Light Factorization for Mixed-Frequency Shadows in Augmented Reality

Derek Nowrouzezahrai[1] Stefan Geiger[2] Kenny Mitchell[3] Robert Sumner[1] Wojciech Jarosz[1] Markus Gross[1,2]
[1] Disney Research Zurich [2] ETH Zurich [3] Black Rock Studios

ABSTRACT

Integrating animated virtual objects with their surroundings for high-quality augmented reality requires both geometric and radiometric consistency. We focus on the latter of these problems and present an approach that captures and factorizes external lighting in a manner that allows for realistic relighting of both animated and static virtual objects. Our factorization facilitates a combination of hard and soft shadows, with high-performance, in a manner that is consistent with the surrounding scene lighting.

Index Terms: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities—; I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture—

1 INTRODUCTION

Shadows provide important perceptual cues about the shape and relative depth of objects in a scene, as well as the surrounding lighting. Incorporating realistic shadows in a manner that is consistent with a scene's lighting is an important problem in augmented reality.

We address the problem of shadowing animated virtual characters, as well as static objects, in augmented reality with lighting captured from the real-world. Typically, real-world illumination causes both hard and soft shadows, the latter due to light reflecting off surrounding objects as well as emission from broad area lights, and the former due to smaller light sources such as the sun.

We factorize light in a well-founded manner, allowing hard and soft shadows to be consistently computed in real-time with a combination of shadow-mapping and basis-space relighting approaches.

After a brief summary of previous work (Section 2), we overview basic concepts and notation (Section 3), discuss geometry and light calibration (Sections 4 and 5), detail our light factorization (Section 6) and shading/shadowing models (Section 7), and discuss conclusions and future work (Section 9). Sections 5 and 7 focus on the mathematical derivations of our factorization and shading models; however, despite an involved exposition, our run-time implementation is quite straightforward and readers not interested in these details can skip ahead to Section 8 for implementation details.

2 PREVIOUS WORK

The seminal work by Sloan et al. [10] on *precomputed radiance transfer* proposed a technique for compactly storing precomputed reflectance and shadowing functions, for static scenes, in a manner that allows for high-performance relighting under novel illumination at run-time. In follow-up work, Sloan et al. use a different representation for similar reflectance/shadowing functions that allowed for approximate shading of deformable objects [11]. Ren et al. [6] extend this line of PRT work to fully dynamic scenes, allowing for soft shadows to be computed on animating geometry from dynamic environmental lighting in real-time. We combine ideas from several of these approaches and elaborate on technical specifics in more detail in Sections 3, 5 and 7.

Debevec and Malik [1] pioneered the area of *image-based lighting*, detailing an approach for capturing environmental lighting from the real-world and using this to shade virtual objects. More advanced lighting capture techniques exist, combining knowledge of the surrounding geometry with more detailing directional capture (e.g., from omni-directional cameras) [7], however we build on the simplicity and efficiency of Debevec and Malik's approach.

Figure 1: Hard and soft shadowing using our light factorization.

In augmented reality, work on shading virtual objects can be roughly divided into three groups: traditional computer graphics models, discretized lighting models, and basis-space relighting. The first set of work simply applies simple point/directional lighting models to compute shading (and possibly shadows) from virtual object onto the real scene. With such simple models, the shading of real and virtual objects is inconsistent, causing an unacceptable perceptual gap. Discretized lighting models use more advanced radiosity and instant-radiosity based approaches [5] to compute realistic shading on the virtual objects (at a higher performance cost); however, integrating these techniques in a manner that is consistent with the shading of the real-world objects is an open problem. Thus, the increased realism of the virtual object shading is still overshadowed by the discrepancy between virtual and real shading. Basis-space relighting approaches capture lighting from the real-world and use it to light a virtual object [3]. By construction, the shading on the virtual object will be consistent (to varying degrees) with the real-world shading. However, the coupling of shading between virtual and real objects is a difficult radiometric problem where even slight errors can cause objects to appear to "float" or stand-out from the real-world objects. We address a core component of this problem, computing consistent shading/shadowing on virtual objects and onto perceptually important regions of the real-world. Furthermore, we support animated objects, whereas prior basis-space approaches only handle static geometry.

3 OVERVIEW AND NOTATION

We adopt our mathematical notation from the real-time rendering literature: italics for scalars and 3D points/vectors (e.g., ω), boldface for column vectors and vector-valued functions (e.g., $\mathbf{y}$), and sans serif for matrices/tensors (e.g., M).

3.1 Spherical Harmonics

Many light transport signals, such as the incident radiance at a point, are naturally expressed as spherical functions. The spherical harmonic (SH) basis is the spherical analogue of the Fourier basis and can be used to compactly represent such functions. We summarize some key SH properties (that we will exploit during shading)

below, and leave rendering-specific properties to Section 7.

Definition. The SH basis functions are defined as follows:

$$y_l^m(\omega) = K_l^m \begin{cases} \sqrt{2}\, P_l^{-m}(\cos\theta)\, \sin(-m\phi), & m < 0 \\ \sqrt{2}\, P_l^m(\cos\theta)\, \sin(m\phi), & m > 0 \\ P_l^0(\cos\theta), & m = 0 \end{cases}, \quad (1)$$

where $\omega = (\theta, \phi) = (x, y, z)$ are directions on the sphere S^2, P_l^m are associated Legendre polynomials, K_l^m is a normalization constant, l is a band index, and $-l \le m \le l$ indexes basis functions in band-l. Basis functions in band-l are degree l polynomials in (x, y, z). SH is an orthonormal basis, satisfying $\int_{S^2} y_l^m(\omega)\, y_{l'}^{m'}(\omega) d\omega = \delta_{lm,l'm'}$, where the Kroenecker delta $\delta_{x,y}$ is 1 iff $x = y$.

Projections and Reconstruction. A spherical function $f(\omega)$ can be projected onto the SH basis, yielding a coefficient vector

$$\mathbf{f} = \int_{S^2} f(\omega)\, \mathbf{y}(\omega)\, d\omega\,, \quad \text{and:} \quad f(\omega) \approx \widetilde{f}(\omega) = \sum_{i=0}^{n^2} \mathbf{f}_i\, y_i(\omega)\,, \quad (2)$$

where $\mathbf{y}$ is a vector of SH basis functions, the order n of the SH expansion denotes a reconstruction up to band $l = n-1$ with n^2 basis coefficients, and we use a single indexing scheme for the basis coefficients/functions where $i = l(l+1)+m$.

Zonal Harmonics. The $m = 0$ subset of SH, called the zonal harmonics (ZH), exhibit circular symmetry about the z-axis and can be efficiently rotated to align along an arbitrary axis ω_a. Given an order-n *ZH coefficient vector*, $\mathbf{z}$, with only n elements (at the $m = 0$ projection coefficients), we can compute the *SH coefficients* corresponding to this function rotated from z to ω_a as in [11]

$$g_l^m = \sqrt{4\pi/(2l+1)}\, \mathbf{z}_l\, y_l^m(\omega_a)\,. \quad (3)$$

In Section 7, we exploit this fast rotation expression for efficient rendering with a variety of different surface BRDF models.

SH Products. Given functions f and g, with SH coefficient vectors $\mathbf{f}$ and $\mathbf{g}$, the SH projection of the product $h = f \times g$ is

$$\mathbf{h}_i = \int_{S^2} h(\omega)\, y_i(\omega)\, d\omega \approx \int_{S^2} \left[\sum_j \mathbf{f}_j y_j(\omega)\right]\left[\sum_k \mathbf{g}_k y_k(\omega)\right] y_i(\omega)\, d\omega$$

$$= \sum_{jk} \mathbf{f}_j\, \mathbf{g}_k \int_{S^2} y_j(\omega)\, y_k(\omega)\, y_i(\omega)\, d\omega = \sum_{jk} \mathbf{f}_j\, \mathbf{g}_k\, \Gamma_{ijk}\,, \quad (4)$$

where Γ is the SH triple-product tensor. Computing these general products is expensive, despite Γ's sparsity, but by fixing one of the functions in the product, a specialized *product matrix* can be used:

$$\left[\mathsf{M}^{\mathbf{f}}\right]_{ij} = \sum_k \mathbf{f}_k \Gamma_{ijk} \quad \text{such that} \quad \mathbf{h} = \mathsf{M}^{\mathbf{f}} \cdot \mathbf{g}. \quad (5)$$

In Section 7, we discuss these specialized product matrices in the context of shading.

3.2 Placement

An essential component of any reliable AR system is the computation of a consistent coordinate frame relative to the camera. Two common approaches are marker-based and markerless tracking.

Marker-based approaches compute a coordinate frame that remains consistent and stable under significant camera motion. Markerless based tracking instead relies on computer vision algorithms to establish this coordinate frame, and these techniques can suffer from instabilities, especially during camera (and scene) motion. The strength of markerless tracking lies in its generality: no markers are required whereas, for marker-based tracing, if a marker is (even partially) occluded, unexpected results may be computed.

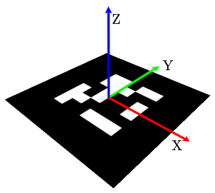
Our work focusses on consistent lighting and shading given a pre-calibrated AR coordinate frame, and so we instead build on previous tracking techniques as described in Section 4.

4 Geometric Calibration

We combine marker-based and markerless approaches to obtain robust correspondence, even under a freely moving camera.

We place a marker on the planar surface we wish to place our virtual objects on, detect the position and orientation of the marker using the ARToolKitPlus library [12], and find a mapping between this coordinate frame and the coordinate frame computed with the PTAM markerless tracking library [4].

To do so, PTAM computes a homography from corresponding feature points of the two captured images (from similar viewpoints). The dominant plane of the feature points is placed at the $z = 0$ plane, forming our initial coordinate frame. The ARToolKitPlus has a coordinate system centered on the marker (see inset) and, given the two images, can compute the positions (p_1, p_2) and rotations $(\mathsf{R}_1, \mathsf{R}_2)$ relative to the marker. PTAM can similarly compute positions $(\hat{p}_1, \hat{p}_2)$ and rotations $(\widehat{\mathsf{R}}_1, \widehat{\mathsf{R}}_2)$ with respect to its coordinate frame.



In a global coordinate frame $(\mathsf{R}_1, \mathsf{R}_2)$ and $(\widehat{\mathsf{R}}_1, \widehat{\mathsf{R}}_2)$ specify the same set of rotations, and so the rotation that maps from ARToolKitPlus' frame to PTAM's frame is $\mathsf{R} = \widehat{\mathsf{R}}_1^{-1} \cdot \mathsf{R}_1 = \widehat{\mathsf{R}}_2^{-1} \cdot \mathsf{R}_2$.

All that remains is to determine the relative scale and translation between the two frames. First we compute the intersection point o of two rays with origins $(\hat{p}_1, \hat{p}_2)$ and directions $(\hat{d}_1, \hat{d}_2) = (\mathsf{R} \cdot (-p_1), \mathsf{R} \cdot (-p_2))$. These rays are not only guaranteed to intersect, but will do so at the origin of the marker (relative to the PTAM frame). The parametric intersection distance and relative scale are

$$t = \frac{(\hat{d}_1)_x \left[(\hat{p}_2)_y - (\hat{p}_1)_y\right] - (\hat{d}_1)_y \left[(\hat{p}_2)_x - (\hat{p}_1)_x\right]}{(\hat{d}_1)_y\, (\hat{d}_2)_x - (\hat{d}_1)_x\, (\hat{d}_2)_y} \text{ and } s = \frac{t\, \|\hat{d}_2\|}{\|p_2\|}\,.$$

5 Real-World Lighting

In order to shade virtual characters, and static geometry, in a manner that is consistent with the real-world, it is important to capture and apply the lighting from the surrounding environment to these virtual elements. We combine the two most common virtual lighting models in mixed reality, traditional point/directional lights and environmental light probes, in a novel manner.

Point and directional lights are a convenient for lighting virtual objects, supporting many surface shading (BRDF) and shadowing models. However, these lights rarely match lighting distributions present in real-world scenes, even with many such lights as in e.g. instant radiosity based approaches [5]. Moreover, the hard shadows that result from using these approaches can appear unrealistic, especially when viewed next to the soft shadows of real objects.

On the other hand, environmental light probes can be used to shade virtual objects with lighting from the real scene, increasing the likelihood of consistent appearance between real and virtual objects (see Figure 2). One drawback is that, while shadow functions can be precomputed for static virtual objects, it is difficult to efficiently compute soft shadows (from the environmental light) *from* virtual objects *onto* the real world *and* animated virtual objects.

We first discuss two techniques for capturing real-world lighting (Section 5.1), followed by a factorization of the this lighting (Section 6) that allows us to both shade and shadow virtual objects in a manner that is consistent with the real scene (Section 7).

5.1 Capturing Environmental Lighting

In Section 7 we show how to compute the shade at a point x in the direction towards the eye ω_o, $L_{\text{out}}(x, \omega_o)$. This requires (among other things) the *incident* lighting distribution at x, $L_{\text{in}}(x, \omega)$. In our work, we assume that the spatial variation of lighting in the scene can be aggregated into the directional distribution, represented as

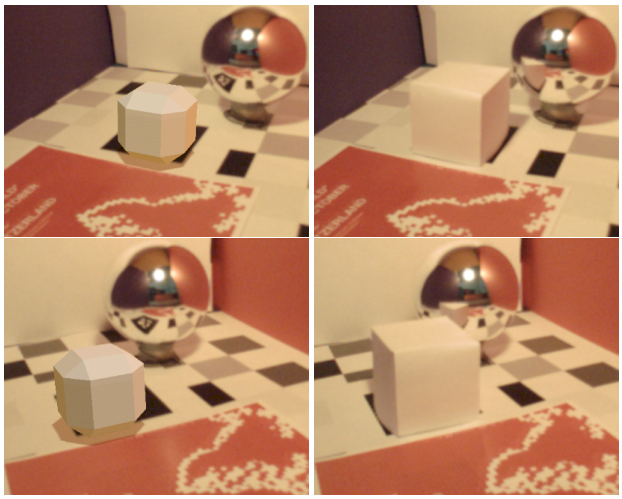


Figure 2: Consistent lighting between virtual and real objects.

an environment map of incident light, so that $L_{\text{in}}(x,\omega) = L_{\text{env}}(\omega)$. We now outline the two approaches we use to capture L_{env}.

Mirror Ball Capture. We place a mirror sphere at a known position relative to a marker and use the ARToolkitPlus marker-based feature tracking library [12] to detect the camera position relative to the sphere. We require a parameterization of the the sphere image we capture in order to project it into SH (see below). If we normalize the image coordinates of the (cropped) sphere image to $(u,v) = [-1,1] \times [-1,1]$, we can map spherical coordinates as $\omega_{uv} = (\theta,\phi) = (\arctan(v/u), \pi\sqrt{u^2+v^2})$. Furthermore, when discretizing Equation 2, we compute

$$\mathbf{f} \approx \sum_{u,v} f(\omega_{uv})\ \mathbf{y}(\omega_{uv})\ d\omega_{uv}\ , \tag{6}$$

where $d\omega_{uv} = (2\pi/w)^2\ \text{sinc}(\theta)$ uses the width w in pixels of the (square) cropped mirror sphere image [1].

Free Roaming Capture. In the case where using a marker-based system is not feasible, we can capture an approximation of the environment lighting using the PTAM library [4].

We capture images of the surrounding environment and, for each image, place a virtual omnidirectional camera at the mean distance of all feature points computed by PTAM for that image. The image is then projected to the 6 faces of a virtual cube (placed in a canonical orientation upon system initialization).

We similarly require a discrete projected solid angle measure when computing Equation 6 using this cube map parameterization. With the normalizaed image coordinates (u,v) on a cube's face and the width w in pixels for a side of the cube, we have (as in [8])

$$t = 1 + u^2 + v^2 \text{ and } d\omega_{uv} = \left[4\, t^{-(3/2)}\right] / w^2\ . \tag{7}$$

In Section 8 we discuss how we implement high-performance capture and discretized SH projection completely on the GPU.

6 Lighting Factorization

We propose a two-term factorization of the environmental lighting into a *directional* term $L_{\text{d}}(\omega)$ and a residual *global* lighting $L_{\text{g}}(\omega)$, and we will enforce that $L_{\text{env}}(\omega) = L_{\text{d}}(\omega) + L_{\text{g}}(\omega)$.

Given the SH projection coefficients for each color channel of L_{env}, $\{\mathbf{L}^{[r]}, \mathbf{L}^{[g]}, \mathbf{L}^{[b]}\}$, our factorization seeks to compute a *dominant* light direction/color, treat it as a separate incident lighting signal, and leave a residual lighting signal which corresponds to non-dominant lighting (e.g., from broad area light sources and smooth indirect light bouncing off of surrounding surfaces).

Dominant Light Direction. Starting with the simpler case of monochromatic incident light $L(\omega)$ (with SH coefficients $\mathbf{L}$), the linear ($l = 1$) SH coefficients are scaled linear monomials in y, z and x respectively, and thus encode the 1^{st}-moment vector direction (ignoring coordinate permutations and scale factors) of the spherical function they represent (in this case, the monochromatic light):

$$\vec{d} = \int_{S^2} \left[\omega_x, \omega_y, \omega_z\right]^{\text{T}} \cdot L(\omega)\ d\omega\ , \tag{8}$$

where $(\omega_x, \omega_y, \omega_z)$ are the Cartesian coordinates of ω. In this case, $\vec{d}$ is the principal light vector and, from Equations 1 and 8, we can solve for the normalized principal light direction in terms of $\mathbf{L}$ as

$$\overline{(-\mathbf{L}_3, -\mathbf{L}_1, \mathbf{L}_2)} = \left[\sum_{i=1}^{3} (L_i)^2\right]^{-\frac{1}{2}} (-\mathbf{L}_3, -\mathbf{L}_1, \mathbf{L}_2)\ , \tag{9}$$

where we use SH single indexing here for compactness. While we could use more complex approaches (e.g., [8]) for the trichromatic lighting case, we instead choose a simpler, more efficient technique: we convert trichromatic lighting coefficients into monochromatic coefficients using the standard color-to-grayscale conversion. Thus, the dominant light direction can be extracted from L_{env} as

$$\omega_d = \overline{\begin{bmatrix} -\mathbf{L}_3^{[r]} & -\mathbf{L}_3^{[g]} & -\mathbf{L}_3^{[b]} \\ -\mathbf{L}_1^{[r]} & -\mathbf{L}_1^{[g]} & -\mathbf{L}_1^{[b]} \\ \mathbf{L}_2^{[r]} & \mathbf{L}_2^{[g]} & \mathbf{L}_2^{[b]} \end{bmatrix} \cdot \begin{bmatrix} 0.3 \\ 0.59 \\ 0.11 \end{bmatrix}}\ . \tag{10}$$

Dominant Light Color. Given the dominant lighting direction ω_d, we now determine the dominant light color in this direction.

To determine this color, we place a planar reflector perpendicular to a unit intensity SH directional light at ω_d, and compute its albedo so that it reflects unit radiance.

Given a planar diffuse reflector with a normal ω_d, the SH coefficients of a directional light at ω_d that yields unit outgoing radiance on the plane are $\alpha y_l^m(\omega_d)$, where $\alpha = 16\pi/17$ for order-3 and order-4 SH, and $32\pi/31$ for order-5 and order-6 SH[1] [8]. Similar scaling factors can be derived for non-diffuse reflectance, but we have found that using this factor yields plausible results, regardless of the underlying surface BRDF used at run-time (see Section 7).

We can now analytically solve for the color as

$$c^{[\cdot]} = \left[\sum_{m=-1}^{1} \alpha\ y_1^m(\omega_d)\ \mathbf{L}_{2+m}^{[\cdot]}\right] / \left[\sum_{m=-1}^{1} (\alpha\ y_1^m(\omega_d))^2\right]\ . \tag{11}$$

Final Factorization. Given ω_d and RGB color $(c^{[r]}, c^{[g]}, c^{[b]})$, the *directional* term of our factorization and its SH projection are

$$L_{\text{d}}(\omega) = c^{[\cdot]}\ \delta(\omega_d) \quad \text{and} \quad \mathbf{L}_{\text{d}}^{[\cdot]} = \alpha\ c^{[\cdot]}\ y_l^m(\omega_d)\ . \tag{12}$$

The *global* term and its SH projection are

$$L_{\text{g}}(\omega) = L_{\text{env}}(\omega) - L_{\text{d}}(\omega) \quad \text{and} \quad \mathbf{L}_{\text{g}}^{[\cdot]} = \mathbf{L}^{[\cdot]} - \mathbf{L}_{\text{d}}^{[\cdot]}\ . \tag{13}$$

Note that the factorization is "perfect", in the sense that the environmental light can be perfectly reconstructed from the directional and global terms (both in the primal and SH spaces; see Figure 3). In Section 7, we use this factorization to combine several shading/shadowing models, resulting in realistic shading with hard and soft shadows that is consistent with shading on real-world objects.

[1]The α values are shared across two orders because the SH projection of the clamped cosine kernel vanishes for all even bands above $l = 2$.

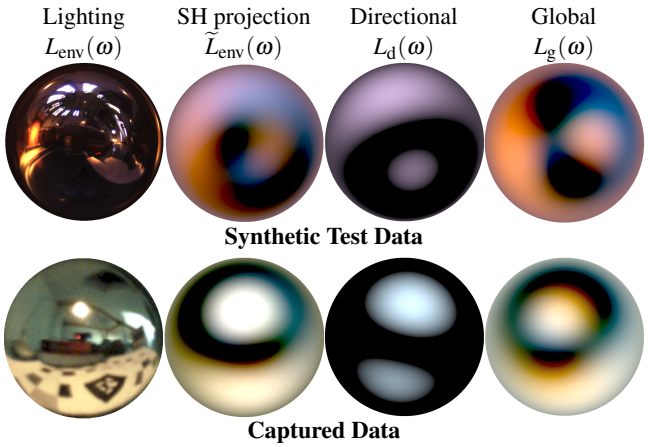


Figure 3: Factorization into directional and global lighting terms.

7 Shading with Factorized Real-World Lighting

Direct lighting at a point x towards the eye from the environment is

$$L_{\text{out}}(x,\omega_o) = \int_{S^2} L_{\text{env}}(\omega)\, V(x,\omega)\, f^+(x,\omega_o,\omega)\, d\omega \qquad (14)$$

where $V(x,\omega)$ is the binary visibility function that is 1 for directions that are not occluded by geometry, and 0 otherwise, and $f^+(x,\omega_o,\omega) = f(x,\omega_o,\omega)\ (n_x\cdot\omega)$ is a combination of the view-evaluated BRDF $f(x,\omega_o,\omega)$ and a cosine foreshortening term.

There are several challenges to accurately solving Equation 14. Firstly, we must integrate over all lighting directions. Secondly, during integration, we need to evaluate the binary visibility function which, in the most general case, requires tracing rays through the scene for each lighting direction (and at each x).

Previous work has either approximated L_{env} (or, more generally, L_{in}) with many point lights, or assumed static geometry where V can be precomputed and stored at a discrete set of x's. In the first case, a point light approximation allows for visibility to be computed using e.g. shadow maps, however many such point lights may be required and the cost of computing and shading with many shadow maps quickly becomes the bottleneck of these approaches. In the latter case, if static scene geometry is assumed, SH based shading approaches (which we will discuss below) can be used to quickly integrate over the many lighting directions (without explicitly discretizing them into e.g. individual point lights) and, along with precomputed visibility data, can compute soft shadows that respond to the dynamic lighting environment. Unfortunately, these approaches cannot handle animating or deforming geometry, since the visibility changes at run-time in these instances.

We instead exploit our factorization, using a combination of approaches to solve the two problems of light integration and dynamic visibility computation. Substituting Equation 13 into 14 yields

$$\begin{aligned} L_{\text{out}}(x,\omega_o) &= \int_{S^2} L_{\text{d}}(\omega)\, V(x,\omega)\, f^+(x,\omega_o,\omega)\, d\omega + \\ &\quad \int_{S^2} L_{\text{g}}(\omega)\, V(x,\omega)\, f^+(x,\omega_o,\omega)\, d\omega \\ &= L^{\text{d}}_{\text{out}}(x,\omega_o) + L^{\text{g}}_{\text{out}}(x,\omega_o) \end{aligned} \qquad (15)$$

and we will discuss solutions to each of these terms independently.

7.1 Efficient Computation of $L^{\text{d}}_{\text{out}}$

Substituting Equation 12 into the definition of $L^{\text{d}}_{\text{out}}$ yields

$$\begin{aligned} L^{\text{d}}_{\text{out}}(x,\omega_o) &= \int_{S^2} c^{[\cdot]}\, \delta(\omega_d)\, V(x,\omega)\, f^+(x,\omega_o,\omega)\, d\omega \\ &= c^{[\cdot]}\, V(x,\omega_d)\, f^+(x,\omega_o,\omega_d)\ . \end{aligned} \qquad (16)$$

Model	$f^+(x,\omega_o,\omega)$	ZH coefficients $\mathbf{z}_l$
Lambertian	$(n_x\cdot\omega)$	$[0.282, 0.326, 0.158, 0]$
Phong	$(\omega_r\cdot\omega)^s$	$\left[\frac{0.282(s+2)}{s+1}, 0.489, \frac{0.631s(s+2)}{s^2+4s+3}, \frac{0.746(s-1)}{(s+4)}\right]$
Mirror	$\delta(\omega=\omega_r)$	$[0.282, 0.489, 0.631, 0.746]$

ω_r is the reflection of ω_o about n_x and s is the Phong exponent.

Table 1: Analytic form of the BRDFs we use and their ZH coefficients.

In this form, integration over light directions is replaced by a single evaluation and visibility can be computed using shadow maps. The BRDF models we use when evaluating f^+ (as well as their ZH representations; see Section 7.2) are summarized in Table 1.

Although Equation 16 requires a simple application of shadowed point lighting, the parameters of this model are carefully derived in our factorization to maintain consistency with the global shading term, which we evaluate without explicitly sampling any lighting directions (Section 7.2). The combination of $L^{\text{d}}_{\text{out}}$ and $L^{\text{g}}_{\text{out}}$, and the manner in which L_{d} and L_{g} are derived and applied, which makes the use of point lighting acceptable for our solution.

7.2 Efficient Computation of $L^{\text{g}}_{\text{out}}$

Unlike $L^{\text{d}}_{\text{out}}$ in Equation 16, $L^{\text{g}}_{\text{out}}$ cannot reduce into a single sampling operation since L_{g} is composed of lighting from all directions. Sampling and evaluating the three terms in the integrand, for all directions at run-time, is not feasible. We exploit the smoothness of L_{g} and perform this computation efficiently in the SH domain.

As discussed earlier, the two main challenges when computing Equation 14 are integration over all light directions and evaluation of the visibility function (at all directions and all x). The expression below for $L^{\text{d}}_{\text{out}}$ also exhibits these problems,

$$L^{\text{g}}_{\text{out}}(x,\omega_o) = \int_{S^2} L_{\text{g}}(\omega) V(x,\omega) f^+(x,\omega_o,\omega) d\omega\ , \qquad (17)$$

and we will first discuss the *integration problem* (assuming we have a solution to the visibility problem), and then discuss several approaches we employ for solving the *visibility problem*[2].

Integration with SH. As we readily have the SH projection of L_{g}, $\mathbf{L}_{\text{g}}$ (from Section 5.1), suppose we can express V and f^+ with SH projections $\mathbf{V}$ and $\mathbf{f}$, then the most general solution to Equation 17 using SH involves summing over the triple-product tensor,

$$\begin{aligned} L^{\text{g}}_{\text{out}} &\approx \int_{S^2} \left[\sum_i \left[\mathbf{L}_{\text{g}}\right]_i y_i(\omega)\right] \left[\sum_j \mathbf{V}_j y_j(\omega)\right] \left[\sum_k \mathbf{f}_k y_k(\omega)\right] d\omega \\ &= \sum_{ijk} \left[\mathbf{L}_{\text{g}}\right]_i\ \mathbf{V}_j\ \mathbf{f}_k\ \Gamma_{ijk}\ , \end{aligned} \qquad (18)$$

which is a computationally expensive procedure, particularly when executed at every x. Alternatively, in the case where one of the three terms in the integrand is known beforehand, we can precompute the SH product matrix of this function to accelerate the computation. For example, we could precompute a product matrix for $\mathbf{L}_{\text{g}}$ as

$$\left[\mathsf{M}^{\mathbf{L}_{\text{g}}}\right]_{ij} = \sum_k \left[\mathbf{L}_{\text{g}}\right]_k\ \Gamma_{ijk} \text{ such that } L^{\text{g}}_{\text{out}} \approx \sum_i \left[\mathsf{M}^{\mathbf{L}_{\text{g}}}\cdot\mathbf{f}\right]_i \mathbf{V}_i\ . \qquad (19)$$

Equation 19 avoids the per-point evaluation of the triple-product tensor in Equation 18, offloading this computation to a one-time (per lighting update) evaluation of the triple-product when computing $\mathsf{M}^{\mathbf{L}_{\text{g}}}$; the run-time now involves a simpler matrix-vector product following by a dot-product. Note that we construct the

[2]We drop the $[\cdot]$ superscript for color channel indexing, and assume that all equations are applied to each color channel independently.

product matrix for the lighting, instead of the BRDF, since lighting will change at most once per-frame whereas the view-evaluated BRDF(s) changes at least once per frame (and potentially once per point if we do not assume a distant viewer model).

We can further simplify run-time evaluation if *two* of the three integrand terms are known apriori. For example, in the case of static geometry and diffuse reflection, the product $T(x,\omega) = V(x,\omega) \times f^+(x,\omega)$ can be projected into SH during precomputation, yielding the following run-time computation[3]

$$[\mathbf{T}_\mathbf{x}]_i = \int_{S^2} T(x,\omega) y_i(\omega) d\omega \text{ such that } L^{\mathrm{g}}_{\mathrm{out}} \approx \sum_i [\mathbf{T}_\mathbf{x}]_i \left[\mathbf{L}_\mathrm{g}\right]_i, \quad (20)$$

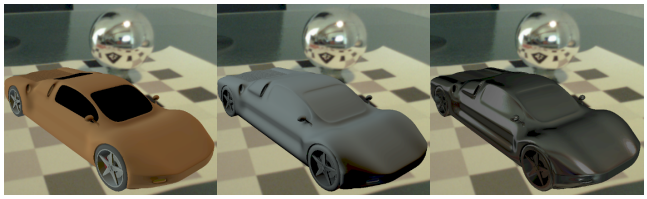
which corresponds to the standard PRT double product [10] and can be easily derived using the orthonormality property of SH.

One detail we have not discussed is how to compute the SH projection of the view-evaluated BRDF. We currently support the three common BRDF models in Table 1. Each of these BRDFs are circularly symmetric about an axis: the Lambertian clamped cosine is symmetric about the normal n_x, and the Phong and Mirror BRDFs are symmetric about the reflection vector ω_r. At run-time, the SH coefficients of the BRDFs is computed using Equation 3 and the order-4 ZH coefficients listed in Table 1.

We note that, in the case of perfect mirror reflection, instead of inducing a blur on the (very sharp) mirror reflection "lobe", we exploit the fact that we have captured $L_{\mathrm{env}}(\omega)$ and can readily sample $L_\mathrm{g}(\omega)$. In this case, we sample the SH-projected visibility when reconstructing the global lighting shade as

$$L^{\mathrm{g}}_{\mathrm{out}} \approx L_\mathrm{g}(\omega_r) \sum_i \mathbf{V}_i \, y_i(\omega_r) \, . \quad (21)$$

Figure 4 illustrates the different BRDF components for an object shaded with only the *global* lighting.



Diffuse Glossy Mirror

Figure 4: Shading, including soft shadows, from global/ambient light (directional light omitted) due to each BRDF component ($>$ 70 FPS).

While we have discussed shadowing for *directional* lighting, we have so far assumed that the SH projection of binary visibility at x, $\mathbf{V}$, was readily available for use in shadowing the *global* lighting. Below we discuss the different ways we compute $\mathbf{V}$.

SH Visibility. We discuss different shadowing models for global lighting, depending on the type of virtual object we are shading.

For static objects, we use either standard PRT (Equation 20) for diffuse objects or precomputed SH visibility product matrices and Equation 19 (replacing $\mathsf{M}^{\mathbf{L}_\mathrm{g}}$ with a product matrix for visibility).

For animating or deformable objects, we segment shadowing into two components: *cast shadows* from the object onto the environment, and *self shadows* from the object onto itself.

For cast shadows, we are motivated by previous work on *spherical blocker approximations* [6]. We fit a small number (10 to 20) of spheres to our animating geometry (e.g., in its rest pose for articulated characters) and skin their positions during animation. Shadowing is due to the spherical proxy geometry, as opposed to the actual underlying geometry. The use of spherical blocker approximation is justified in two ways. Firstly, since the global lighting component is a smooth spherical function, equivalent to large and broad area light sources, fine scale geometry details will be lost in the smooth shadows that are produced by these types of area lights. Secondly, SH visibility can be efficiently computed using sphere blockers, as we will now discuss.

Considering only a single sphere blocker, the visibility function due to that blocker is a circularly symmetric function, and so if we align the vector from x to the center of the blocker with the z axis, the single sphere visibility is defined as in [6],

$$V_\mathrm{s}(\omega) = \begin{cases} 0, & \text{if } \arccos(\omega \cdot z) \le \arcsin\left(\frac{r}{d}\right) \\ 1, & \text{otherwise} \, , \end{cases} \quad (22)$$

and has ZH coefficients

$$\mathbf{V}_\mathrm{S} = \int_{S^2} V_\mathrm{S}(\omega) \, \mathbf{y}(\omega) d\omega$$
$$= \left[1.772\left(1+\sqrt{1-R}\right), -1.535R, -1.982R\sqrt{1-R}, \frac{0.586R\left(-4d^2+5r^2\right)}{d^2} \right], \quad (23)$$

where $R = (r/d)^2$. If we approximate our animated object with a single sphere, applying Equation 3 to Equation 23 will yield $\mathbf{V}$, which can be used in Equation 19 to compute the final shade. However, it is rarely the case that a single blocker sphere is sufficient. In the case of multiple spheres, SH products of the visibility from individual spheres must be taken to compute the total visibility $\mathbf{V}$. Instead of computing SH visibility for a sphere at a time and applying Equation 4 (which requires expensive summation of Γ, for each sphere), we compute an analytic product matrix for a single sphere blocker oriented about an arbitrary axis by combining Equations 23, 3 and 5.

In the case of only a few blocker spheres (e.g., if we only model cast shadows from the feet of a character onto the ground, using two spheres), applying the precomputed sphere blocking product matrix to $\mathbf{L}_\mathrm{g}$ (or $\mathbf{f}$) and shading with Equation 19 is an efficient solution. However, as we increase the number of spherical blockers (and thus, the number of product matrix multiplications), performance degrades rapidly. In this case, we accelerate this SH multi-product by performing computation in the *logarithmic SH* domain [6].

The technique of Ren et al. [6] computes the SH projection of $\log(V_s(\omega))$ (which, in the canonical orientation, is still a ZH) and tabulates these coefficients as a function of (r/d). Rotated log SH coefficients are computed for each sphere blocker, using Equation 3, and summed together yielding a net log SH visibility vector, $\mathbf{V}_{\log}$. We use the tabulated values provided by Ren et al. and directly apply the "optimal linear" SH *exponentiation* operator to convert the net log SH visibility to the final SH visibility vector as

$$\mathbf{V} \approx \exp\left(\frac{[\mathbf{V}_{\log}]_0}{\sqrt{4\pi}}\right) \left(a \, \|\widehat{\mathbf{V}}_{\log}\| \, \mathbf{1} + b \, \|\widehat{\mathbf{V}}_{\log}\| \, \widehat{\mathbf{V}}_{\log}\right) , \quad (24)$$

where $\widehat{\mathbf{V}}_{\log}$ is $\mathbf{V}_{\log}$ with its DC term set to 0, and $\mathbf{1} = (\sqrt{4\pi}, 0, \ldots, 0)$ is the SH projection of the constant function $one(\omega) = 1$.

For *self-shadows*, Ren et al. [6] propose "sphere replacement rules" for handling shade points that lie on the surface of the mesh (potentially within the volume of several blocker spheres), however we use a simpler solution that yields suitable results.

We precompute and store the DC projection of visibility, which is related to ambient occlusion, at vertices of the dynamic object. At run-time, we need only scale the unshadowed lighting by this occlusion factor. This is equivalent to using Equation 19 with only $\mathbf{V}_0 \neq 0$; in the case of triple product integration, if one of the terms has only a non-zero DC component, this is equivalent to scaling the double product integration by the DC component of the third (DC-only) term in integrand [8]. Unshadowed light is computed by rotating the appropriate ZH vector from Table 1 using Equation 3, and then computing the double product integral (as in Equation 20)

[3] Note that $f^+(x,\omega_o,\omega) = f^+(x,\omega)$ in the case of diffuse reflectance.

of $\mathbf{L}_g$ with these rotated BRDF coefficients. Figure 5 illustrates the contribution of the different shadowing components we use.

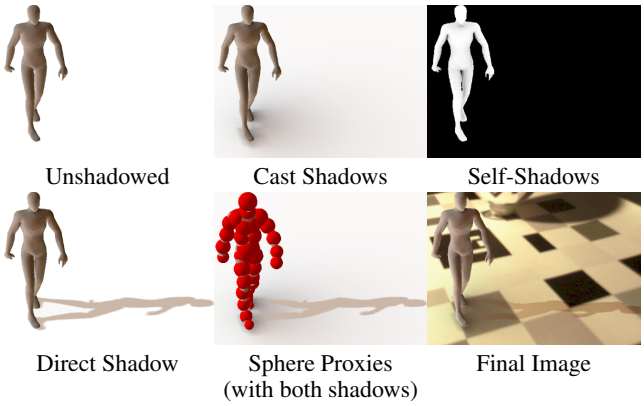


Figure 5: Top row: smooth cast shadows (middle) and self-shadows (right) due to global lighting give subtle depth and lighting cues compared to no global shadows (left). Bottom row: compositing the final shading with directional shadows (left), visualizing the blocker spheres with directional and cast shadows (middle), and the final composited image (right). Images rendered at > 70 FPS.

8 Implementation and Performance

We benchmark our system on an Intel Core2 Duo 2.8 GHz Laptop with an NVidia Quadro FX 770M GPU. We use the PlayStation Eye camera for capture, supporting 640×480 capture.

Our end-to-end algorithm performs following computations:

- At initialization, compute geometric calibration (Section 4),
- Capture lighting from mirror sphere or free roaming (Section 5),
- Compute SH projection of L_{env} (see details below; Equation 6),
- Compute $\mathbf{L}_d$ and $\mathbf{L}_g$ using lighting factorization (Section 6),
- Compute L^{d}_{out} (see details below; Section 7.1 and Equation 16),
- Compute L^{g}_{out} (Section 7.2), and
- Composite *direct* and *global* shading components.

We compute the SH projection of the captured lighting by precomputing $\mathbf{y}(\omega_{uv})\, d\omega_{uv}$ values in a texture and use graphics hardware to quickly multiply this precomputed texture with the captured lighting image, $L_{\text{env}}(\omega_{uv})$. Depending on the spherical parameterization (sphere vs. cube), we use the appropriate definition of $d\omega_{uv}$ as well as the appropriate texture image layout (i.e., six textures are required for the cube map case). The summation in Equation 6 is computed with a multi-pass sum on the GPU. Table 2 summarizes the performance breakdown of this component of our algorithm.

Mirror Sphere Capture and Projection			
Find Markers	LDR to HDR	SH Project	Total
4 ms	0.6 ms	1.7 ms	6.3 ms

Free Roaming Capture and Projection				
Find Homography	LDR to HDR	Map to Cube	SH Project	Total
2 ms	0.8 ms	1.1 ms	1.8 ms	5.7 ms

Table 2: Performance breakdown for capturing and projecting real-world illumination into SH using our two capture methods.

When computing L^{d}_{out} with Equation 16, we use a 1024×1024 24-bit shadow map, and 4×4 percentage-closer filtering [2]. All shading computations are performed on the GPU using GLSL shaders with lighting coefficients computed every frame, which allows for dynamic lighting response.

All of our results run at higher than 70 FPS, including all geometric calibration, lighting and shading computations.

9 Conclusions and Future Work

We presented a technique to factor environmental light emitting from, and bouncing off, the real-world. Our factorization is designed to directly support both hard and soft shadowing techniques. Using basis-space relighting techniques and GPU acceleration, we can efficiently compute shadows from both static and animated virtual objects. The manner in which we factor and combine different illumination contributions is novel, and generates more consistent shading than previously possible (e.g., with only the individual application of any one of the techniques we incorporate).

Traditional cast shadows in AR are sharp and colored in a manner that does not respond to the surrounding lighting, whereas our hard shadows are shaded based on dynamic environmental ambient light. Moreover, soft shadows due to residual global lighting add physically-based smooth shading, increasing perceptual consistency in a manner similar to diffuse interreflection.

Our factorization roughly identifies a direct lighting and "intelligent" bounced lighting terms, however we still apply direct-illumination integration to these components, ignoring the effects of indirect light bounces *from* the virtual objects *onto* the real-world geometry. In the future we plan on incorporating such effects by, for example, modeling light bouncing off of the sphere proxies [9].

Acknowledgements

We thank Paul Debevec for the light probes, and Zhong Ren for the humanoid mesh and tabulated log SH sphere blocker data.

References

[1] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 1997 Papers*, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[2] R. Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, NY, USA, 2005. ACM.

[3] S. Heymann, A. Smolic, K. Müller, and B. Froehlich. Illumination reconstruction from real-time video for interactive augmented reality. International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '05), 2005.

[4] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, Nov 2007.

[5] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. ISMAR '10: Proceedings of the 2010 9th IEEE International Symposium on Mixed and Augmented Reality, Oct. 2010.

[6] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM SIGGRAPH 2006 Papers*, pages 977–986, NY, USA, 2006. ACM.

[7] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. IEEE Transactions on Visualization and Computer Graphics, 1999.

[8] P.-P. Sloan. Stupid spherical harmonics (SH) tricks. Game Developers Conference, 2009.

[9] P.-P. Sloan, N. K. Govindaraju, D. Nowrouzezahrai, and J. Snyder. Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics Conference*, Washington, DC, USA, 2007. IEEE Computer Society.

[10] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH 2002*, NY, USA, 2002. ACM.

[11] P.-P. Sloan, B. Luna, and J. Snyder. Local, deformable precomputed radiance transfer. In *SIGGRAPH 2005*, NY, USA, 2005. ACM.

[12] D. Wagner and D. Schmalstieg. ARToolKitPlus for pose tracking on mobile devices artoolkit. Proceedings of 12th Computer Vision Winter Workshop (CVWW '07), 2007.