Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments

Peter-Pike Sloan
Microsoft Research
ppsloan@microsoft.com

Jan Kautz
Max-Planck-Institut für Informatik
jnkautz@mpi-sb.mpg.de

John Snyder
Microsoft Research
johnsny@microsoft.com

Abstract

We present a new, real-time method for rendering diffuse and glossy objects in low-frequency lighting environments that captures soft shadows, interreflections, and caustics. As a preprocess, a novel global transport simulator creates functions over the object's surface representing transfer of arbitrary, low-frequency incident lighting into *transferred radiance* which includes global effects like shadows and interreflections from the object onto itself. At run-time, these transfer functions are applied to actual incident lighting. Dynamic, local lighting is handled by sampling it close to the object every frame; the object can also be rigidly rotated with respect to the lighting and vice versa. Lighting and transfer functions are represented using low-order spherical harmonics. This avoids aliasing and evaluates efficiently on graphics hardware by reducing the shading integral to a dot product of 9 to 25 element vectors for diffuse receivers. Glossy objects are handled using matrices rather than vectors. We further introduce functions for radiance transfer from a dynamic lighting environment through a preprocessed object to neighboring points in space. These allow soft shadows and caustics from rigidly moving objects to be cast onto arbitrary, dynamic receivers. We demonstrate real-time global lighting effects with this approach.

Keywords: Graphics Hardware, Illumination, Monte Carlo Techniques, Rendering, Shadow Algorithms.

1. Introduction

Lighting from area sources, soft shadows, and interreflections are important effects in realistic image synthesis. Unfortunately, general methods for integrating over large-scale lighting environments [8], including Monte Carlo ray tracing [7][21][25], radiosity [6], or multi-pass rendering that sums over multiple point light sources [17][27][36], are impractical for real-time rendering.

Real-time, realistic global illumination encounters three difficulties – it must model the complex, spatially-varying BRDFs of real materials (*BRDF complexity*), it requires integration over the hemisphere of lighting directions at each point (*light integration*), and it must account for bouncing/occlusion effects, like shadows, due to intervening matter along light paths from sources to receivers (*light transport complexity*). Much research has focused on extending BRDF complexity (e.g., glossy and anisotropic reflections), solving the light integration problem by representing incident lighting as a sum of directions or points. Light integration thus tractably reduces to sampling an analytic or tabulated BRDF at a few points, but becomes intractable for large light sources. A second line of research samples radiance and preconvolves it with kernels of various sizes [5][14][19][24][34]. This solves the light integration problem but ignores light transport complexities like shadows since the convolution assumes the incident radiance is unoccluded and unscattered. Finally, clever techniques exist to simulate more complex light transport, especially shadows. Light integration becomes the problem; these techniques are impractical for very large light sources.

Our goal is to better account for light integration and light transport complexity in real-time. Our compromise is to focus on low-frequency lighting environments, using a low-order spherical harmonic (SH) basis to represent such environments efficiently without aliasing. The main idea is to represent how an object scatters this light onto itself or its neighboring space.

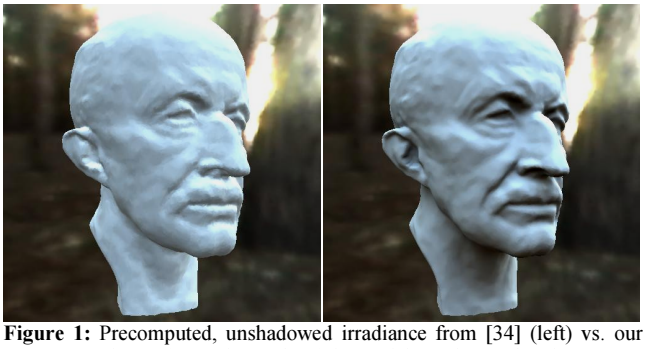


Figure 1: Precomputed, unshadowed irradiance from [34] (left) vs. our precomputed transfer (right). The right model can be rendered at 129Hz with self-shadowing and self-interreflection in **any** lighting environment.

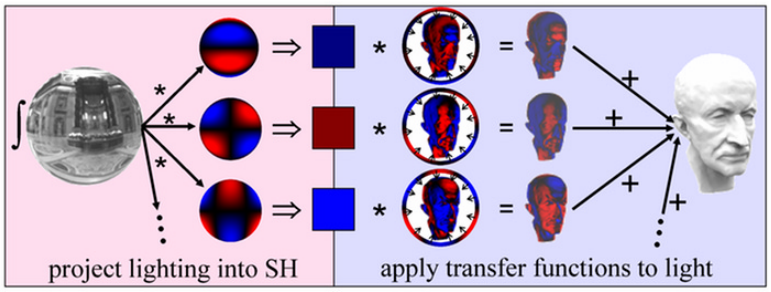
To describe our technique, assume initially we have a convex, diffuse object lit by an infinitely distant environment map. The object's shaded "response" to its environment can be viewed as a *transfer function*, mapping incoming to outgoing radiance, which in this case simply performs a cosine-weighted integral. A more complex integral captures how a concave object shadows itself, where the integrand is multiplied by an additional transport factor representing visibility along each direction.

Our approach is to precompute for a given object the expensive transport simulation required by complex transfer functions like shadowing. The resulting transfer functions are represented as a dense set of vectors or matrices over its surface. Meanwhile, incident radiance need not be precomputed. The graphics hardware can dynamically sample incident radiance at a number of points. Analytic models, such as skylight models [33] or simple geometries like circles, can also be used.

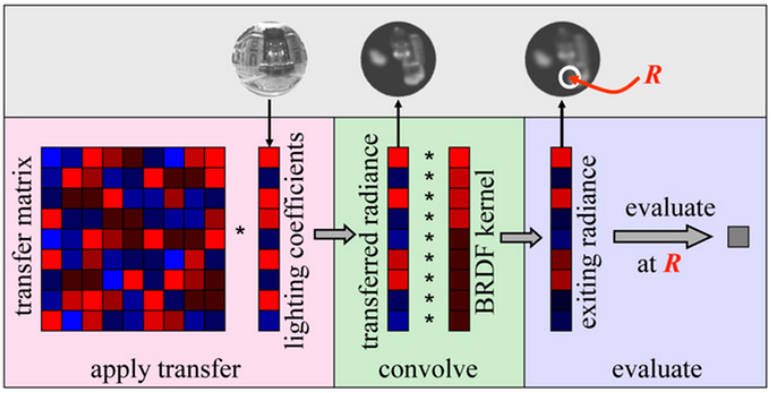
By representing both incident radiance and transfer functions in a linear basis (in our case, SH), we exploit the linearity of light transport to reduce the light integral to a simple dot product between their coefficient vectors (diffuse receivers) or a simple linear transform of the lighting coefficient vector through a small transfer matrix (glossy receivers). Low-frequency lighting environments require few coefficients (9-25), enabling graphics hardware to compute the result in a single pass (Figure 1, right). Unlike Monte-Carlo and multi-pass light integration methods, our run-time computation stays constant no matter how many or how big the light sources, and in fact relies on large-scale, smooth lighting to limit the number of SH coefficients necessary.

We represent complex transport effects like interreflections and caustics in the transfer function. Since these are simulated as a preprocess, only the transfer function's basis coefficients are affected, not the run-time computation. Our approach handles both surface and volume-based geometry. With more SH coefficients, we can even handle glossy (but not highly specular) receivers as well as diffuse, including interreflection. 25 coefficients suffice for useful glossy effects. In addition to transfer from a rigid object to itself, called *self-transfer*, we generalize the technique to *neighborhood-transfer* from a rigid object to its neighboring space, allowing cast soft shadows, glossy reflections, and caustics on dynamic receivers, see Figure 7.

Overview As a preprocess, a global illumination simulator is run over the model that captures how it shadows and scatters light onto itself. The result is recorded as a dense set of vectors (diffuse case) or matrices (glossy case) over the model. At run-time (Figure 2), incident radiance is first projected to the SH basis. The model's field of transfer vectors or matrices is then applied to the lighting's coefficient vector. If the object is **diffuse**, a **transfer vector** at each point on the object is dotted with the lighting's coefficients to produce correctly self-scattered shading. If the object is **glossy**, a **transfer matrix** is applied to the lighting coefficients to produce the coefficients of a spherical function representing self-scattered incident radiance at each point. This function is convolved with the object's BRDF and then evaluated at the view-dependent reflection direction to produce the final shading.



diffuse surface self-transfer



glossy surface self-transfer

Figure 2: Self-Transfer Run-Time Overview. Red signifies positive SH coefficients and blue, negative. For a diffuse surface (top row), the SH lighting coefficients (on the left) modulate a field of transfer vectors over the surface (middle) to produce the final result (right). A transfer vector at a particular point on the surface represents how the surface responds to incident light at that point, including global transport effects like self-shadowing and self-interreflection.. For a glossy surface (bottom row), there is a matrix at each point on the model instead of a vector. This matrix transforms the lighting coefficients into the coefficients of a spherical function representing transferred radiance. The result is convolved with the model's BRDF kernel and evaluated at the view-dependent reflection direction *R* to yield the result at one point on the model.

2. Related Work

Scene relighting precomputes a separate global illumination solution per light source as we do; linear combinations of the results then provide limited dynamic effects. Early work [2][11] adjusts intensities of a fixed set of sources and is not intended to fit general lighting environments. Nimeroff, et al. [33] precompute a "steerable function" basis for general skylight illumination on a fixed view. Their basis, essentially the spherical monomials, is related to the SH by a linear transformation and thus shares some of its properties (e.g., rotational invariance) but not others (e.g., orthonormality). Teo, et al. [40] generalize to non-infinite sources, using principal component analysis to reduce the basis set. Our work differs by computing a transfer field over the object's surface in 3D rather than over a fixed 2D view to allow viewpoint changes. Dobashi, et al. [10] use the SH basis and transfer vector fields over surfaces to allow viewpoint change but restrict lighting changes to the directional intensity distribution of an existing set of non-area light sources in diffuse scenes. Debevec, et al. [9] relight faces using a directional light basis. Real-time rendering requires a fixed view.

Shadow maps, containing depths from the light source's point of view, were first used by Williams [43] to simulate point light source shadows. Many extensions of the basic technique, some suitable for real-time rendering, have since been described: *percentage-closer filtering* [35], which softens shadow edges, *layered depth maps* [26] and *layered attenuation maps* [1], which more accurately simulate penumbra shape and falloff, and *deep shadow maps* [29], which generalize the technique to partially transparent and volume geometry. All these techniques assume point or at least localized light sources; shadowing from larger light sources has been handled by multi-pass rendering that sums over a light source decomposition into points or small sources [17][27][36]. Large light sources become very expensive.

Another technique [39] uses FFT convolution of occluder projections for soft shadowing with cost independent of light source size. Only shadows between pre-segmented clusters of objects are handled, making self-shadows on complex meshes difficult.

Finally, *accessibility shading* [32] is also based on precomputed global visibility, but is a scalar quantity that ignores changes in lighting direction.

Methods for nonlocal lighting on micro-geometry include the *horizon map* [4][31], which efficiently renders self-shadowing from point lights. In [20], this technique is tailored to graphics hardware and generalized to diffuse interreflections, though interreflection change due to dynamic lighting is still not real-time. By precomputing a higher-dimensional texture, *polynomial texture maps* [30] allow real-time interreflection effects as well as shadowing. A similar approach using a steerable basis for directional lighting is used in [3]. Like our approach, these methods precompute a simple representation of a transfer function, but one based on directional light sources and thus requiring costly multi-pass integration to simulate area lights. We compute self-transfer directly on each preprocessed 3D object rather than mapping it with 2D micro-geometry textures, allowing more global effects. Finally, our neighborhood transfer extends these ideas to cast shadows, caustics, and reflections.

Caching onto diffuse receivers is useful for accelerating global illumination. Ward et. al. [41] perform caching to simulate diffuse interreflection in a ray tracer. Photon maps [21] also cache but perform forward ray tracing from light sources rather than backwards from the eye, and handle specular bounces in the transport (as does our approach). We apply this caching idea to real-time rendering, but cache a transfer function parameterized by a SH lighting basis rather than scalar irradiance.

Precomputed transfer using light-field remapping [18] and dynamic ray tracing [16] has been used to achieve highly specular reflections and refractions. We apply a similar precomputed, per-object decomposition but designed instead for soft shadows and caustics on nearly diffuse objects in low-frequency lighting. Irradiance volumes [15] allow movement of diffuse receivers in precomputed lighting. Unlike our approach, lighting is static and the receiver's effect on itself and its environment is ignored.

Spherical harmonics have been used to represent incident radiance and BRDFs for offline rendering and BRDF inference [4] [38][42]. Westin, et al. [42] use a matrix representation for 4D BRDF functions in terms of the SH basis identical to our transfer matrix. But rather than the BRDF, we use it to represent global and spatially varying transport effects like shadows. The SH basis has also been used to solve ambiguity problems in computer vision [12] and to represent irradiance for rendering [34].

3. Review of Spherical Harmonics

Definition Spherical harmonics define an orthonormal basis over the sphere, $\boldsymbol{S}$, analogous to the Fourier transform over the 1D circle. Using the parameterization

$$s = (x, y, z) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta),$$

the basis functions are defined as

$$Y_l^m(\theta,\varphi) = K_l^m\, e^{im\varphi} P_l^{|m|}(\cos\theta),\ l \in \boldsymbol{N},\ -l \le m \le l$$

where P_l^m are the associated Legendre polynomials and K_l^m are the normalization constants

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}\ .$$

The above definition forms a complex basis; a real-valued basis is given by the simple transformation

$$y_l^m = \begin{cases} \sqrt{2}\,\mathrm{Re}(Y_l^m), & m > 0 \\ \sqrt{2}\,\mathrm{Im}(Y_l^m), & m < 0 \\ Y_l^0, & m = 0 \end{cases} = \begin{cases} \sqrt{2}\,K_l^m \cos(m\varphi) P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}\,K_l^m \sin(-m\varphi) P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

Low values of l (called the *band* index) represent low-frequency basis functions over the sphere. The basis functions for band l reduce to polynomials of order l in x, y, and z. Evaluation can be done with simple recurrence formulas [13][44].

Projection and Reconstruction Because the SH basis is orthonormal, a scalar function f defined over $\boldsymbol{S}$ can be *projected* into its coefficients via the integral

$$f_l^m = \int f(s)\, y_l^m(s)\, ds \qquad (1)$$

These coefficients provide the n-th order reconstruction function

$$\tilde{f}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m\, y_l^m(s) \qquad (2)$$

which approximates f increasingly well as the number of bands n increases. Low-frequency signals can be accurately represented with only a few SH bands. Higher frequency signals are bandlimited (i.e., smoothed without aliasing) with a low-order projection.

Projection to n-th order involves n^2 coefficients. It is often convenient to rewrite (2) in terms of a singly-indexed vector of projection coefficients and basis functions, via

$$\tilde{f}(s) = \sum_{i=1}^{n^2} f_i\, y_i(s) \qquad (3)$$

where $i=l(l+1)+m+1$. This formulation makes it obvious that evaluation at s of the reconstruction function represents a simple dot product of the n^2-component coefficient vector f_i with the vector of evaluated basis functions $y_i(s)$.

Basic Properties A critical property of SH projection is its rotational invariance; that is, given $g(s) = f(Q(s))$ where Q is an arbitrary rotation over $\boldsymbol{S}$ then

$$\tilde{g}(s) = \tilde{f}(Q(s)) \qquad (4)$$

This is analogous to the shift-invariance property of the 1D Fourier transform. Practically, this property means that SH projection causes no aliasing artifacts when samples from f are collected at a rotated set of sample points.

Orthonormality of the SH basis provides the useful property that given any two functions a and b over $\boldsymbol{S}$, their projections satisfy

$$\int \tilde{a}(s)\, \tilde{b}(s)\, ds = \sum_{i=1}^{n^2} a_i\, b_i\ . \qquad (5)$$

In other words, integration of the product of bandlimited functions reduces to a dot product of their projection coefficients.

Convolution We denote convolution of a circularly symmetric kernel function $h(z)$ with a function f as $h * f$. Note that h must be circularly symmetric (and hence can be defined as a simple function of z rather than s) in order for the result to be defined on $\boldsymbol{S}$ rather than the higher-dimensional rotation group $\boldsymbol{SO}(3)$. Projection of the convolution satisfies

$$(h * f)_l^m = \sqrt{\frac{4\pi}{2l+1}}\, h_l^0\, f_l^m = \alpha_l^0\, h_l^0\, f_l^m\ . \qquad (6)$$

In other words, the coefficients of the projected convolution are simply scaled products of the separately projected functions. Note that because h is circularly symmetric about z, its projection coefficients are nonzero only for m=0. The convolution property provides a fast way to convolve an environment map with a hemispherical cosine kernel, defined as $h(z) = \max(z, 0)$, to get an irradiance map [34], for which the h_l^0 are given by an analytic formula. The convolution property can also be used to produce prefiltered environment maps with narrower kernels.

Product Projection Projection of the product of a pair of spherical functions $c(s) = a(s)b(s)$ where a is known and b unknown can be viewed as a linear transformation of the projection coefficients b_j via a matrix $\hat{a}$:

$$\begin{aligned} c_i &= \int a(s) \left(b_j\, y_j(s)\right) y_i(s)\, ds \\ &= \left(\int a(s)\, y_i(s)\, y_j(s)\, ds\right) b_j = \left(a_k \int y_i(s)\, y_j(s)\, y_k(s)\, ds\right) b_j = \hat{a}_{ij}\, b_j \end{aligned} \qquad (7)$$

where summation is implied over the duplicated j and k indices. Note that $\hat{a}$ is a symmetric matrix. The components of $\hat{a}$ can be computed by integrating the triple product of basis functions using recurrences derived from the well-known Clebsch-Gordan series [13][44]. It can also be computed using numerical integration without SH-projecting the function a beforehand. Note that the product's order n projection involves coefficients of the two factor functions up to order $2n$-1.

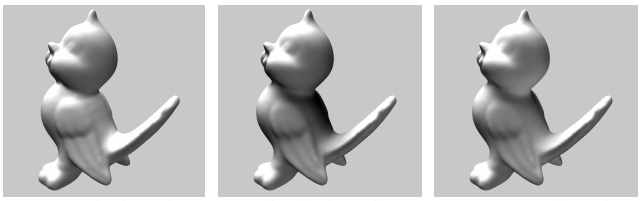
Rotation A reconstruction function rotated by Q, $\tilde{f}(Q(s))$, can be projected into SH using a linear transformation of f's projection coefficients, f_i. Because of the rotation invariance property, this linear transformation treats the coefficients in each band independently. The most efficient implementation is achieved via a *zyz* Euler angle decomposition of the rotation Q, using a fairly complicated recurrence formula [13][44]. Because we deal only with low-order functions, we have implemented their explicit rotation formulas using symbolic integration.

4. Radiance Self-Transfer

Radiance self-transfer encapsulates how an object $\boldsymbol{O}$ shadows and scatters light onto itself. To represent it, we first parameterize incident lighting at points $p \in \boldsymbol{O}$, denoted $L_p(s)$, using the SH basis. Incident lighting is therefore represented as a vector of n^2 coefficients $(L_p)_i$. We sample the lighting **dynamically** and **sparsely** near the surface, perhaps at only a single point. The assumption is that lighting variation over $\boldsymbol{O}$ not due to its own presence is small (see Section 6.1). We also **precompute** and store **densely** over $\boldsymbol{O}$ transfer vectors or matrices.

A *transfer vector* $(M_p)_i$ is useful for diffuse surfaces and represents a linear transformation on the lighting vector producing scalar exit radiance, denoted L'_p, via the inner product

$$L'_p = \sum_{i=1}^{n^2} (M_p)_i\, (L_p)_i\ . \qquad (8)$$



(a) unshadowed (b) shadowed (c) interreflected

Figure 3: Diffuse Surface Self-transfer.

In other words, each component of $(M_p)_i$ represents the linear influence that a lighting basis function $(L_p)_i$ has on shading at p.

A *transfer matrix* $(\mathcal{M}_p)_{ij}$ is useful for glossy surfaces and represents a linear transformation on the lighting vector which produces projection coefficients for an entire spherical function of transferred radiance $L'_p(s)$ rather than a scalar; i.e.,

$$(L'_p)_i = \sum_{j=1}^{n^2} (\mathcal{M}_p)_{ij} (L_p)_j \quad . \tag{9}$$

The difference between incident and transferred radiance is that $L'_p(s)$ includes shadowing/scattering effects due to the presence of $\boldsymbol{O}$ while $L_p(s)$ represents incident lighting assuming $\boldsymbol{O}$ was removed from the scene. Components of $(\mathcal{M}_p)_{ij}$ represent the linear influence of the j-th lighting coefficient of incident radiance $(L_p)_j$ to the i-th coefficient of transferred radiance $(L'_p)_i$. The next sections derive transfer vectors for diffuse surfaces and transfer matrixes for glossy surfaces due to self-scattering on $\boldsymbol{O}$.

4.1 Diffuse Transfer [transfer vector for known normal]

First assume $\boldsymbol{O}$ is diffuse. The simplest transfer function at a point $p \in \boldsymbol{O}$ represents *unshadowed diffuse transfer*, defined as the scalar function

$$T_{DU}(L_p) = (\rho_p / \pi) \int L_p(s)\, H_{Np}(s)\, ds$$

producing exit radiance which is invariant with view angle for diffuse surfaces. Here, ρ_p is the object's albedo at p, L_p is the incident radiance at p assuming $\boldsymbol{O}$ was removed from the scene, N_p is the object's normal at p, and $H_{Np}(s) = \max(N_p \bullet s, 0)$ is the cosine-weighted, hemispherical kernel about N_p. By SH-projecting L_p and H_{Np} separately, equation (5) reduces T_{DU} to an inner product of their coefficient vectors. We call the resulting factors the *light function*, L_p, and *transfer function,* M_p. In this first simple case, $M_p^{DU}(s) = H_{Np}(s)$.

Because N_p is known, the SH-projection of the transfer function $(M_p^{DU})_i$ can be precomputed, resulting in a *transfer vector*. In fact, storing is unnecessary because a simple analytic formula yields it given N_p. Because M_p^{DU} is inherently a low-pass filter, second-order projection (9 coefficients) provides good accuracy in an arbitrary (even non-smooth) lighting environment [34].

To include shadows, we define *shadowed diffuse transfer* as

$$T_{DS}(L_p) = (\rho_p / \pi) \int L_p(s)\, H_{Np}(s)\, V_p(s)\, ds$$

where the additional visibility function, $V_p(s) \to \{0,1\}$, equals 1 when a ray from p in the direction s fails to intersect $\boldsymbol{O}$ again (i.e., is unshadowed). As with unshadowed transfer, we decompose this integral into two functions, using an SH-projection of L_p and the transfer function

$$M_p^{DS}(s) = H_{Np}(s) V_p(s) \quad . \tag{10}$$

Separately SH-projecting L_p and M_p again reduces the integral in T_{DS} to an inner product of coefficient vectors.

Transfer is now nontrivial; we precompute it using a transport simulator (Section 5), storing the resulting transfer vector $(M_p)_i$ at many points p over $\boldsymbol{O}$. Unlike the previous case, second-order projection of M_p^{DS} may be inaccurate even for smooth lighting environments since V_p can create higher-frequency lighting locally, e.g., by self-shadowing "pinholes". 4-th or 5-th order projection provides good results on typical meshes in smooth lighting environments.

Finally, to capture diffuse interreflections as well as shadows, we define *interreflected diffuse transfer* as

$$T_{DI}(L_p) = T_{DS}(L_p) + (\rho_p / \pi) \int \bar{L}_p(s)\, H_{Np}(s)\, (1 - V_p(s))\, ds$$

where $\bar{L}_p(s)$ is the radiance from $\boldsymbol{O}$ itself in the direction s. The difficulty is that unless the incident radiance emanates from an infinitely-distant source, we don't actually know $\bar{L}_p(s)$ given the incident radiance only at p because $\bar{L}_p$ depends on the exit radiance of points arbitrarily far from p and local lighting varies over $\boldsymbol{O}$. If lighting variation is small over $\boldsymbol{O}$ then $\bar{L}_p$ is well-approximated as if $\boldsymbol{O}$ were everywhere illuminated by L_p. T_{DI} thus depends linearly on L_p and can be factored as in the previous two cases into a product of two projected functions: one light-dependent and the other geometry-dependent.

Though precomputed interreflections must make the assumption of spatially invariant incident lighting over $\boldsymbol{O}$, simpler shadowed transfer need not. The difference is that shadowed transfer depends only on incident lighting at p, while interreflected transfer depends on many points $q \neq p$ over $\boldsymbol{O}$ at which $L_q \neq L_p$. Thus, as long as the incident radiance field is sampled finely enough (Section 6.1), local lighting variation can be captured and shadowed transfer will be correct.

The presence of $\bar{L}$ makes it hard to explicitly denote the transfer function for interreflections, $M_p^{DI}(s)$. We will see how to compute its projection coefficients numerically in Section 5.

4.2 Glossy Transfer [transfer matrix for unknown direction]

Self-transfer for glossy objects can be defined similarly, but generalizes the kernel function to depend on a (view-dependent) reflection direction R rather than a (fixed) normal N. Analogous to the H kernel from before, we model glossy reflection as the kernel $G(s,R,r)$ where a scalar r defines the "glossiness" or broadness of the specular response. We believe it is possible to handle arbitrary BRDFs as well using their SH projection coefficients [38] but this remains for future work.

We can then define the analogous three glossy transfer functions for the unshadowed, shadowed, and interreflected cases as

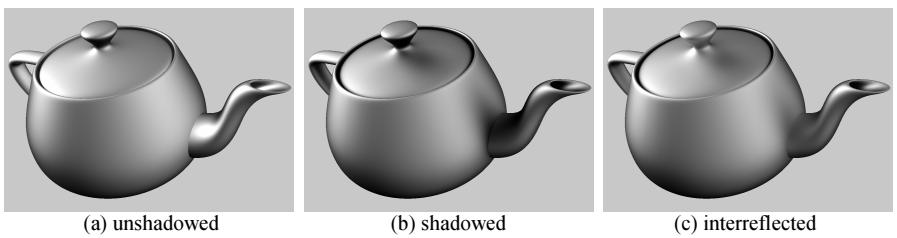
$$T_{GU}(L_p, R, r) = \int L_p(s)\, G(s, R, r)\, ds$$

$$T_{GS}(L_p, R, r) = \int L_p(s)\, G(s, R, r)\, V_p(s)\, ds$$

$$T_{GI}(L_p, R, r) = T_{GS}(L_p) + \int \bar{L}_p(s)\, G(s, R, r)\, (1 - V_p(s))\, ds$$

which output scalar radiance in direction R as a function of L_p and R, quantities both unknown at precomputation time. Since transfer is no longer solely a function of s, it can't be reduced to a simple vector of SH coefficients

Instead of parameterizing scalar transfer by R and r, a more useful decomposition is to transfer the incident radiance $L_p(s)$ into a whole sphere of transferred radiance, denoted $L'_p(s)$. Assuming the glossy kernel G is circularly symmetric about R (i.e., a simple Phong-like model) $L'_p(s)$ can then be convolved with $G^*_r(z) = G(s, (0,0,1), r)$ and evaluated at R to produce the final



(a) unshadowed (b) shadowed (c) interreflected

Figure 4: Glossy Surface Self-transfer.

result (see bottom of Figure 2, and further details in Section 6).

Transfer to L'_p can now be represented as a matrix rather than a vector. For example, glossy shadowed transfer is

$$\mathcal{M}_p^{GS}(L_p, s) = L_p(s) V_p(s) \qquad (11)$$

a linear operator on L_p whose SH-projection can be represented as the symmetric matrix $\hat{V}_p$ via equation (7). Even with smooth lighting, more SH bands must be used for L'_p as $\boldsymbol{O}$'s glossiness increases; non-square matrices (e.g., 25×9) mapping low-frequency lighting to higher-frequency transferred radiance are useful under these conditions. For shadowed glossy transfer (but not interreflected), an alternative still uses a vector rather than a matrix to represent $\mathcal{M}_p^{GS}$ by computing the product of V_p with L_p on-the-fly using the tabulated triple product of basis functions in equation (7). We have not yet implemented this alternative.

4.3 Limitations and Discussion

An important limitation of precomputed transfer is that material properties of $\boldsymbol{O}$ influencing interreflections in T_{DI} and T_{GI} (like albedo or glossiness) must be "baked in" to the preprocessed transfer and can't be changed at run-time. On the other hand, the simpler shadowed transfer without interreflection does allow run-time change and/or spatial variation over $\boldsymbol{O}$ of the material properties. Error arises if blockers or light sources intrude into $\boldsymbol{O}$'s convex hull. $\boldsymbol{O}$ can only move rigidly, not deform or move one component relative to the whole. Recall also the assumption of low lighting variation over $\boldsymbol{O}$ required for correct interreflections.

Finally, note that diffuse transfer as defined produces radiance *after leaving the surface*, since it has already been convolved with the cosine-weighted normal hemisphere, while glossy transfer produces radiance *incident on the surface* and must be convolved with the local BRDF to produce the final exit radiance. It's also possible to bake in a fixed BRDF for glossy $\boldsymbol{O}$, making the convolution with G unnecessary at run-time but limiting flexibility.

5. Precomputing Radiance Self-Transfer

As a preprocess, we perform a global illumination simulation over an object $\boldsymbol{O}$ using the SH basis over the infinite sphere as emitters. Our light gathering solution technique is a straightforward adaptation of existing approaches [7][25] and could be accelerated in many ways; its novelty lies in how it parameterizes the lighting and collects the resulting integrated transfers. Note that all integrated transfer coefficients are *signed* quantities.

The simulation is parameterized by an n-th order SH projection of the unknown sphere of incident light L; i.e., by n^2 unknown coefficients L_i. Though the simulation results can be computed independently for each L_i using the SH basis function $y_i(s)$ as an emitter, it is more efficient to compute them all at once. The infinitely-distant sphere L will be replaced at run-time by the actual incident radiance field around $\boldsymbol{O}$, L_p.

An initial pass simulates direct shadows from paths leaving L and reaching sample points $p \in \boldsymbol{O}$. In subsequent passes, interreflections are added, representing paths from L that bounce a number of times off $\boldsymbol{O}$ before arriving at p (Lp, LDp, $LDDp$, etc.). In each pass, energy is gathered to every sample point p. Large emitters (i.e., low-frequency SH basis) make a gather more efficient then a shooting-style update [6]. Note that this idea of caching onto diffuse (or nearly diffuse) receivers is not new [21][41].

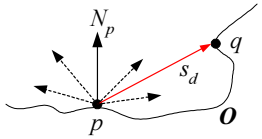
To capture the sphere of directions at sample points $p \in \boldsymbol{O}$, we generate a large (10k-30k), quasi-random set of directions $\{s_d\}$, $s_d \in \boldsymbol{S}$. We also precompute evaluations for all the SH basis functions at each s_d. The s_d are organized in hierarchical bins formed by refining an initial icosahedron with 1→2 bisection into equal-area spherical triangles (1→4 subdivision does not lead to equal area triangles on the sphere as it does in the plane). We use 6 to 8 subdivision levels, creating 512 to 2048 bins. Every bin at each level of the hierarchy contains a list of the s_d within it.

In the first pass, for each $p \in \boldsymbol{O}$, we cast shadow rays in the hemisphere about p's normal N_p, using the hierarchy to cull directions outside the hemisphere. We tag each direction s_d with an occlusion bit, $1 - V_p(s_d)$, indicating whether s_d is in the hemisphere and intersects $\boldsymbol{O}$ again (i.e., is self-shadowed by $\boldsymbol{O}$). An occlusion bit is also associated with the hierarchical bins, indicating whether any s_d within it is occluded. Self-occluded directions and bins are tagged so that we can perform further interreflection passes on them; completely unoccluded bins/samples receive only direct light from the environment.

For diffuse surfaces, at each point $p \in \boldsymbol{O}$ we further compute the transfer vector by SH-projecting M_p from (10). For glossy surfaces, we compute the transfer matrix by SH-projecting $\mathcal{M}_p$ from (11). In either case, the result represents the radiance collected at p, parameterized by L. SH-projection to compute the transfers is performed by numerical integration over the direction samples s_d, summing into an accumulated transfer using the following rules:

diffuse:	$(M_p)_i^0 \mathrel{+}= \left(\rho_p/\pi\right) V_p(s_d)\, H_N(s_d)\, y_i(s_d)$
glossy:	$(\mathcal{M}_p)_{ij}^0 \mathrel{+}= V_p(s_d)\, y_j(s_d)\, y_i(s_d)$
Transfer integration over s_d [shadow pass, iteration 0]	

The superscript 0 refers to the iteration number. The vector M_p or matrix $\mathcal{M}_p$ at each point p is initialized to 0 before the shadow pass, which then sums over all s_d at every p. The rules are derived using equation (1) for diffuse transfer integration, and equation (7) for glossy transfer integration.



Later interreflection passes traverse the bins having the occlusion bit set during the shadow pass. Instead of shadow rays, they shoot rays that return transfer from exiting illumination on $\boldsymbol{O}$. If the ray (p,s_d) intersects another point $q \in \boldsymbol{O}$ (where q is closest to p), we sample the radiance exiting from q in the direction $-s_d$. The following update rules are used, where the superscript b is the bounce pass iteration:

diffuse:	$(M_p)_i^b \mathrel{+}= \left(\rho_p/\pi\right)\left(1 - V_p(s_d)\right)(M_q)_i^{b-1}\, H_N(s_d)$
glossy:	$(\mathcal{M}_p)_{ij}^b \mathrel{+}= \left(1 - V_p(s_d)\right) \left(\sum_k \alpha_k\, (G^*_{r_q})_k\, (\mathcal{M}_q)_{kj}^{b-1}\, y_k\left(\text{reflect}(-s_d, N_q)\right)\right) y_i(s_d)$
Transfer integration over s_d [interreflection passes, iteration b]	

As in the shadow pass, we begin by initializing transfer vectors or matrices to 0 before accumulating transfer over directions s_d. The diffuse rules are derived from the definition of T_{DI} and equation (1); glossy rules from the definition of T_{GI} and equations (6) and (7). The middle factor in the glossy transfer definition represents radiance emanating from q back to p from the previous bounce pass, b-1. Since $\mathcal{M}_q$ stores incident radiance, it must be convolved with $\boldsymbol{O}$'s BRDF at q to obtain exiting radiance in the $-s_d$ direction, yielding a summation over k. Recall that α_k is the k-th convolution coefficient, expressed in singly-indexed notation. The "reflect" operator simply reflects its first vector argument with respect to its second. We observe that equation (7) implies $(\mathcal{M}_p)_{ij}$ is a symmetric matrix for shadowed glossy transfer since it is formed by the product of two spherical functions; this is untrue for interreflected glossy transfer.

Interreflection passes are repeated until the total energy of a given pass falls below a threshold. For typical materials, it diminishes quite rapidly. The sum of transfers from all bounce passes then

accounts for interreflections. Our implementation simulates diffuse and glossy transfer at the same time.

A simple enhancement to this simulation allows mirror-like surfaces within $\boldsymbol{O}$. We do not record transfers on such surfaces. Instead, a ray striking a mirrored surface is always reflected and then propagated until a non-mirrored surface is reached. Thus our paths at successive iterations can be represented as ($L[S]^*p$, $L[S]^*D[S]^*p$, $L[S]^*D[S]^*D[S]^*p$, etc.), where D is a diffuse or glossy bounce and S is a specular one. This captures caustics onto diffuse or glossy receivers that respond dynamically to lighting change (Figure 9).

6. Run-time Rendering of Radiance Transfer

We now have a model $\boldsymbol{O}$ capturing radiance transfer at many points p over its surface, represented as vectors or matrices. Rendering $\boldsymbol{O}$ requires the following steps at run-time:

1. compute incident lighting $\{L_{Pi}\}$ at one or more sample points P_i near $\boldsymbol{O}$ in terms of the SH basis,
2. rotate these L_{Pi} to $\boldsymbol{O}$'s coordinate frame and blend them (see below) to produce a field of incident lighting L_p over $\boldsymbol{O}$, and
3. perform a linear transformation on $(L_p)_i$ at each point p on $\boldsymbol{O}$ to obtain exit radiance. This requires a dot product with $(M_p)_i$ for diffuse surfaces (equation (8)), or a matrix-vector multiplication with $(\mathcal{M}_p)_{ij}$ for glossy surfaces (equation (9)).
4. Glossy surfaces need a final step in which the radiance vector resulting from step 3 is convolved with $\boldsymbol{O}$'s BRDF at p, and then evaluated at the view-dependent reflection direction R.

Step 1 can load a precomputed environment map, evaluate analytic lighting models in software, or sample radiance using graphics hardware. Rotation for Step 2 is outlined in Section 3, and is done once per object, not for each p. It is necessary because transfer is stored using a common coordinate system for $\boldsymbol{O}$. If $\boldsymbol{O}$ is rigidly moving, it is more efficient to rotate the few radiance samples in L_{Pi} to align with $\boldsymbol{O}$ than it is to rotate $\boldsymbol{O}$'s many transfer functions. We currently perform this rotation in software.

For diffuse surfaces, a simple implementation of step 3 is to store the transfer vector per vertex and perform the dot product in a vertex shader. The transfer vectors can also be stored in texture maps rather than per-vertex and evaluated using a pixel shader. Since the coefficients are signed quantities not always in the [-1,1] range, DirectX 8.1 pixel shaders (V1.4) or their OpenGL counterpart (extension by ATI) must be used, since they provide a larger range of [-8,8]. Our pixel shader needs 8 instructions to perform the dot-product and stores L_P's coefficients in constant registers.

For colored environments or simulation of color bleeding on $\boldsymbol{O}$, three passes are required, each performing a separate dot-product for the r, g, and b channels. Otherwise a single pass suffices.

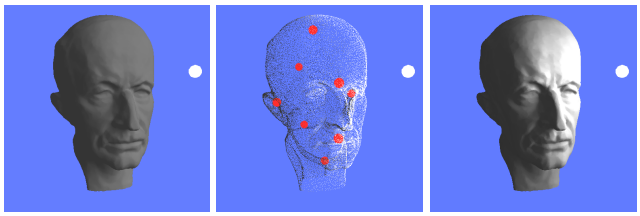
For glossy self-transfer, we perform the matrix transform from equation (9) in software because the transfer matrix is too big to be manipulated in either current vertex or pixel shaders. The result is $(L'_p)_i$ the SH coefficients of transferred radiance at points p over $\boldsymbol{O}$. Then in a pixel shader, we perform a convolution with a simple cosine-power (Phong lobe) kernel for G^* and evaluate the result in the reflection direction R. The result can be written

$$\sum_{i=1}^{n^2} \alpha_i \, G_i^* \left(\sum_{j=1}^{n^2} \left(\mathcal{M}_p\right)_{ij} \left(L_p\right)_j \right) y_i(R) \qquad (12)$$

We evaluate SH-projections up to n=5 on graphics hardware.

6.1 Spatial Sampling of the Incident Radiance Field

A simple and useful approach for dynamically sampling incident radiance is to sample it at $\boldsymbol{O}$'s center point. To handle local lighting variation over $\boldsymbol{O}$, a more accurate technique samples incident lighting at multiple points (Figure 5). A good set of sample points can be obtained using the ICP (iterated closest point) algorithm [28] as a preprocess, given a desired number of points as input. This produces a representative set of points P_i near $\boldsymbol{O}$ and distributed uniformly over it where incident lighting can be sampled at run-time. We can also precompute coefficients at each p over $\boldsymbol{O}$ that blend contribution from each of the resulting sampled radiance spheres L_{Pi} to produce an incident radiance field over $\boldsymbol{O}$, denoted previously by L_p.



(a) single sample (b) ICP points (c) multiple samples

Figure 5: ICP can be used to precompute good locations for sampling the incident radiance field over an object. Note the improved locality of lighting in (c) compared to (a) when the lighting is sampled at the 8 points in (b) rather than at the object center.

6.2 Sampling SH Radiance on Graphics Hardware

Graphics hardware is useful to capture the radiance samples $\{L_{Pi}\}$ in a dynamic scene. To do this, 6 images are rendered from each P_i corresponding to the 6 faces of the cube map spherical parameterization. $\boldsymbol{O}$ itself should be removed from these renderings. Cube map images can then be projected to their SH coefficients using the integral in equation (1), as was done in [4].

For efficiency, we precompute textures for the basis functions weighted by differential solid angle, $B_l^m(s) = y_l^m(s)\,ds(s)$, each evaluated over the cube map parameterization for s. The resulting integral then becomes a simple dot product of the captured samples of $L_P(s)$ with the textures $B_l^m(s)$.

Ideally, this computation would be performed on the graphics hardware. Precision issues and inability to do inner products in hardware force us to read back the sampled radiance images and project them in software. In this case, it is important to reduce the resolution of read-back images as much as possible.

Low-order SH projection can be computed with very low-resolution cube maps, assuming they have been properly bandlimited. For example, spherical signals already bandlimited to 6-th order can be projected using six 4×4 images with about 0.3% average-case squared error and about 1% worst-case squared error, where error is normalized by assuming unit-power signals (i.e., signals whose integrated square over the sphere is 1).[1] For 6×8×8 maps, this error reduces to 0.003% mean and 0.02% worst-case. Unfortunately, typical signals aren't spherically bandlimited. Another analysis shows that, assuming continuous bilinear reconstruction over the sampled 2D images, projection to 6-th order using 6×8×8 images yields 0.7% and 2% average and worst-case squared error, while 6×16×16 yields 0.2% and 0.5% squared error, and 6×32×32 yields 0.05% and 0.1% squared error.

We extract 6×16×16 images from the hardware. As is always true in point-sampled rendering, aliasing of the 2D images is still a problem because the above analysis uses bilinear reconstruction from point samples as the reference. To reduce aliasing, we supersample the cube map images by a factor of 2 in each dimension, and do a box-filtered decimation in hardware before reading back and projecting. The basis function textures are also supersampled and decimated in the same way as a preprocess. A radiance sample, including read-back and SH projection, takes about 1.16ms on a PIII-933 PC with an ATI Radeon 8500.

[1] More precisely, average-case error is the integrated squared difference between the reference and reconstruction signals, averaged over all unit-power signals. Worst-case error is the same integrated error, but for the worst-case unit-power signal.

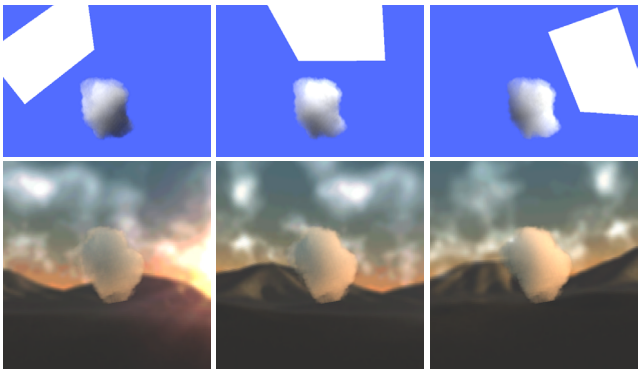


Figure 6: Volumetric self-transfer captures how this cloud model shadows itself. Lighting can be changed in real-time (first row). The same model can also be placed in other environments (second row).

7. Self-Transfer for Volumetric Models

Self-transfer on volumetric data uses the same framework as surfaces. The resulting precomputed model allows run-time changes to the lighting, with correct shadowing and interreflections in any low-frequency lighting environment (Figure 6).

Our simple simulator currently works only for diffuse volumes. As with surface transfer, a preprocessing step simulates lighting on the volume using the SH basis functions as emitters. For shadowed transfer without interreflection (i.e., direct shadowing), we gather energy from the emitter to every voxel p of the volume, attenuated by its path through the volume. The required numerical integration over directions s_d can be expressed as

$$(M_p)_i^0 \mathrel{+}= A(p \to p+Ds_d)\, y_i(s_d)$$

where $A(p \to q)$ is the volume's integrated attenuation along the path from p to q, and D is the distance until the ray (p,s_d) exits the volume. To include interreflections, we traverse every voxel p and forward-scatter its transfer along random directions s_d. The transfer is deposited to all voxels q along s_d until exiting the volume, using the rule

$$(M_q)_i^b \mathrel{+}= A(p \to q)\,(M_p)_i^{b-1}$$

More passes over the volume produce further indirect bounces.

Rendering is performed in the traditional way: by drawing slices through the 3D volume in back to front order using alpha blending to account for transparency. Each slice is a 2D image containing samples of the transfer vector. A pixel shader computes the dot-product between the lighting's coefficients and the transfer vector's required to shade each slice.

8. Radiance Neighborhood-Transfer

Neighborhood-transfer precomputes an object ***O***'s influence on its neighboring environment with respect to parameterized, low-frequency lighting. Transport simulation is identical to that for self-transfer in Section 5, but takes place with respect to points in a 3D space surrounding ***O***, not on it. At run-time, an arbitrary receiver ***R*** can be placed in this volume to capture shadows, reflections, and caustics cast by ***O*** onto ***R*** without knowing ***R*** in advance. For example, a moving vehicle ***O*** can cast shadows over a terrain ***R*** (Figure 7). Cast shadows and lighting also respond to lighting change; for example, moving the lights move soft shadows on ***R***. This generalizes irradiance volumes [15] by accounting for glossy transfer and allowing dynamic lighting.

Because ***R*** is unknown during the precomputation step, ***O***'s neighborhood volume must store a transfer matrix rather than a vector. This is true even for diffuse receivers, because we do not know in advance what ***R***'s normal will be. Our current implementation precomputes the transfer matrix $\mathcal{M}_p$ at each point within a simple 3D grid surrounding ***O***. At run-time, we perform the matrix transform from equation (9) in software at each point

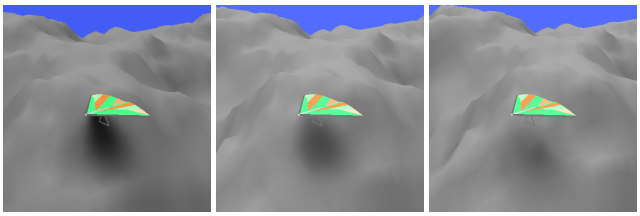


Figure 7: Neighborhood transfer captures how this hang glider blocks light to a volume of points below it. This allows cast soft shadow onto a bumpy terrain as the glider moves.

in the volume and upload the result to the graphics hardware. The result is a volume texture containing coefficients of transferred radiance $(L'_p)_i$ which is applied to ***R***.

Then in a pixel shader this transferred radiance is used to light the receiver. A diffuse receiver convolves the radiance with the cosine weighted hemisphere H^* using equation (6) and then evaluates the resulting SH projection at ***R***'s normal vector. Glossy receivers perform equation (12).

Receivers having precomputed self-transfer raise the difficulty that ***O*** and ***R*** do not share a common coordinate system. Thus, one of the two object's dense set of transfer samples must be dynamically rotated to align with the other's. The SH-rotation operation required is currently impractical for hardware evaluation. Improving hardware should soon ease this difficulty.

Compared to self-transfer, neighborhood-transfer incurs some additional approximation errors. Cast shadow or light from multiple neighborhood-transfer objects onto the same receiver is hard to combine. Local lighting variation not due to ***O*** or ***R***'s presence is also a problem; lighting must be fairly constant across ***O***'s entire neighborhood to provide accurate results. In particular, errors such as missing shadows will result when objects besides ***O*** and ***R*** intrude into ***O***'s neighborhood. ***O***'s neighborhood must also be large enough to encompass any cast shadow or light it may cast on ***R***. Nevertheless, neighborhood transfer captures effects impossible to obtain in real-time with previous methods.

9. Results

Full statistics are summarized in Table 1. We achieve real-time performance for all models except the transfer matrix ones (teapot, buddha, glider). For these models, multiplication with 25x25 or 9x25 transfer matrices over the surface in software forms the bottleneck. Real-time results can be achieved even for these models after first fixing the light (allowing the view to be moved) or the view (allowing the light to be changed) and represent the second and third "render rate" entries in the table after slashes. The reason is that fixing either the view (which then fixes the

model	transfer type	transfer shape	transfer sampling	preproc. time	render rate
head (fig 1)	DS	25-M	50k ver. mesh	1.1h	129
bird (fig 4)	DS	25-M	50k ver. mesh	1.2h	125
ring (fig 9)	DS	25-M	256x256 grid	8m	94
buddha_d (fig 11)	DS	25-M	50k ver. mesh	2.5h	125
buddha_g (fig 11)	GS	25x25-$\mathcal{M}$	50k ver. mesh	2.5h	3.6/16/125
tyra_d (fig 11)	DS	25-M	100k ver. mesh	2.4h	83
tyra_g (fig 11)	GS	25x25-$\mathcal{M}$	100k ver. mesh	2.4h	2.2/9.4/83
teapot (fig 5)	GS	25x25-$\mathcal{M}$	150k ver. mesh	4.4h	1.7/7.7/49
cloud (fig 6)	DV	25-M	32x32x32 vol.	15m	40
glider (fig 7)	N	9x25-$\mathcal{M}$	64x64x8 vol.	3h	4/120/4

Table 1: Results. Transfer types are DS=diffuse surface self-transfer, GS=glossy surface self-transfer, DV=diffuse volume self-transfer, and N=neighborhood transfer. Timings are on a 2.2GHz Pentium 4 with ATI Radeon 8500 graphics card. Render rates are in Hz.

reflection vector R) or the light L_p reduces the computation in (12) to a simple dot product, which can then be done in hardware. Fixed light rendering is slower than fixed view because the fixed light mode requires evaluation of the SH basis at a changing view vector, followed by a dot product, while fixed view requires only the dot product and is identical in performance to diffuse transfer.

Rendering quality can be judged from images in this paper (Figures 1 and 3-12) all of which were computed with the PC graphics hardware. Self-shadowing and interreflection effects are convincing and robust. No depth tolerances are required to prevent self-shadowing artifacts as they are with the standard shadow buffer technique [43]. Even when the lighting contains very high frequencies (Figure 9, top row of Figure 10, and Figure 12), pleasing images are produced without temporal artifacts but with some blurring of self-shadows; the result looks, and indeed is, identical to blurring the incident lighting.

Figure 10 compares shadowing results across different SH orders. Small light sources (top row) require more bands; larger ones are approximated very well with fewer bands. Using up to the quartic band (fifth order with 25 coefficients) provides good results and is a good match to today's graphics hardware. Note that quality is not dictated solely by how much energy is ignored in SH-projecting the lighting – diffuse and glossy objects are effective low-pass filters of incident radiance. With self-transfer effects though, the extent of this low-pass filtering depends on the object's geometry, varies spatially, and typically requires more than third order (quadratics), unlike unshadowed diffuse transfer [34].

Because of its rotational invariance (equation (4)), we consider the SH basis especially useful for our low-frequency lighting application compared to alternatives like spherical wavelets [37]. When dynamically sampling incident radiance, this property eliminates aliasing which would otherwise produce temporal artifacts, like shading "wobble", if projected to other bases with the same number of coefficients. Ringing or Gibbs phenomenon (oscillatory undershoot and overshoot in the reconstruction function) can be a problem when the lighting environment has significant energy near its highest represented band [10][42]. We notice ringing artifacts only on very simple models such as the ones in Figures 9 and 10; artifacts are masked on complex meshes. Of course, reducing lighting frequency by attenuating higher frequency bands, called "windowing", also reduces ringing (see Figure 10, columns f and g).

10. Conclusions and Future Work

Much important shading variation over a complex object such as a human face is due to itself. Precomputed radiance self-transfer is a general method for capturing the occlusion and scattering effects an object has on itself with respect to any low-frequency lighting environment. When the actual incident lighting is substituted at run-time, the resulting model provides global illumination effects like soft shadows, interreflections, and caustics in real-time. Using graphics hardware, incident lighting can be sampled every frame and at multiple points near the object allowing dynamic, local lighting. Neighborhood-transfer generalizes the concept by recording transfer over 3D space, allowing cast soft shadows and caustics onto arbitrary receivers.

In future work, we want to apply precomputed transfer to more sophisticated transport models, especially subsurface scattering [22]. We believe the smoothness of exiting radiance produced by this model makes it particularly suitable for SH-parameterized transfer. It would also be valuable to combine existing shadowing techniques with ours, by decomposing the scene's lighting into high and low frequency terms. Compression of transfer fields is an important but unaddressed problem. Extension to deformable objects like human characters could be achieved by parameterizing the precomputed self-transfer in the same way as the deformation, assuming the number of independent degrees of freedom remains manageable. Finally, we are interested in tracking fast-improving PC graphics hardware so that all computation, including transfer matrix transforms and SH-rotation, may eventually be performed on the GPU.

Acknowledgements: Thanks to Paul Debevec for his light probes (http://www.debevec.org), to the Stanford University Computer Graphics Laboratory for the happy Buddha model (http://www-graphics.stanford.edu/data/3Dscanrep), and Cyberware for the tyrannosaur model. Jason Mitchell and Michael Doggett of ATI and Matthew Papakipos of NVidia kindly provided graphics hardware. We also wish to acknowledge Michael Cohen for early discussions and Charles Boyd and Hans-Peter Seidel for support.

References

[1] Agrawala, M, Ramamoorthi, R, Heirich, A, and Moll, L, Efficient Image-Based Methods for Rendering Soft Shadows, SIGGRAPH '00, 375-384.

[2] Airey, J, Rohlf, J, and Brooks, F, Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments,1990 Symposium on Interactive 3D Graphics, 24(2), 41-50.

[3] Ashikhmin, M, and Shirley, P, Steerable Illumination Textures, ACM Transactions on Graphics, 2(3), to appear.

[4] Cabral, B, Max, N, and Springmeyer, R, Bidirectional Reflection Functions from Surface Bump Maps, SIIGRAPH '87, 273-281.

[5] Cabral, B, Olano, M, and Nemec, P, Reflection Space Image Based Rendering, SIGGRAPH '99, 165-170..

[6] Cohen, M, and Wallace, J, Radiosity and Realistic Image Synthesis, Academic Press Professional, Cambridge, 1993.

[7] Cook, R, Porter, T, and Carpenter, L, Distributed Ray Tracing, SIGGRAPH '84, 137-146.

[8] Debevec, P, Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photogaphy, SIGGRAPH '98, 189-198.

[9] Debevec, P, Hawkins, T, Tchou, C, Duiker, H, Sarokin, W, and Sagar, M, Acquiring the Reflectance Field of a Human Face, SIGGRAPH 2000, 145-156.

[10] Dobashi, Y, Kaneda, K, Nakatani, H, and Yamashita, H, A Quick Rendering Method Using Basis Functions for Interactive Lighting Design, Eurographics '95, 229-240.

[11] Dorsey, J, Sillion, F, and Greenberg, D, Design and Simulation of Opera Lighting and Projection Effects, SIGGRAPH '91, 41-50.

[12] D'Zmura, M, Shading Ambiguity: Reflection and Illumination. In Computational Models of Visual Processing (1991), Landy and Movshon, eds., MIT Press, Cambridge, 187-207.

[13] Edmonds, A, Angular Momentum in Quantum Mechanics, Princeton University, Princeton, 1960.

[14] Greene, N, Environment Mapping and Other applications of World Projections, IEEE CG&A, 6(11):21-29, 1986.

[15] Greger, G., Shirley, P, Hubbard, P, and Greenberg, D, The Irradiance Volume, IEEE Computer Graphics And Applications, 6(11):21-29, 1986.

[16] Hakura, Z, and Snyder, J, Realistic Reflections and Refractions on Graphics Hardware with Hybrid Rendering and Layered Environment Maps, Eurographics Workshop on Rendering, 2001, 289-300.

[17] Haeberli, P, and Akeley, K, The Accumulation Buffer: Hardware Support for High-Quality Rendering, SIGGRAPH '90, 309-318.

[18] Heidrich, W, Lensch, H, Cohen, M, and Seidel, H, Light Field Techniques for Reflections and Refractions, Eurographics Rendering Workshop 99,195-375.

[19] Heidrich, W, Seidel H, Realistic, Hardware-Accelerated Shading and Lighting, SIGGRAPH '99, 171-178.

[20] Heidrich, W, Daubert, K, Kautz, J, and Seidel, H, Illuminating Micro Geometry based on Precomputed Visibility, SIGGRAPH '00, 455-464.

[21] Jensen, H, Global Illumination using Photon Maps, Eurographics Workshop on Rendering 1996, 21-30.

[22] Jensen, H, Marschner, S, Levoy, M, and Hanrahan, P, A Practical Model for Subsurface Light Transport, SIGGRAPH '01, '511-518.

[23] Kautz, J, and McCool, M, Interactive Rendering with Arbitrary BRDFs using Separable Approximations, Eurographics Workshop on Rendering 99,.247-260.

[24] Kautz, J, Vazquez, P, Heidrich, W, and Seidel, H, A Unified Approach to Prefiltered Environment Maps, Eurographics Workshop on Rendering 2000, 185-196.

[25] Kajiya, J, The Rendering Equation, SIGGRAPH '86, 143-150.

[26] Keating, B, and Max, N, Shadow Penumbras for Complex Objects by Depth-Dependent Filtering of Multi-Layer Depth Images, Eurographics Rendering Workshop, 1996, pp.205-220.

[27] KELLER, A, Instant Radiosity, SIGGRAPH '97, 49-56.
[28] LINDE, Y, BUZO, A, AND GRAY, R, An algorithm for Vector Quantizer Design, IEEE Transactions on Communication COM-28, 1980,84-95.
[29] LOKOVIC, T, AND VEACH, E, Deep Shadow Maps, SIGGRAPH '00, pp.385-392.
[30] MALZBENDER, T, GELB, D, AND WOLTERS, H, Polynomial Texture Maps, SIGGRAPH '01, 519-528.
[31] MAX, N, Horizon Mapping: Shadows for Bump-Mapped Surfaces, The Visual Computer, July 1998, 109-117.
[32] MILLER, G, Efficient Algorithms for Local and Global Accessibility Shading, SIGGRAPH '94, 319-326.
[33] NIMEROFF, J, SIMONCELLI, E, AND DORSEY, J, Efficient Re-rendering of Natural Environments, Eurographics Workshop on Rendering 1994, 359-373.
[34] RAMAMOORTHI, R, AND HANRAHAN, P, An Efficient Representation for Irradiance Environment Maps, SIGGRAPH '01, 497-500.
[35] REEVES, W, SALESIN, D, AND COOK, R, Rendering Antialiased Shadows with Depth Maps, SIGGRAPH '87, '283-291.
[36] SEGAL, M, KOROBKIN, C, VAN WIDENFELT, R, FORAN, J, AND HAEBERLI, P, Fast Shadows and Lighting Effects Using Texture Mapping, SIGGRAPH '92, '249-252.
[37] SCHRÖDER, P, AND SWELDENS, W, Spherical Wavelets: Efficiently Representing the Sphere, SIGGRAPH '95, '161-172.
[38] SILLION, F, ARVO, J, WESTIN, S, AND GREENBERG, D, A Global Illumination Solution for General Reflectance Distributions, SIGGRAPH '91, 187-196.
[39] SOLER, C, AND SILLION, F, Fast Calculation of Soft Shadow Textures Using Convolution, SIGGRAPH '98, '321-332.
[40] TEO, P, SIMONCELLI, E, AND HEEGER, D, Efficient Linear Re-rendering for Interactive Lighting Design, October 1997 Report No. STAN-CS-TN-97-60, Stanford University, 1997.
[41] WARD, G, RUBINSTEIN, F, AND CLEAR, R, A Ray Tracing Solution for Diffuse Interreflection, SIGGRAPH '88, '85-92.
[42] WESTIN, S, ARVO, J, TORRANCE, K, Predicting Reflectance Functions from Complex Surfaces, SIGGRAPH '92, 255-264.
[43] WILLIAMS, L, Casting Curved Shadows on Curved Surfaces, SIGGRAPH '78, 270-274.
[44] ZARE, R, Angular Momentum: Understanding Spatial Aspects in Chemistry and Physics, Wiley, New York, 1987.

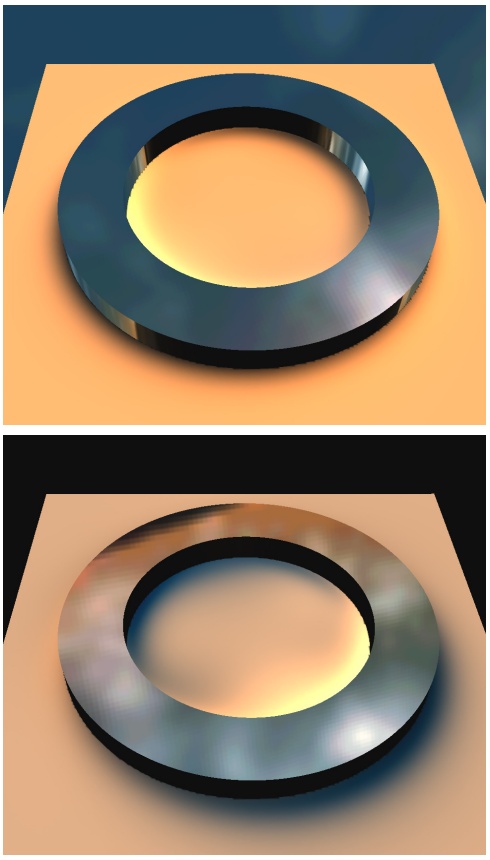


Figure 9: **Real-time, dynamic caustics and shadows using diffuse self-transfer.** The ring model is rendered with a traditional environment map; the ground plane was rendered using pre-computed self-transfer results from a caustic simulation combining both the specular ring and a diffuse ground (Section 5). A 25 component transfer vector was recorded over a 256x256 grid on the ground plane. These two images were generated by rotating an acquired environment around the model. A frame rate of 130Hz is obtained in our implementation .

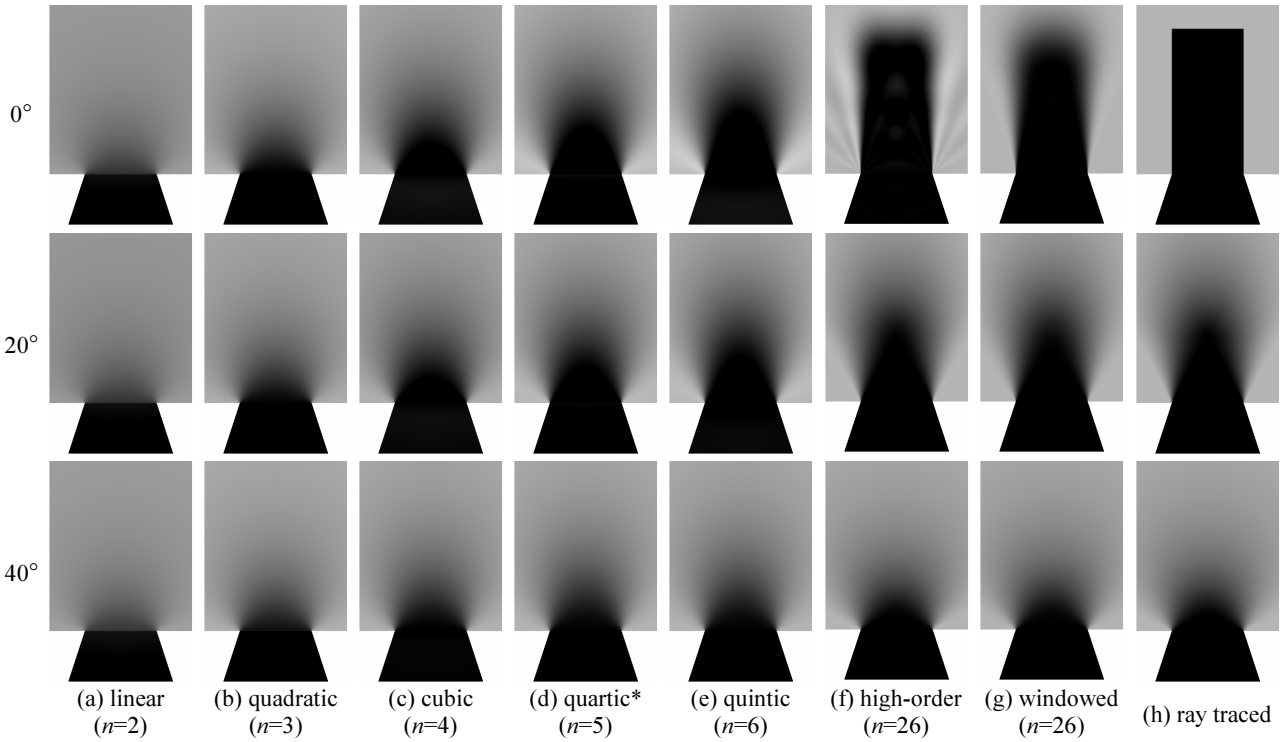


Figure 10: **Comparison of SH orders for representing diffuse self-transfer.** Shadows are cast onto a ground plane from a single polygon seen obliquely at bottom. Angular radius of a constant circular light used for illumination is shown at left. Higher-orders provide greater accuracy for small lights, but give rise to ringing (which is reduced by windowing in g). Note that n^2 coefficients are required for projection order n. We use the quartic projection (starred) for other result images in this paper.

(a) diffuse, unshadowed (b) diffuse, interreflected (c) glossy, unshadowed (d) glossy, interreflected

Figure 11: Diffuse and Glossy Self-transfer. Unshadowed transfer (a,c) includes no global transport effects. Interreflected transfer (b,d) includes both shadows and interreflections.

(a) no interreflections (shadowed transfer) (b) 1-bounce interreflections (c) 2-bounce interreflections

Figure 12: **Interreflections in Self-Transfer.** Top row shows diffuse transfer; bottom row shows glossy transfer. Note the reflections under the knob on the lid and from the spout onto the body. Run-time performance is insensitive to interreflections; only the preprocessed simulation must include additional bounces. Further bounces after the first or second typically provide only subtle change.