluated at runtime, allowing later changes to surface parameters, while reflected and refracted contributions are handled with pointers to sub-expressions. In contrast, we seek to simulate complex lighting and full global illumination, with many possible illumination paths. Therefore, we cannot afford to store or sum all subexpressions. Instead, we show that the final symbolic expression is a polynomial and only precompute its coefficients. We also allow editing of general parametric and measured BRDFs.

BRDF Representations and Editing: Recent work in BRDF editing has begun to allow edits while rendering important visual effects such as environment maps [Colbert et al. 2006] and general complex lighting with cast shadows [Ben-Artzi et al. 2006]. However, they are limited to direct lighting, which neglects many aspects of appearance in realistic settings with global illumination. We utilize existing BRDF representations, giving the user the ability to edit them interactively in a scene, with complex lighting, shadows and interreflections. Our method supports analytic models like Blinn-Phong or Cook-Torrance, measured half-angle distributions [Ashikhmin et al. 2000; Ngan et al. 2005], and variants of data-driven factored or curve-based models [McCool et al. 2001; Lawrence et al. 2006]. We allow users to either edit values corresponding to parameters in standard analytic BRDFs, or 1D curves for measured data.

3 General Theoretical Framework

This section introduces a general theoretical framework for BRDF editing with global illumination, independent of any specific implementation. It builds on the geometric and reflection operators introduced by Arvo et al. [1994] and shows how BRDF editing can be formulated in terms of a new bilinear reflection operator $\mathbf{K}$. The operator notation is compact, and alleviates the need for complex integrals in the equations. In Sec. 4, we discuss our approximations within this framework, that make the computation tractable.

$B(x, \omega_o)$	Outgoing radiance (image)
$E(x, \omega_o)$	Emissive radiance of light sources
$L(x, \omega_i)$	Local incident radiance
$R(x, \omega_i, \omega_o)$	BRDFs of all points in the scene
$T^N(x, \omega_o)$	Precomputed multi-bounce tensor
$\rho^m(\omega_i, \omega_o)$	BRDF of object m
$b_j^m(\omega_i, \omega_o)$	Basis function j for the BRDF of object m
$H^m(x)$	Spatial weight map or texture for object m
$\mathbf{G}$	Linear geometric operator
	$\mathbf{G} : B(x, \omega_o) \mapsto L(y, \omega_i)$
	$(\mathbf{G}B)(x, \omega) \equiv B(x'(x, \omega), -\omega)$
$\mathbf{K}(R)$	Reflection operator (equation 2)
c_j^m	BRDF coefficients (equation 4)
d_m	Equivalent albedo of object m (appendix B)
$\overline{d}_z$	Product of albedos d_m
$\vec{X}_i$	Light path with contribution $f(\vec{X}_i)$
F_{jn}^i	Tensor coefficient after freezing BRDFs
J	Number of BRDF bases (usually 64 or 128)
M	Number of objects ($M \sim 5$)
W	Total basis functions ($W = JM \sim 500$)
Z	Number of terms for diffuse $\overline{d}_z$ ($Z \sim 64$)

Table 1: *Table of Notation*

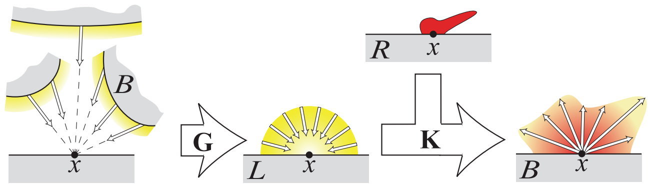

Figure 2: *Schematic of the rendering equation. Outgoing radiance from all surfaces in the scene (left) is converted by the* $\mathbf{G}$ *operator into a local incident radiance field at each location (middle), which is then reflected via the bilinear reflection operator* $\mathbf{K}$*, that takes as input both the incident lighting and the BRDF.*

3.1 Basic Framework using the bilinear K operator

The rendering equation [Kajiya 1986] can be written as a linear operator equation [Arvo et al. 1994]. We build on this foundation and write the rendering equation as:

$$B = E + \mathbf{K}(R)\,\mathbf{G}B, \tag{1}$$

where $B(x, \omega_o)$ is the outgoing surface radiance, $E(x, \omega_o)$ is the emission and $\mathbf{G}$ is the linear geometric operator that converts the outgoing radiance from distant surfaces to a local incident radiance field as in [Arvo et al. 1994]. Figure 2 shows a schematic, and Table 1 summarizes notation.

Arvo et al. [1994] define $\mathbf{K}$ as a linear reflection operator on the local incident radiance field L. In our first departure from previous representations, we make explicit $\mathbf{K}$'s dependence on the BRDFs R of scene objects. We write $\mathbf{K}$ as a bilinear operator that takes an additional input $R(x, \omega_i, \omega_o)$ which describes the BRDF at each point in the scene and is the kernel of integration in $\mathbf{K}$,

$$\mathbf{K} \;:\; L(x, \omega_i), R(x, \omega_i, \omega_o) \mapsto B(x, \omega_o)$$

$$(\mathbf{K}(R)\,L)(x, \omega_o) \equiv \int_{\Omega_{2\pi}} R(x, \omega_i, \omega_o) L(x, \omega_i) \cos\theta_i d\omega_i. \tag{2}$$

Note that $\mathbf{K}$ is bilinear, or linear with respect to *both* inputs—incident lighting L, and the BRDFs of objects in the scene R. That is, for scalars a and b,

$$\begin{aligned} \mathbf{K}(aR_1 + bR_2)\,L &= a\mathbf{K}(R_1)\,L + b\mathbf{K}(R_2)\,L \\ \mathbf{K}(R)\,(aL_1 + bL_2) &= a\mathbf{K}(R)\,L_1 + b\mathbf{K}(R)\,L_2. \end{aligned} \tag{3}$$

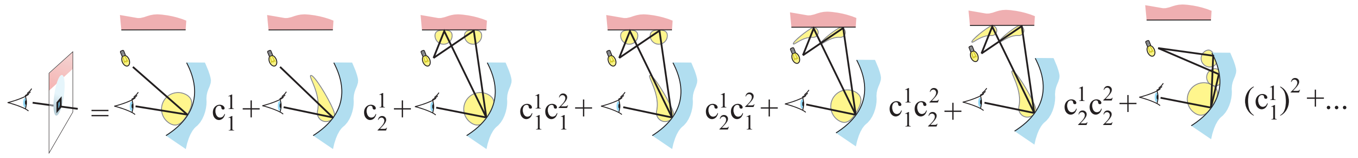
$= c_1^1 + c_2^1 + c_1^1 c_1^2 + c_2^1 c_1^2 + c_1^1 c_2^2 + c_2^1 c_2^2 + (c_1^1)^2 + \ldots$

Figure 3: *The final value of each pixel is a polynomial in the BRDF coefficients. Here we show an example with 2 surfaces and two basis BRDFs shown in yellow (diffuse and specular). Note that the BRDFs we use in practice are different. The combinatorics of multivariate polynomial coefficients make a tensor notation particularly useful.*

We now seek to relate R (and hence $\mathbf{K}$) to editable BRDFs of individual objects. We assume there are M objects in the scene, and for now that each object has a single BRDF. Let object m have[1] BRDF ρ^m. We assume the BRDF can be represented as a linear combination of functions over the domain (ω_i, ω_o),

$$\rho^m(\omega_i, \omega_o) = \sum_{j=1}^{J} c_j^m b_j^m(\omega_i, \omega_o). \tag{4}$$

The BRDF basis functions b could be spherical harmonics, wavelets or any other linear basis. We follow previous BRDF editing methods [Ben-Artzi et al. 2006; Lawrence et al. 2006], that have used box functions over a suitable 1D parameterization such as the half-angle, as described in Appendix A. They have shown that a 1D parameterization is appropriate for most BRDF edits, as well as being compatible with parametric edits of most common BRDF models.

Our goal is to use these basis BRDFs to create a method that allows us to alter the kernel of integration in the $\mathbf{K}$ operator by specifying different weights c_j^m. We first need to use the b_j^ms to describe R over all surfaces. In order to encode per-object BRDFs, we define a surface mask $H^m(x)$ that is 1 if x is on object m, and 0 otherwise.[2]

$$R(x, \omega_i, \omega_o) = \sum_{m=1}^{M} H^m \rho^m = \sum_{m=1}^{M} H^m(x) \sum_{j=1}^{J} c_j^m b_j^m(\omega_i, \omega_o) \tag{5}$$

The super/subscripts in the above equation implicitly define basis functions for the full spatially varying R,

$$R = \sum_{m=1}^{M} \sum_{j=1}^{J} c_j^m R_j^m \ ; \ R_j^m(x, \omega_i, \omega_o) = H^m(x) b_j^m(\omega_i, \omega_o). \tag{6}$$

For simplicity, we will often use a single index w (or u or v) to refer to the double script ${}_j^m$, with $w \in [1, W] : W = MJ$.

3.2 Polynomial Representation for Multi-Bounce

The solution of equation 1 can be expressed as the expansion

$$B = E + \mathbf{K}(R)\,\mathbf{G}E + \mathbf{K}(R)\,\mathbf{G}\mathbf{K}(R)\,\mathbf{G}E + \ldots \tag{7}$$

where each term N has an intuitive interpretation, as corresponding to N bounces of light from source(s) to viewer.

All current relighting methods rely on the linearity of B with respect to E. Previous BRDF editing methods also take advantage of the linearity of the 1-bounce term in B (i.e., $\mathbf{K}(R)\,\mathbf{G}E$) with respect to $\mathbf{K}$ (and hence with respect to R), requiring them to render using only direct lighting.

However, *this linearity no longer applies* for BRDF editing with global illumination because the operator $\mathbf{K}(R)$ is applied multiple times. Even for 2-bounce reflections, the system becomes quadratic, and must be represented with a quadratic multivariable polynomial in the c_ws. The N-bounce solution is an order N polynomial. We now show how to extend the general PRT approach to these polynomial equations. We start by considering 2-bounce reflection,

$$\begin{aligned} B^2 &= \mathbf{K}(R)\,\mathbf{G}\mathbf{K}(R)\,\mathbf{G}E && (8)\\ &= \mathbf{K}(\sum_u c_u R_u)\,\mathbf{G}\mathbf{K}(\sum_v c_v R_v)\,\mathbf{G}E && (9)\\ &= \sum_u c_u \left(\mathbf{K}(R_u)\,\mathbf{G} \sum_v c_v \left(\mathbf{K}(R_v)\,\mathbf{G}E\right)\right) && (10)\\ &= \sum_u \sum_v c_u c_v \mathbf{K}(R_u)\,\mathbf{G}\mathbf{K}(R_v)\,\mathbf{G}E. && (11) \end{aligned}$$

The bilinear nature of $\mathbf{K}$ is crucial here. We use the linearity of $\mathbf{K}$ with respect to the BRDF to get equation 10, and the linearity of $\mathbf{K}$ and $\mathbf{G}$ with respect to radiance to get equation 11.

For BRDF editing, the coefficients (c_u and c_v) become the variables of our computation. The fixed quantities for precomputation are the basis BRDF distributions (R_u and R_v), that depend only on the parameterizations of the various BRDFs. $\mathbf{G}$ and E are also known, defined by the geometry and lighting of the scene, respectively. We precompute a 2-bounce transport function T_{uv}^2 and calculate B^2 as

$$\begin{aligned} T_{uv}^2 &= \mathbf{K}(R_u)\,\mathbf{G}\mathbf{K}(R_v)\,\mathbf{G}E \\ B^2 &= \sum_u \sum_v T_{uv}^2 c_u c_v. && (12) \end{aligned}$$

Most generally, T_{uv}^2 and B^2 are defined over all spatial locations and view directions. In practice, since we fix the view, $T_{uv}^2(x, \omega_o(x))$ is an order 2 tensor (that is, a matrix for 2-bounce reflection) at each pixel. $B^2(x, \omega_o(x))$ at each pixel is a quadratic polynomial in the variables c, with coefficients given by T^2. When evaluated, it becomes the radiance function for the scene due to light that has reflected off exactly 2 surfaces. More explicitly,

$$B^2 = T_{11}(c_1)^2 + T_{12}c_1c_2 + \ldots T_{uv}c_uc_v + \ldots T_{WW}(c_W)^2. \tag{13}$$

Figure 3 illustrates such polynomials for paths of length 1 and 2. All 1- and 2-term combinations of the BRDFs are possible, including repetitions of the same basis function, as in $T_{11}(c_1^1)^2$, since concave objects can reflect onto themselves. Following the same logic, the N-bounce energy is

$$B^N = \sum_{w_N} \sum_{w_{N-1}} \ldots \sum_{w_1} T_{w_N w_{N-1} \ldots w_1}^N c_{w_N} c_{w_{N-1}} \ldots c_{w_1}, \tag{14}$$

where T^N is an order N tensor for each pixel, whose size varies with the number of objects and BRDF basis functions per object. We are evaluating a multi-variable, degree N polynomial where each w runs over all W values (all basis functions). The variables of this polynomial are the unknown scalar c's. The coefficients are stored in T^N,

$$T_{w_N w_{N-1} \cdots w_1}^N = \mathbf{K}(R_{w_N})\,\mathbf{G}\mathbf{K}(R_{w_{N-1}})\,\mathbf{G} \ldots \mathbf{K}(R_{w_1})\,\mathbf{G}E. \tag{15}$$

Finally, we construct the image to be displayed by adding the contributions from the different path-lengths:

$$B = E + B^1 + B^2 + \ldots + B^{\overline{N}}, \tag{16}$$

where we cut off the expansion in equation 7 to $\overline{N} + 1$ terms.

[1]We use superscripts to denote some property of the function, and parentheses for explicitly raising to a power, so ρ^2 is the second in a series, whereas $(\rho)^2$ is ρ squared.

[2]H can also take on non-binary values to encode spatial weight maps for combining BRDFs [Lensch et al. 2001; Lawrence et al. 2006], and/or for describing textures.

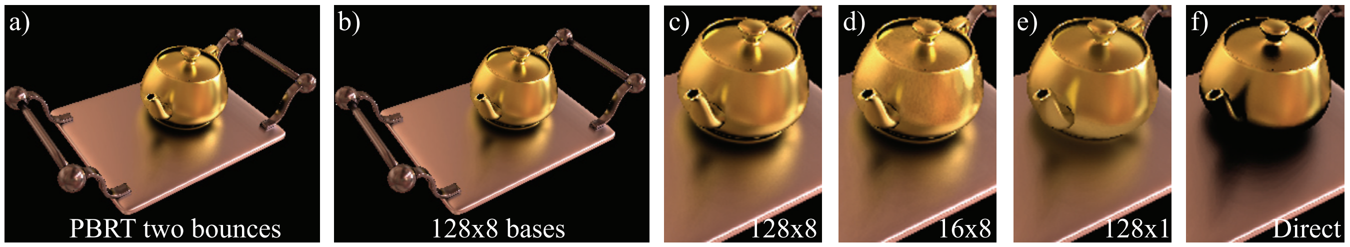

Figure 4: *An evaluation of the accuracy of different two-bounce decreasing-frequency BRDF series, precomputed and rendered in our editing system. (a) Full resolution BRDFs rendered offline in PBRT—hence a series of* (∞, ∞) *(b) A* $(128, 8)$ *series (c)* $(128, 8)$ *(d)* $(16, 8)$ *(e)* $(128, 1)$ *or diffuse for second bounce from the eye (f) Direct lighting only (Ben-Artzi et al. 06). We see that a very low-frequency second-bounce BRDF approximation* $(128, 8)$ *in (b) and (c) is essentially exact. Moreover, even a diffuse second bounce approximation* $(128, 1)$ *in (e) provides the correct shiny material perception of the tray and indirect reflections of tray and teapot. By contrast, direct lighting shows only a black shadow of the teapot on the shiny tray, and the spout and handle do not reflect in the teapot.*

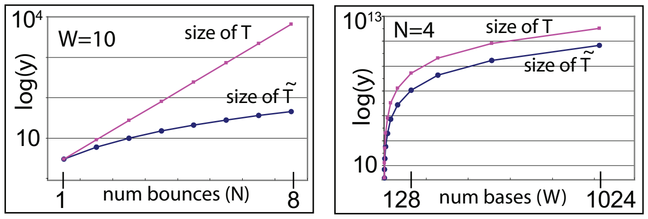

Figure 5: *Rate of growth for the precomputed data-structure, T with and without taking symmetry into account. Left: The base of the exponential growth is effectively reduced due to symmetry (smaller slope). Right: The slope of both growth rates is the same; symmetry offers only a constant offset, corresponding to a scale factor (90 here) for the same asymptotic behavior.*

4 Managing Complexity

Section 3 has presented a theoretical framework that is fully general. However, we need to manage complexity in order to develop a tractable algorithm. In particular, the number of terms in equation 14 is $(W)^N$. Recall that W is already JM, the number of basis BRDFs times the number of objects. For typical values of M (5) and J (64-128), (so $W \sim 500$) the computational and storage cost for all but the first bounce become prohibitive. In this section, we derive efficient and perceptually accurate approximations.

Some efficiency is obtained by taking advantage of symmetry in the terms of our polynomial. Referring back to equations 12 and 13, we note that T_{12} and T_{21} both get multiplied by $c_1 c_2$. We can therefore define $\tilde{T}$ by summing all T entries which differ by only a permutation of indices. It can be shown that the number of terms now grows as $\binom{N+W-1}{N}$ which is slower than $(W)^N$ (Fig. 5(left)). We can therefore consider a larger number of bounces—in practice, we use up to $N = 4$ bounces, which we find sufficient for convergence.

However, as Fig. 5(right) shows, symmetry only reduces the complexity by a constant factor for growth with respect to W. While this factor is about 90 for the case of $N = 4$ shown, the $O((W)^4)$ behavior has not been reduced.

4.1 Low-Frequency BRDF approximations

To deal with the explosion in the number of bases for R, we make the important observation that equation 15 does not require us to use the same set of BRDFs R for every occurrence of $\mathbf{K}(R)$. We can define a hierarchy of BRDFs[3] for each object, using less bases to represent the BRDF when considering light-surface interactions that are not directly visible to the viewer. For any bounce n on object m,

$$\rho_n^m(\omega_i, \omega_o) \quad = \quad \sum_{j=1}^{J_n} c_j^m b_j^m(\omega_i, \omega_o)$$
$$J = J_N \geq \ldots \geq J_n \geq \ldots \geq J_1 \geq 1, \qquad (17)$$

where c_j^m and b_j^m correspond to the appropriate hierarchy (and are not the same for different ρ_n.) This creates lower-frequency BRDF approximations, motivated by the often-discussed low-frequency nature of indirect lighting and by experiments by [Nayar et al. 2006] and theoretical work that shows that, at each bounce, the BRDFs act as a low-pass filter [Ramamoorthi and Hanrahan 2001; Durand et al. 2005].

We now denote the distribution of BRDFs R with a superscript indicating the number of bases used per object,

$$B^N = \mathbf{K}(R^{J_N})\,\mathbf{G}\mathbf{K}(R^{J_{N-1}})\,\mathbf{G}\cdots\mathbf{K}(R^{J_n})\,\mathbf{G}\cdots\mathbf{K}(R^{J_1})\,\mathbf{G}E. \qquad (18)$$

When equations 14 and 15 use the corresponding hierarchy of R's, the subscripts are modified to run over a smaller domain such that $w_n \in [1, MJ_n]; MJ_n = W_n$. Specifically,

$$T^N_{(m_N j_N)\cdots(m_n j_n)\cdots(m_1 j_1)}(x, \omega_o) = \qquad (19)$$
$$\mathbf{K}(R^{J_N}_{m_N j_N})\,\mathbf{G}\ldots\mathbf{K}(R^{J_n}_{m_n j_n})\,\mathbf{G}\ldots\mathbf{K}(R^{J_1}_{m_1 j_1})\,\mathbf{G}E,$$

where the subscripts $w_n = (m_n j_n)$ explicitly denote the object m_n and BRDF basis function j_n. This is the most general form of the multi-bounce tensor T^N.

A variety of decreasing-frequency BRDF series within our editing system are illustrated in Fig. 4. The scene is lit by an environment map and modeled after a figure in [Ben-Artzi et al. 2006]. For clarity in the comparisons, we use only two bounces, and show series $(J_2, J_1) \equiv (128, 8); (16, 8); (128, 1)$. Figure 4f shows direct lighting only, as in [Ben-Artzi et al. 2006]—this omits important effects for the perception of materials and shininess, like the reflection of the teapot in the shiny tray (a black shadow results instead), or the reflection of the spout in the teapot. Figure 4a is ground truth, rendered offline (with two bounces) in PBRT [Pharr and Humphreys 2004].

Figure 4 underscores that *further bounces can be represented with very low-frequency BRDFs.* The ground truth in (a) is essentially identical to the $(128, 8)$ BRDF series in (b) and (c), that uses only 8 BRDF bases for the second bounce. In fact, our direct material perception primarily responds to the glossy tray reflecting nearby objects like the teapot. Therefore, even the purely diffuse approximation for further bounces from the eye $(128, 1)$ in (e) is usually adequate. In that case, the teapot appears diffuse in the reflection in (e), but this approximation is only for objects seen indirectly through reflections, and not easily noticeable.

Often, our choices are dictated by available computational resources. For a given complexity $J_2 J_1 = 128$, two possible options are $(16, 8)$ in (d) and $(128, 1)$ in (e). Both images are quite accurate, and can be edited within our system. They make different tradeoffs. The glossy reflection of the teapot

[3]This hierarchy is never directly exposed to the user. The user simply edits BRDFs in the usual way, by adjusting parameters or editing high-resolution 1D curves. The system automatically filters these to lower-frequency versions where needed, or computes diffuse equivalents as described in appendix B.

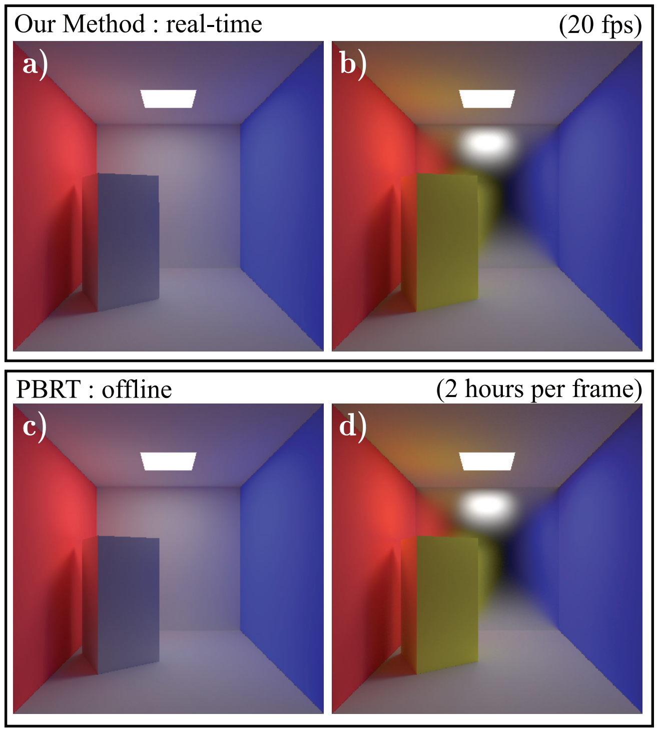

Figure 6: *BRDF edits in Cornell Box. We edit the color of the small box, and also make the back wall shinier, all while rendering interactively with four bounces of global illumination. Notice the correct color bleeding in the scene, and correct glossy reflections of the scene in the back wall of (b). We compare our results to offline ground truth with PBRT in (c) and (d).*

in the tray is slightly more accurate in $(16, 8)$ because of a better BRDF approximation for the second bounce. However, the first bounce is represented at lower frequency than $(128, 1)$—for example, direct reflections on the teapot are somewhat less sharp; this can become more noticeable for measured BRDFs and very shiny materials.

The observations from Fig. 4 indicate that using a diffuse equivalent for further bounces is a reasonable and efficient approximation, and we use it for some of our examples (Sec. 4.2). For more complex effects like caustics, we explore instead a series where the second bounce from the viewer uses a low-frequency approximation (Sec. 4.3).

4.2 Diffuse approximation for further bounces

In the limit, $J_{N-1} = 1$, and we approximate each object's BRDF using a single basis function that is a diffuse lobe, scaled by the "equivalent albedo" of the BRDF. See appendix B for a derivation of the equivalent albedo, d_m. We usually use four bounces $(J, 1, 1, 1)$ (with typically J=64).

$$B^N \approx \mathbf{K}(R^J)\,\mathbf{G}\mathbf{K}(R^1)\,\mathbf{G}...\mathbf{K}(R^1)\,\mathbf{G}E \qquad (20)$$

$$T^N_{(j)(m_{N-1})\cdots(m_1)} = \mathbf{K}(R^J_j)\,\mathbf{G}\mathbf{K}(R^1_{m_{N-1}})\,\mathbf{G}\ldots\mathbf{K}(R^1_{m_1})\,\mathbf{G}E,$$

where we have simplified the index pairs $(m_n j_n)$ in the general tensor of equation 19 as follows. We drop the m_N subscript in the first index pair, since only one object, $m_N(x)$, is visible through a pixel. We also drop the j_n subscripts for further bounces, since there is only one basis BRDF when using R^1. Thus, we also simply use 'j' instead of 'j_N' for the bounce closest to the eye.

This approximation fully treats the first bounce from the viewer, including glossy reflections of the nearby scene. Bounces further from the viewer (and hence reflections of objects not seen directly) are treated as diffuse. The complexity at each pixel reduces from $O((W)^N)$ to $O(J(M)^N)$. We will later see how this reduces further to $O(J)$ for rendering, since we edit only one object or material at a time.

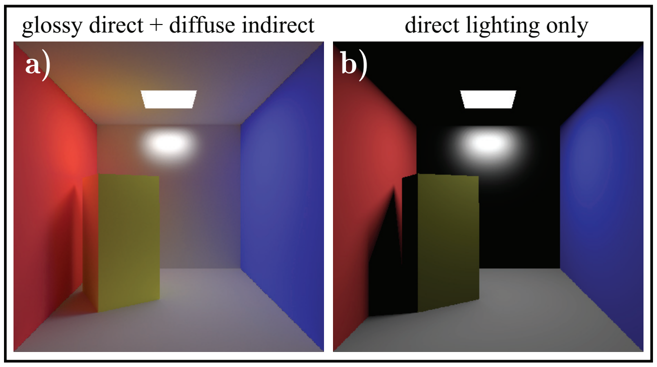

Figure 7: *Qualitative errors in simpler approximations. Compare (a) and (b) to Fig. 6b. (b) is the direct lighting approximation of Ben-Artzi et al., which fails to capture many important global illumination effects. (a) is a sum of (b) and diffuse indirect lighting, capturing some global effects but providing an inconsistent material perception for glossy objects like the back wall.*

Evaluation: Figures 6a and 6b are produced with our system. In (a), all surfaces are diffuse, while in (b) we edit the back wall to make it a glossy material, and change the color of the inner box to yellow. It is clear our method enables perception of material appearance, because objects correctly reflect other nearby objects (see the glossy interreflections of the room in the back wall in (b)), while also accurately preserving global effects like color bleeding onto the large inner box. We compare to ground truth using offline path tracing with PBRT in Figs. 6c and 6d, which confirms the accuracy of the approximation.

By contrast, Fig. 7b shows the direct lighting approximation of [Ben-Artzi et al. 2006] for the configuration in Fig. 6b. Not only is it missing a lot of energy, but it also lacks the reflections of the room in the back wall, which makes it difficult to assess the desired glossiness while editing.

Note that our method treats the first bounce from the eye with the full BRDF to get glossy reflections of nearby objects,

$$B \approx E + \mathbf{K}(R^J)\,\mathbf{G}E + \mathbf{K}(R^J)\,\mathbf{G}\sum_{N=2}^{\overline{N}} \left(\mathbf{K}(R^1)\,\mathbf{G}\right)^{N-1} E.$$

Figure 7a compares to an alternative coarser approximation we originally tried using our framework. This simply adds a diffuse indirect solution to the full direct lighting result. It is essentially a series $(1, 1, 1, 1)$ for the indirect illumination, and therefore the most efficient technique,

$$B \approx E + \mathbf{K}(R^J)\,\mathbf{G}E + \mathbf{K}(R^1)\,\mathbf{G}\sum_{N=2}^{\overline{N}} \left(\mathbf{K}(R^1)\,\mathbf{G}\right)^{N-1} E,$$

where the main difference is that $\mathbf{K}(R^1)$ is used instead of $\mathbf{K}(R^J)$ for the leftmost operator of the multiple-bounce terms. Figure 7a is clearly better than direct lighting only—some global illumination is usually better than none.

However, a comparison with Fig. 6b shows that while further bounces can be approximated as diffuse, the first bounce from the eye does need the full high-frequency BRDF. Unlike our method, Fig. 7a gives an inconsistent material appearance of the back wall, that may be difficult to interpret while editing. While the direct reflection of the light source is glossy, the indirect reflections of the room appear diffuse.

4.3 Slower Decay of BRDF Series

Treating later bounces as diffuse works well in most scenes (see Figs. 1, 4e, 6 and 9). However, in some configurations like concave curved reflectors, higher frequency indirect effects like caustics are visually important [Durand et al. 2005].

To handle such challenging situations, we need to reduce the BRDF frequencies more slowly, using more (8-16) basis functions for the second bounce from the eye. (Cases where even the third or higher bounces away from the eye need to be high-frequency are quite rare, though our general framework does not preclude taking them into account.) We compensate for the extra memory cost either by reducing the number of bases for the first bounce (Fig. 4d) or by using fewer bounces (Fig. 11 has only two bounces).

We have already seen an example of using 8 BRDF basis functions for the second bounce in Fig. 4(b-d), that gives a more accurate reflection of the teapot in the shiny tray. In practice, our editing system also includes diffuse approximations for the third and fourth bounces to augment the series in Fig. 4 (see table 2). An even more challenging example is Fig. 11, that involves caustic effects. In this case, we use 16 BRDF basis functions for the second bounce, with a BRDF series of the form $(J, J/4) \equiv (64, 16)$.

5 Implementation

We now describe our implementation, starting with our precomputation method (Sec. 5.1), followed by the rendering algorithm (Sec. 5.2), and some practical extensions (Sec. 5.3).

5.1 Monte Carlo Precomputation

We need to precompute the tensors defined by equation 19 at each pixel. The important special case for diffuse approximation in further bounces is given by equation 20. Recall that the $\mathbf{K}$ operators involve integrals over the hemisphere, which means that each $T^N(x)$ requires nested (high-dimensional) integrals over ray paths. This is similar to traditional global illumination. We adapt Monte Carlo path tracing [Kajiya 1986] for precomputation because of its flexibility, ease of implementation and negligible memory overhead.

Each value in the different tensors can be seen as a separate integral over the space of paths. However, it is easier and more similar to traditional path tracing to sample path space for all integrals at the same time. We generate random paths, and for each path update the appropriate tensor integrals. We must modify three basic aspects of the path tracer. First, we cannot generate rays by sampling the BRDF (since it is unknown). Second, we must add each path's contribution to the correct tensor element, as opposed to simply contributing to the final pixel radiance. Third, we must compute the contribution of each path using basis BRDFs.

Consider a given path $\vec{X}$ from a point on a light source (ℓ) to the eye (e), that passes through points $x_N, x_{N-1}, \ldots, x_1$ on objects $m_N, m_{N-1}, \ldots, m_1$, as illustrated in Fig. 8.

Sampling path space: According to Monte Carlo theory, any random sampling can be used to generate new directions when building up the path. We follow standard path tracing and generate rays recursively from the eye. We sample the light source at each bounce to generate all path lengths.

Path tracers usually importance sample the BRDF to select a new direction at each intersection. Unfortunately, we have no knowledge of the final BRDF. We cannot sample according to the basis BRDFs either because they will be combined with arbitrary weights at runtime. The simplest approach would be to sample the cosine-weighted hemisphere uniformly, but this would yield high variance when sharp specular lobes are used. Instead, we take advantage of the general form of typical BRDFs and sample according to a mixture of 70% diffuse, and 30% high-gloss (Blinn exponent of 200). This places more importance on specular directions and enables low-noise results for the full range of glossy to near-mirror BRDFs in a practical editing session.

Tensor update: For each random path, we need to accumulate a contribution to the appropriate tensor element. In effect, we are computing coefficients in a symbolic polynomial representation of the basis BRDFs (in the spirit of symbolic rendering by [Séquin and Smyrl 1989]). In our case, we have chosen bases that do not overlap, and therefore a given path requires updating exactly one tensor element. The j index in equation 20 is determined by the basis function that contains the configuration of incoming and outgoing directions at the last intersection point (in Fig. 8 this is x_4). The outgoing direction for the bounce to the eye ($x_4 - e$) partitions the space of incoming directions into bands corresponding to our different box-basis functions (a, b, and c in Fig. 8). In the example, band c contains the incoming path direction, which determines j. More generally, we would choose the j for which $b_j(\omega_i, \omega_o)$ is non-zero. In the case of the diffuse approximation for further bounces, we only care about the objects containing the further bounces (the indices $m_{N-1}...m_1$ in equation 20). In Fig. 8, these are DCB. The more slowly decaying BRDF series with multiple specular bounces would use a similar band selection for the second bounce from the eye, as for the first.

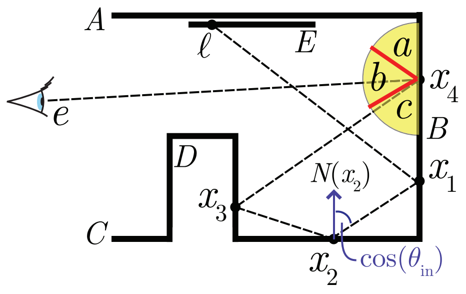

Figure 8: Consider a path $\vec{X}$ from the light (ℓ) to the eye (e). The light hits objects B, C, D, B before reaching the eye. At the final bounce (x_4), the view direction defines how the basis functions of B's BRDF divide the incoming light directions. Of the three bases (a, b, and c), the configuration of the last bounce places it in c. Therefore, this path contributes to $T^4_{Bc,D,C,B}$.

Tensor values involve a standard Monte Carlo sum,

$$T^N_{w_N w_{N-1} \ldots w_1}(x) = \frac{1}{Q} \sum_i \frac{f(\vec{X}_i)}{p(\vec{X}_i)}, \tag{21}$$

where Q is the number of paths through pixel x, and the sum runs only over paths $\vec{X}_i$ that correspond to the specific subscripts (bands and objects) in T^N. $f(\vec{X}_i)$ is the contribution of $\vec{X}_i$, and $p(\vec{X}_i)$ is the probability of generating it.

Path Contribution: In standard path tracing, the path contribution $f(\vec{X}_i)$ is the direct visible lighting at x_1, multiplied by the product of BRDFs (corresponding to $\mathbf{K}$) and cosine terms at each intersection (the visibility in $\mathbf{G}$ is already considered when creating valid paths). In our case, we must instead multiply by the appropriate *basis* BRDFs.

For the first bounce from the eye, we use

$$\tilde{b}_j(e, x_N, x_{N-1}) = b_j^{m_N(x_N)}(\omega_i, \omega_o) \cos \theta_i, \tag{22}$$

where $\omega_i(x_N, x_{N-1})$ and $\theta_i(x_N, x_{N-1})$ depend on the direction of the incident ray, and $\omega_o(e, x_N)$ on the outgoing view direction. For the slowly decaying series, a very similar form can be used for the second bounce from the eye, simply considering a lower-frequency $\tilde{b}_j(x_N, x_{N-1}, x_{N-2})$. For the other bounces, we use the single diffuse basis:

$$D(x_n, x_{n-1}) = \frac{1}{\pi} \cos(\theta_i(x_n, x_{n-1})). \tag{23}$$

Finally, $f(\vec{X})$ is a product of the terms at each bounce. For the diffuse approximation for further bounces, this is

$$f(\vec{X}) = \tilde{b}_j^{m_N(x_N)}(\omega_i, \omega_o) D(x_{N-1}, x_{N-2}) \ldots D(x_1, \ell) E(\ell). \tag{24}$$

Optimizations: Our precomputation is essentially the same complexity as rendering a single image with MCPT. Moreover, many standard path tracing optimizations can still be applied. For example, we have adapted irradiance caching [Ward et al. 1988]—instead of generating paths that terminate at the light source, we find the direct lighting at the last surface point x_1. We cache the irradiance in a preprocess that samples visibility on a grid within each triangle.

5.2 Rendering in Real Time

We now focus on the runtime rendering computation for each pixel. For compactness of notation, this subsection will deal primarily with the diffuse approximation for further bounces, but more slowly decaying series use very similar methods, and are discussed briefly at the end.

To simplify notation, we denote the tensor as $T^N_{jz}(x)$, where the single "super-index" z is a short-hand for writing out $m_{N-1}...m_1$ in equation 20 (viewed as an index, $z \in [1, Z = (M)^{N-1}]$). Similarly, we also denote the product of diffuse equivalents d_m of each object by $\overline{d}_z$,

$$\begin{aligned} z &\equiv \{m_{N-1}, \cdots, m_n, \cdots, m_1\} \\ \overline{d}_z &\equiv d_{m_{N-1}} d_{m_{N-2}} \cdots d_{m_n} \cdots d_{m_2} d_{m_1}. \end{aligned} \tag{25}$$

Note that z represents a list of indices, while $\overline{d}_z$ is a single number, corresponding to the product of the albedos d_m.

Finally, we can adapt equation 14 for rendering,

$$B^N(x) = \sum_{j=1}^{J} \sum_{z=1}^{Z} T^N_{jz}(x)\; c_j \overline{d}_z. \tag{26}$$

During the edit, the user specifies the BRDF coefficients c_j (either directly by editing a curve, or implicitly by editing a parametric model). The diffuse equivalents d (and hence $\overline{d}_z$) are then computed as described in appendix B. Finding the c_j and $\overline{d}_z$ occurs once per frame for the whole image. Using the precomputed $T^N_{jz}(x)$, the double summation in equation 26 must now be evaluated at each pixel.

Object Freezing: Equation 26 requires $O(JZ)$ operations per pixel. On modern hardware, this is fast, but still not real-time (requiring a couple of seconds per update). To reduce complexity, we observe that a user edits the BRDF of only one object at a time. We use a run-time precomputation that performs the bulk of the calculations in Equation 26 by temporarily "freezing" the BRDFs of the remaining objects.

Recall that $\overline{d}_z$ represents a multi-variable polynomial in the d's of the objects in the scene. For example, if we have $z = \{1, 3, 2, 1, 5, 3\}$, $\overline{d}_z = (d_1)^2 d_2 (d_3)^2 d_5$. However, if all but one of the d's are fixed, this becomes just a single-variable polynomial in the unfrozen d of the object being edited. For example, if all but object 1 are "frozen", we can define a constant $A = d_2(d_3)^2 d_5$, so that $\overline{d}_z$ becomes a simple quadratic polynomial, $\overline{d}_z = A \cdot (d_1)^2$ in only the edited variable d_1.

To implement this scheme more formally, we need a helper function $n(z, i)$ that tells us how many times a given edited object i appears in z. In the above example, $n(z, i) = 2$ (for $i = 1$) that tells us $\overline{d}_z$ is a quadratic polynomial in d_i alone. We can also define the constant A more formally as $A = \overline{d}_z/(d_i)^n$. Finally, we compute at run-time a new tensor of coefficients at each pixel, where each row represents a single-variable polynomial in d_i,

$$F^i_{jn}(x) = \sum_N \sum_z \begin{cases} n(z,i) \neq n: & 0 \\ n(z,i) = n: & T^N_{jz}(x) \frac{\overline{d}_z}{(d_i)^n} \end{cases} \tag{27}$$

Our real-time rendering step is now a direct evaluation of

$$B(x) = \sum_{n=0}^{\overline{N}-1} (d_i)^n \sum_{j=1}^{J} c_j F^i_{jn}(x). \tag{28}$$

For each power $(d_i)^n$, we simply evaluate a dot-product $c_j F_j$, essentially as in a standard linear PRT formulation.

The computation in equation 27 requires $O(JZ)$ operations per pixel, comparable to simply evaluating equation 26 directly once. This requires a short (usually 5-10 seconds) mode switch each time the user begins editing a different object. The real-time rendering in equation 28 is now $O(J)$ (the number of bounces $\overline{N}$ is a small constant, usually four.)

Two further optimizations are possible. In a practical editing session, the coefficients c_j change slowly from frame to frame, especially if we transform into a wavelet representation. This temporal coherence can be directly exploited using the incremental wavelet rendering method described in [Ben-Artzi et al. 2006]. Finally, if we are rendering a pixel of an object that is not being edited, the c_j do not change at all. (Note however, that the object's appearance will still be affected because of global illumination.) This makes it possible to further precompute $V^i_n = \sum_j F^i_{jn} c_j$, reducing the cost to evaluating a simple polynomial in d_i.

Slower Decaying Series: We briefly describe the generalization to more slowly decaying series, as in Sec. 4.3. The general rendering operation of Equation 26 is now best described as a triple-summation, since we are dealing with three distinct representations of R: R^{J_N}, $R^{J_{N-1}}$, and R^1.

$$B^N(x) = \sum_{j=1}^{J_N} \sum_{w=1}^{MJ_{N-1}} \sum_{z=1}^{Z} T^N_{jwz}(x)\; c_j \hat{c}_w \overline{d}_z, \tag{29}$$

with $\hat{c}_w$ denoting the lower-frequency BRDF coefficients for the second bounce (J_{N-1} bases on each of the M objects).

Object freezing is a bit more difficult, theoretically requiring the creation of a three-dimensional F^i_{jkn}, where $j \in [1, J_N]$, $k \in [1, J_{N-1}]$, $n \in [0, \overline{N}{-}2]$. In practice, we think of the j as one dimension, and $k' \equiv kn$ as the other.

5.3 Extensions

We briefly describe two important practical extensions.

Objects with Fixed BRDFs: Large scenes can contain many small objects that cause an exponential increase in memory requirements. Such scenes usually do not require editing the BRDFs of *all* objects. When designing the materials in a room, one typically does not want to change the BRDFs of small "placeholder" objects like books or toys. We extend our algorithm by implementing the ability to fix the BRDFs of certain objects at precomputation. Note that their shading is still updated, based on global illumination from other surfaces. This should also not be confused with run-time object freezing above, which occurs temporarily during an editing session.

In precomputation, instead of using diffuse equivalents D or BRDF bases $\tilde{b}$, we must use the full known BRDF $\rho^m(\omega_i, \omega_o) \cos\theta_i$ for reflections from fixed object m. Rendering is unchanged for editable objects, since there are no new BRDF bases. For the fixed object, we still use $T^N_z(x)$ and multiply by the diffuse albedos of editable objects. However, the BRDF bases (and index j) need not be considered. In Fig. 1, the table and its legs have fixed BRDFs.

Spatial Weight Maps: So far, we have focused on BRDF effects. Spatial variation can be handled with textures to modulate the BRDF over an object's surface. If we do not seek to edit them, the textures can be directly incorporated into the BRDF basis functions (as the multiplicative $H^m(x)$ terms in equation 5). Finally, while we have discussed a single BRDF per object for clarity, our framework and implementation can handle multiple co-located BRDFs for each object. If we also seek to modify the spatial blending of BRDFs, as we do for the floor in Fig. 1, we can simply modulate the directly viewed surfaces in image-space by the multiplicative spatial blending weights (weight maps) or texture. For global illumination, we are concerned only with the low-frequency behavior of the weight maps or textures, and we modulate the diffuse equivalent albedos by the average value of the weight map for each BRDF layer.

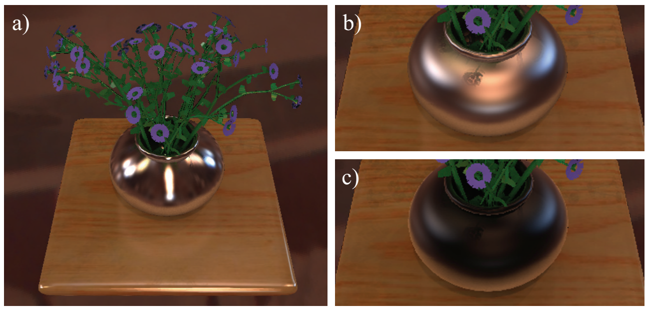

Figure 9: *a) original, b) closeup after anisotropic edit of vase in (a), c) closeup after fresnel effect strongly increased for (b).*

6 Results

Section 6.1 briefly discusses the types of edits performed on the scenes in Figs. 1, 4, 6, 9 and 11, and the global illumination effects involved. Then, Sec. 6.2 gives performance results for precomputation time and memory usage, while Sec. 6.3 discusses rendering frame rates.

6.1 Editing and Visual effects

Cornell Box: Figure 6 shows the Cornell box, where we edit the parameters of a Blinn-Phong and diffuse reflectance model. In going from (a) to (b), we make the back wall glossy with correct interreflections of the nearby scene, and change the color of the inner box, demonstrating accurate color bleeding. Note that even the simple color adjustment was not possible interactively with global illumination in previous methods.[*]

Teatray: Figure 4 shows a teatray scene. The teapot has a Cook-Torrance BRDF with specular and diffuse components, and the handles and tray a Blinn-Phong model. While only a single set of BRDFs for the objects is shown in Fig. 4 for brevity, we can freely edit the teapot, tray and handles in real-time, using any of the BRDF series shown in the figure (extended to three or four bounces).

Vase: Figure 9 shows a variety of flexible BRDFs possible within our system. The flowers are diffuse, while the stems are a diffuse+specular BRDF to enable a glossy coating. The table has diffuse and specular BRDFs, with the diffuse shown textured in Fig. 9. The vase uses a diffuse and a specular layer. The specular BRDF is an Ashikhmin-Shirley BRDF with fixed exponent of 225, but adjustable ratio to adjust the direction and amount of anisotropy (see Appendix A). A second specular layer allows for edits to the Fresnel term. Appendix A describes how to enable Fresnel control for any BRDF via a second additive layer.[†]

Room: Figure 1 shows one potential application of our system to design interiors. In this case, indirect light is critical, since a large part of the scene, like the back wall and much of the couch, would be black with direct lighting only. Most of the indirect illumination is low-frequency, so we use our diffuse approximation for further bounces. We only use three bounces, with a BRDF series $(64, 1, 1)$, since we found that the fourth bounce contributed relatively little to the qualitative appearance of the scene.

This is a complex scene with 8 objects (6 editable), environment lighting from the windows, and a variety of materials including measured reflectance and textures, which can all be edited. Our system renders interactive feedback with global illumination, enabling the user to correctly perceive and edit materials to design the interior of the room.

[*]Since the Cornell Box was designed for verifiable comparison to PBRT, it was precomputed with the floor and back wall using Blinn-Phong with 64 bands, while the other objects use 2 bands to represent a pure diffuse BRDF with editable albedo.

[†]The glossy layers of the stems and table use 64 bands. The Ashikhmin-Shirley layers of the vase use 256 bands. All diffuse layers use 2 bands.

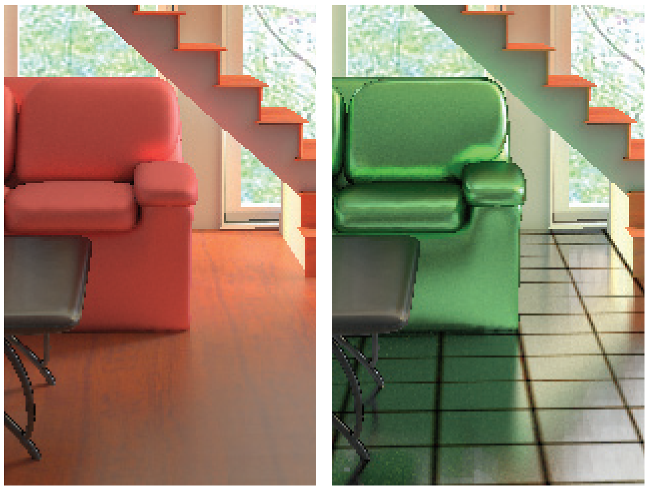

Figure 10: *Closeups for Fig. 1. Note the color bleeding of the couch onto the stairs. Note also the glossy reflection of the couch in the shiny floor on the right, which is a third-bounce effect (since the bottom of the couch is lit only by indirect lighting).*

The effects of global illumination are clearly seen in the closeup views of Fig. 10, where the couch color bleeds onto the stairs. Notice also the glossy reflection of the green couch in the highly shiny floor on the right. The lower portion of the couch is lit only by indirect illumination, so this is actually a glossy reflection of indirect light, and needs at least 3 bounces to simulate properly.[‡]

Ring: Figure 11 shows how our system can be used even to choose materials to create the desired complex global illumination effects like caustics. In this case, we use a slower series decay $(64, 16)$ that includes two specular bounces, to obtain accurate indirect specular reflections. Our system interactively updates both the caustics on the floor from the ring, and the reflection of the floor in the ring accurately, as the BRDFs are edited. Note that the sharpness of the caustics and indirect reflections are maintained even though the second bounce BRDF is still quite low frequency. Without our system, it would be quite difficult to interactively select the glossiness of the plane and ring, to explore possible appearances of the caustics.

6.2 Precomputation Times and Memory Usage

Table 2 describes precomputation times and memory usage. The rows show the scenes (including multiple BRDF series for the teatray). The image resolutions were chosen primarily to fit the resolution of the accompanying video—the teatrays were computed at lower resolution for faster testing and comparison to PBRT renders (which took hours to generate low-noise still images at these resolutions).

Precomputation Time: The precomputation time (on an Intel Xeon 3.2GHz 64 bit machine) ranges from one to several hours, depending linearly on the number of path tracing samples used, and also varying with the number of point lights to sample the environment. Interestingly, these wall clock times are about as fast (and sometimes faster than) for standard PBRT to render a single (uneditable) image of the scene— this is because the complexity of our precomputation is essentially the same as rendering a single image with path tracing. Thus, our precomputation times, while large, are comparable to those for high quality global illumination image synthesis methods.

[‡]Different objects in the room were computed with different BRDF parameterizations and resolutions. The couch and floor have 64 bands for specular BRDFs. The floor also has a diffuse layer with 2 bands. The stairs and sills use 16 bands for glossy BRDFs. The walls and ceiling have only a diffuse BRDF with 2 bands, allowing color and intensity edits, but not glossy.

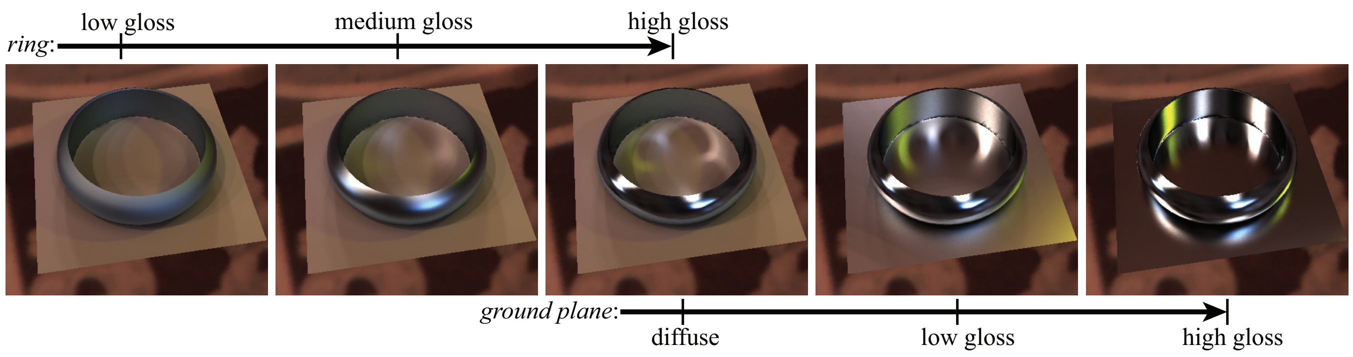

Figure 11: *To obtain accurate indirect reflections, we enable second bounce glossy reflections. We show results with an implementation that uses $J = 64$ for the first bounce from the eye and $J = 16$ for the second. With this approximation, we can not only get caustics on the floor from the ring, but also see the reflection of the floor in the ring accurately, as shown in the rightmost image.*

scene specifications				precomputation			storage per path-length (MB)					rendering	
name	*resolution*	*M*	*BRDF series*	*samples*	*lights*	*time(h:m)*	*1*	*2*	*3*	*4*	*total*	*freeze*	*fps*
Cornell box	450×450	6	64,1,1,1*	12K	81/81	5:58	17	104	365	975	1461	10s	22
Vase	512×512	4	64,1,1,1†	4K	10K/400	5:29	65	262	637	—	964	8s	19
Room	640×480	6	64,1,1‡	8K	14K/800	9:25	27	165	563	—	755	7s	20
Teatrays	320×240	3	128,1,1,1	10K	10K/2K	1:07	13	37	75	125	250	3s	25
			16,8,1,1			1:42	2	37	113	333	485	3s	16
			128,8,1			2:10	13	301	853	—	896	15s	5
Ring	400×400	2	64,16	4K	4K/400	6:15	20	607	—	—	627	16s	6

Table 2: *Precomputation time, storage of precomputed data structures, and rendering time for each of our scenes. The scene specification includes image resolution, number of objects (M), and the BRDF series. Precomputation lists the number of samples per pixel for path tracing, and the number of point lights used to sample the environment for direct/indirect lighting (we usually use fewer samples for indirect illumination). Storage is shown separately (in MB) for each path-length, or number of bounces. Rendering time indicates the time for "object freezing" when selecting a single object to edit, and for real-time rendering.*

Memory Usage: The memory usage grows for each bounce, since there are more polynomial terms in T^N (as shown in Fig. 5 (left)). The growth is relatively slow, being a factor of about 3 for higher bounces. Nevertheless, the highest (usually fourth) bounce requires more than half the total memory. This is an interesting direction for future work, since it is usually the darkest and most amenable to compression. However, in our experiments with direct quantization and wavelet compression, we were not easily able to find a scheme that was free of image artifacts. The problem is different from compression for relighting, since we visualize the BRDF and image directly, making artifacts more apparent. We instead simply drop zero values, and use RGBE compression. At the end, our total precomputed data structures are in the range of several hundred MB. While this is large, it is comparable, for instance, to all-frequency relighting approaches on similar resolution images.

Note that the storage sizes shown in Table 2 are larger than the theoretical runtime memory requirements. Due to our object freezing step, only the smaller F^i_{jn} in equation 27 needs to be in main memory. We can avoid preloading all of the data at the cost of higher object-switch times.

In comparing the different BRDF series for the teatray, $(128, 8, 1)$ requires the most memory, because of the extra factor of 8 for second bounce BRDF bases (as opposed to $(16, 8, 1, 1)$ and $(128, 1, 1, 1)$). To keep memory consumption reasonable, we limit to 3 bounces rather than 4 as with the other series. These numbers are comparable to the ring, that also uses 16 bases in the second bounce. (We similarly limit the number of bounces in the ring to two $(64, 16)$).

6.3 Rendering Speed

For simplicity, we pursue a purely software implementation, although the simple dot-product form of equation 28 indicates that GPU implementations, similar to those recently developed for relighting should also be applicable.

As can be seen in the video and the last column of table 2, our software method achieves frame rates of 15-25fps on most scenes. The time for "object-freezing" when we switch from one object to another for editing is about 5-10 seconds, and isn't disruptive to typical editing sessions, as seen in the video. The rendering and mode switch times are somewhat larger when there are more BRDF bases for the second bounce (one teatray example and ring), on account of the additional complexity. However, they are still interactive, and our video demonstrates interactive editing of the ring scene. In summary, our results and video for the first time show practical real-time BRDF editing, as interactive feedback is rendered with global illumination.

7 Conclusions and Future Work

This paper has described a complete theoretical analysis and practical implementation of a real-time rendering method, which enables interactive editing of BRDFs with global illumination effects. We expect significant applications to design in computer graphics, where the artist can now interactively specify material properties in the final scene, with complex lighting, shadows and interreflections.

In the process, we develop a new precomputation-based framework, that can handle nonlinear effects involving multivariable polynomials for multiple bounces. Our contributions include a general theoretical framework for expressing global illumination as a precomputation-based interactive rendering process based on reflection and geometric operators, an analysis of computational complexity to develop tractable approximations, and effective precomputation and rendering methods and extensions. It is likely that many of these insights could be used in other contexts, like PRT for relighting, or even offline global illumination.

More generally, we have proposed a new and efficient method for *symbolically rendering* an image. Instead of accumulating each path into the color of a single pixel, that path is effectively stored in symbolic form, including the product of all BRDF terms encountered along it. Paths involving the same BRDF terms are accumulated into the appropriate tensor coefficient. This symbolic approach is likely to have broad applications in other domains where we seek to interactively edit or explore the space of material parameters, such as animation, simulation and geometric modeling.

References

Arvo, J., Torrance, K., and Smits, B. 1994. A framework for the analysis of error in global illumination algorithms. In *SIGGRAPH 94*, 75–84.

Ashikhmin, M., Premoze, S., and Shirley, P. 2000. A mirofacet-based BRDF generator. In *SIGGRAPH 00*, 65–74.

Barbic, J., and James, D. 2005. Real-time subspace integration of St. Venant-Kirchoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 982–990.

Ben-Artzi, A., Overbeck, R., and Ramamoorthi, R. 2006. Real-time BRDF editing in complex lighting. *ACM Transactions on Graphics (SIGGRAPH 06) 25*, 3, 945–954.

Cohen, M., and Wallace, J. 1993. *Radiosity and Realistic Image Synthesis.* Academic Press.

Colbert, M., Pattanaik, S., and Krivnek, J. 2006. Brdf-shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Apps. 26*, 1.

Dorsey, J., Arvo, J., and Greenberg, D. 1995. Interactive design of complex time dependent lighting. *IEEE Computer Graphics and Applications 15*, 2 (March), 26–36.

Durand, F., Holzschuch, N., Soler, C., Chan, E., and Sillion, F. X. 2005. A frequency analysis of light transport. *ACM Transactions on Graphics (SIGGRAPH 05) 24*, 3, 1115–1126.

Hasan, M., Pellacini, F., and Bala, K. 2006. Direct to indirect transfer for cinematic relighting. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3, 1089–1097.

Kajiya, J. 1986. The rendering equation. In *SIGGRAPH 86*.

Kontkanen, J., Turquin, E., Holzschuch, N., and Sillion, F. 2006. Wavelet radiance transport for real-time indirect lighting. In *EuroGraphics Symposium on Rendering*.

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 25*, 3, 735–745.

Lensch, H. P. A., Kautz, J., Goesele, M., Heidrich, W., and Seidel, H.-P. 2001. Image-based reconstruction of spatially varying materials. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, 103–114.

McCool, M., Ang, J., and Ahmad, A. 2001. Homomorphic factorization of BRDFs for high-performance rendering. In *SIGGRAPH 01*, 171–178.

Nayar, S., Krishnan, G., Grossberg, M., and Raskar, R. 2006. Fast separation of direct and global components of a scene using high frequency illumination. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3, 935–944.

Ng, R., Ramamoorthi, R., and Hanrahan, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM TOG (SIGGRAPH 2003) 22*, 3, 376–381.

Ngan, A., Durand, F., and Matusik, W. 2005. Experimental analysis of BRDF models. In *EGSR*, 117–126.

Pharr, M., and Humphreys, G. 2004. *Physically Based Rendering: From Theory to Implementation.* Morgan Kaufmann.

Ramamoorthi, R., and Hanrahan, P. 2001. A signal-processing framework for inverse rendering. In *SIGGRAPH 01*, 117–128.

Schlick, C. 1994. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum 13*, 3, 233–246.

Séquin, C. H., and Smyrl, E. K. 1989. Parameterized ray tracing. In *Computer Graphics (SIGGRAPH 89)*, vol. 23.

Sloan, P., Kautz, J., and Snyder, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM TOG (SIGGRAPH 02) 21*, 3.

Sun, W., and Mukherjee, A. 2006. Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3, 477–487.

Wang, R., Tran, J., and Luebke, D. 2006. All-frequency relighting of glossy objects. *ACM TOG 25*, 2, 293–318.

Ward, G., Rubinstein, F., and Clear, R. 1988. A ray tracing solution for diffuse interreflection. In *SIGGRAPH 88*, 85–92.

Appendix A: BRDF Basis Functions

Our framework is general for any choice of linear BRDF basis functions, but benefits from those tailored to the natural space in which material edits occur. In practice, we use the 1D-reparameterized box basis functions of [Ben-Artzi et al. 2006],

$$\rho(\omega_i,\omega_o) = \rho_q(\omega_i,\omega_o) f(\gamma) \quad ; \quad f = \sum_j^J c_j b_j(\gamma(\omega_i,\omega_o)), \qquad (30)$$

where f is the 1D editable factor, and ρ_q is the quotient BRDF that captures more complex, but uneditable behavior like normalization constants and the $\mathcal{GAF}$. f can be set directly by the user by editing a 1D curve, or computed by the system based on the user setting parameters of analytic BRDFs. The form above involves an appropriate 1D parameterization γ of the BRDF's 4D domain. Some examples of γ are: θ_{half}, θ_{diff}, θ_{in}, and θ_{out}.

Some BRDFs have two factors, such as Cook Torrance with $\gamma = \theta_{half}$ to control specular behavior, and $\gamma = \theta_{diff}$ to control the Fresnel effect. We only use a single factor for practical reasons, to keep memory requirements manageable. However, we show below that many of the important "two-curve" edits can be achieved with a single factor, by careful use of the quotient BRDF.

The Fresnel effect is the most common use of the θ_{diff} factor. If we use the Schlick [1994] approximation, a BRDF that includes a Fresnel term (e.g. Cook-Torrance) becomes

$$\rho = \rho_q f(\theta_h)(F + (1-F)(1-\cos\theta_d)^5). \qquad (31)$$

F is a function of the wavelength (color channel) and *index of refraction* only. This allows us to define ρ as the sum of two BRDFs, each with just one editable factor but different ρ_q,

$$\begin{aligned} \rho &= \rho_q f_1 + \rho_{q2} f_2 \quad &;& \quad f_1 = F f(\theta_h) \\ \rho_{q2} &= (1-\cos\theta_d)^5 \rho_q \quad &;& \quad f_2 = (1-F) f(\theta_h) \end{aligned} \qquad (32)$$

At runtime, F is evaluated based on the user's choice of *index of refraction*, and f_1 and f_2 are set via the user interface. The two BRDFs are computed and summed (just as a specular and diffuse layer would be summed) to yield an accurate composite.

Anisotropy is the result of an elongated highlight. As presented in [Ben-Artzi et al. 2006], two factors can be used to adjust the width of the highlight along the tangent and binormal directions. If we know the overall specularity of the material, we can separate the Ashikhmin-Shirley BRDF into a quotient that captures the width of the highlight, and a 1D factor that controls the ratio of the elongation in the two perpendicular directions:

$$\rho_{AS} = \rho_q (\cos\theta_h)^{n_u \cos\phi_h} (\cos\theta_h)^{(r n_u)\sin\phi_h} \quad ; \quad r \equiv n_v / n_u \qquad (33)$$

$$\rho_{AS} = \rho_{qu}(\omega_i,\omega_o)(\gamma_r(\omega_h,\omega_o))^r \qquad (34)$$

$$\gamma_r = (\cos\theta_h)^{n_u \sin\phi_h} \quad ; \quad \rho_{qu} = \rho_q (\cos\theta_h)^{n_u \cos\phi_h} \qquad (35)$$

A similar single-factor form can be obtained if the amount and direction of anisotropy (r) is known, and only the width of the highlight needs to be adjusted.

Appendix B: Equivalent Albedo

We choose the equivalent albedo to match the average BRDF value, or more exactly, the output power for a uniform incident radiance field. This also corresponds formally to choosing the best perturbed $\mathbf{K}$ operator, as in [Arvo et al. 1994]. In other words,

$$\begin{aligned} d &= \frac{1}{\pi}\int\int \rho(\omega_i,\omega_o)\,\cos\theta_i\cos\theta_o\,d\omega_i d\omega_o \qquad (36) \\ &= \frac{1}{\pi}\sum_j c_j \int\int \rho_q(\omega_i,\omega_o) b_j(\gamma(\omega_i,\omega_o))\,\cos\theta_i\cos\theta_o\,d\omega_i, d\omega_o. \end{aligned}$$

The term in the integral now depends only on known quantities—the quotient BRDFs and the basis functions, and can therefore be evaluated by dense Monte Carlo sampling (this needs to be done only once for a given parameterization, not even for each scene). Call this e_j. Finally, at run-time, we simply need to compute

$$d = \frac{1}{\pi}\sum_j c_j e_j, \qquad (37)$$

with the predetermined e_j and the dynamically chosen c_j.