

A Practical Algorithm for Rendering Interreflections with All-Frequency BRDFs

KUN XU, YAN-PEI CAO, and LI-QIAN MA

Tsinghua University

ZHAO DONG

Cornell University

RUI WANG

University of Massachusetts

and

SHI-MIN HU

Tsinghua University

Algorithms for rendering interreflection (or indirect illumination) effects often make assumptions about the frequency range of the materials' reflectance properties. For example, methods based on Virtual Point Lights (VPLs) perform well for diffuse and semi-glossy materials but not so for highly glossy or specular materials; the situation is reversed for methods based on ray tracing. In this article, we present a practical algorithm for rendering interreflection effects with all-frequency BRDFs. Our method builds upon a spherical Gaussian representation of the BRDF, based on which a novel mathematical development of the interreflection equation is made. This allows us to efficiently compute one-bounce interreflection from a triangle to a shading point, by using an analytic formula combined with a piecewise linear approximation. We show through evaluation that this method is accurate for a wide range of BRDFs. We further introduce a hierarchical integration method to handle complex scenes (i.e., many triangles) with bounded errors. Finally, we have implemented the present algorithm on the GPU, achieving

This work was supported by the National Basic Research Project of China (2011CB302205), the Natural Science Foundation of China (61120106007 and 61170153), the National High Technology Research and Development Program of China (2012AA011503), Tsinghua University Initiative Scientific Research Program and the National Science Foundation (CCF-0746577). K. Xu is also supported by the CCF-Intel Young Faculty Researcher Program.

Authors' addresses: K. Xu (corresponding author), Y.-P. Cao, and L.-Q. Ma, TNLlist, Department of Computer Science and Technology, Tsinghua University, Beijing; email: xukun.1985@gmail.com; Z. Dong, Program of Computer Graphics, Cornell University, Ithaca, NY; R. Wang, School of Computer Science, University of Massachusetts, Amherst, MA; S.-M. Hu, TNLlist, Department of Computer Science and Technology, Tsinghua University, Beijing.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 0730-0301/2014/01-ART10 \$15.00

DOI: <http://dx.doi.org/10.1145/2533687>

rendering performance ranging from near interactive to a few seconds per frame for various scenes with different complexity.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Color, shading, shadowing, and texture*; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Ray tracing*

General Terms: Algorithms

Additional Key Words and Phrases: Rendering, interreflections, global illumination, spherical gaussian, all-frequency BRDFs

ACM Reference Format:

Kun Xu, Yan-Pei Cao, Li-Qian Ma, Zhao Dong, Rui Wang, and Shi-Min Hu. 2014. A practical algorithm for rendering interreflections with all-frequency BRDFs. *ACM Trans. Graph.* 33, 1. Article 10 (January 2014) 16 pages. DOI: <http://dx.doi.org/10.1145/2533687>

1. INTRODUCTION

Accurate rendering of interreflection (or indirect illumination) effects has been a long-standing challenge in computer graphics research, particularly when the materials vary across different BRDFs, from diffuse to semi-glossy and to highly glossy. This wide range of frequency scales poses a great challenge for rendering algorithms. Many existing algorithms are efficient for only a certain range of materials. For example, methods based on Virtual Point Lights (VPLs) [Keller 1997] perform well for diffuse and semi-glossy materials, but become increasingly inefficient for highly glossy or nearly specular materials. This is mainly because these methods represent the source of indirect illumination using a discrete point set. Such a representation works well with diffuse materials, due to its nature of low-frequency and smooth filtering. However, when dealing with highly glossy materials, the number of discrete points required in VPL-based methods increases significantly, hence reducing the computation performance and increasing the storage requirement.

On the other hand, methods based on path tracing [Kajiya 1986] are efficient for rendering highly glossy and specular materials, but perform poorly when the materials become diffuse or semi-glossy. This is mainly because these methods stochastically trace view rays upon reflections or refractions from the materials. Therefore highly glossy materials lead to lower variance in the

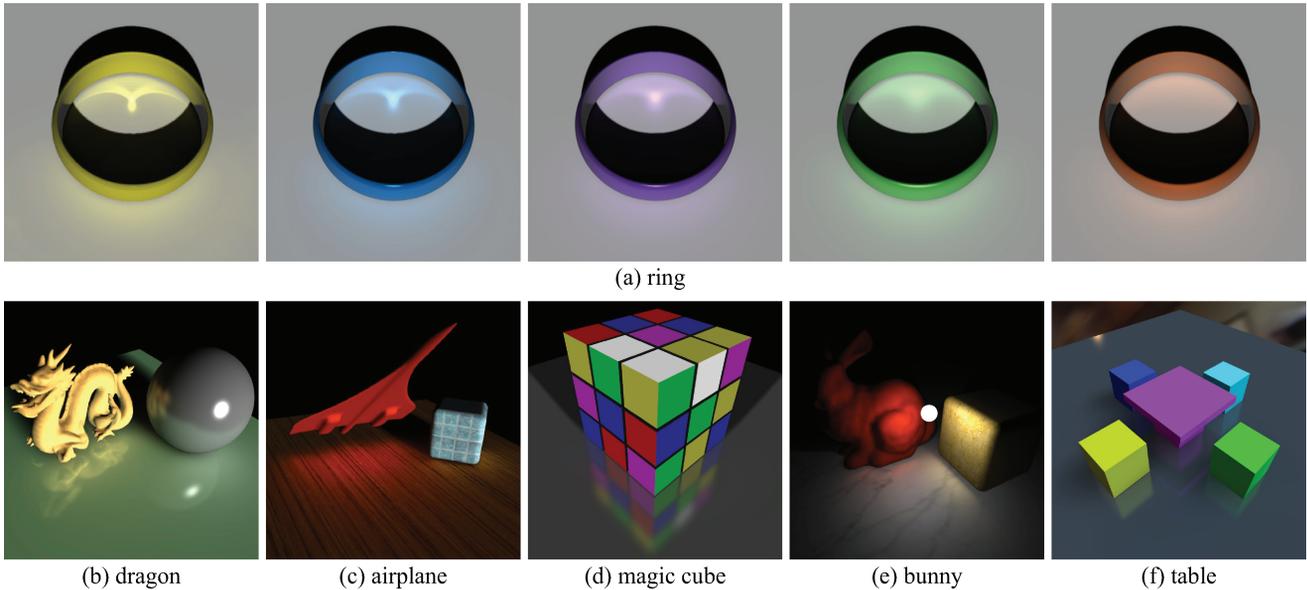


Fig. 1. Our algorithm achieves high-quality rendering of one-bounce interreflections with all-frequency BRDFs. The top row shows caustics on the plane where the BRDF of the ring varies from highly specular to diffuse. The bottom row shows various types of interreflection effects, such as indirect highlights (b), diffuse interreflections (c), glossy interreflections (d), and interreflection effects under different types of lights, such as local lights (e) and environment lights (f). Our algorithm runs at 0.4~4 fps for all the above scenes. The models in (b)(e) are courtesy of the Stanford 3D Scanning Repository.

computation and lower rendering noise; conversely, nearly diffuse materials lead to high variance and consequently increase rendering noise.

These limitations are essentially due to the lack of an algorithm that can efficiently handle a wide range of different materials, including both the diffuse and the specular ends, which is an open problem in rendering research. As a result, rendering a scene consisting of mixed materials (i.e., at different frequency scales) often requires a combination of several algorithms, each dealing with a separate frequency range. Such a rendering scheme not only increases algorithm complexity, but also is difficult to ensure all individual algorithms produce consistent results.

In this paper, we present a practical algorithm for rendering interreflection effects with all-frequency BRDFs. Our method builds upon a Spherical Gaussian (SG) representation of BRDFs [Wang et al. 2009a]. By changing the width of Spherical Gaussian, this representation can faithfully reproduce BRDFs over a wide range of frequency levels (i.e., glossiness). Our main contribution is a novel mathematical development of the interreflection equation. Specifically, by representing both BRDFs and lighting using SGs, we derive a formula for computing one-bounce interreflection from a triangle to a shading point. Using this formula together with a piecewise linear approximation, the interreflection results can be accurately computed. We show through evaluation that this method performs well for a wide range of BRDFs.

To account for complex scenes with many triangles in practical applications, we present a hierarchical integration method with bounced errors to improve the efficiency of our algorithm. Finally, we have implemented the whole algorithm on the GPU, achieving rendering performance ranging from near interactive to a few seconds per frame for various scenes with different complexity, as shown in Figure 1.

2. RELATED WORKS

Rendering interreflection (or indirect illumination) effects is a classic problem in computer graphics. A complete review is beyond the scope of this article. We refer the readers to Ritschel et al. [2012] for a comprehensive survey, and this section only covers the most relevant works. For clarity, we refer to light bouncing surfaces as *reflectors* and final shading surfaces as *receivers*.

Virtual Point Lights (VPLs). An efficient solution for computing interreflections is by representing the indirect lighting as a set of Virtual Point Lights (VPLs). As a classic VPL-based technique, instant radiosity [Keller 1997] creates VPLs by tracing paths from the primary lights, and uses the shadow map algorithm to estimate the total incident illumination from all VPLs to a shading point. To render high-quality complex lighting effects, VPL-based methods usually demand millions of VPLs, which significantly increases computation cost. This is also known as the many-light problem. To address this issue, Walter et al. [2005, 2006] proposed *Lightcuts*, which constructs a hierarchical structure of VPLs to reduce the computation complexity to be sublinear. Row-column sampling [Hašan et al. 2007] and *LightSlice* [Ou and Pellacini 2011] further reduce the computation cost by exploiting the low-rank structure of the light transport matrix. Traditional VPL-based methods are limited to diffuse or semi-glossy reflectors. To address this limitation, Hašan et al. [2009] presented virtual spherical lights to support glossy reflectors. Davidovic et al. [2010] separate light transport into low-rank (global) and high-rank (local) components, and employ different methods for different components to achieve detailed glossy interreflections. Recently, Walter et al. [2012] proposed *bidirectional Lightcuts* to reduce the bias in VPL-based rendering, which is achieved by introducing *virtual sensor points* on eye paths. While accurate, these VPL-based methods perform at offline speed, taking

minutes or hours to run. In addition, highly glossy receivers typically pose a big challenge as they require a very large number of VPLs.

Photon Mapping and Radiance Caching. Photon mapping [Jensen 2001] first traces particles from the primary lights to construct photon maps; then in the second pass, it performs ray tracing and estimates indirect illumination on nonspecular receivers using photon density estimation. By exploiting the GPU, photon mapping can achieve interactive performance [Purcell et al. 2003; Wang et al. 2009b; Fabianowski and Dingliana 2009; McGuire and Luebke 2009; Hachisuka and Jensen 2010]. However, when dealing with highly glossy receivers, the computation cost still increases enormously as many more samples are required to perform accurate photon density estimation. Radiance caching [Krivaneck et al. 2005; Gassenbauer et al. 2009] is an effective technique for accelerating glossy interreflections in Monte Carlo ray tracing, by sparsely caching and interpolating radiance on glossy surfaces.

General Light Transport. A number of recent efforts attempt to handle more general and complex light paths that are difficult for any unbiased methods. Jakob and Marschner [2012] discovered that sets of light paths contributing to the image naturally form low-dimensional manifolds in the path space. They exploited this idea to develop a Markov chain Monte Carlo algorithm that can well handle difficult specular reflection paths. Hachisuka et al. [2012] presented a path space extension to combine Monte Carlo path integration and photon density estimation in a unified framework, which can robustly render scenes with both glossy reflection and caustics. Similarly, Georgiev et al. [2012] integrated photon mapping and bidirectional path tracing into a robust combined algorithm via multiple importance sampling, which can handle specular-diffuse-specular lighting effects. Yet, all these methods run at offline speed.

Precomputed Radiance Transfer (PRT). Precomputed Radiance Transfer (PRT) [Sloan et al. 2002] achieves real-time indirect lighting of static scenes by precomputing light transport matrices and compressing them using certain basis functions to exploit the low-dimensional structure of the matrices. Various basis functions have been employed in PRT, including spherical harmonics [Sloan et al. 2002], wavelet [Ng et al. 2003], spherical Gaussian [Tsai and Shih 2006], piecewise basis [Xu et al. 2008], and discrete spherical function [Zhang et al. 2013]. PRT has been extended to render interreflections with dynamic BRDFs [Sun et al. 2007; Ben-Artzi et al. 2008; Cheslack-Postava et al. 2008; Ren et al. 2013] of static scenes. Interreflections in dynamic scenes have also been studied [Iwasaki et al. 2007; Pan et al. 2007], but is limited to low-frequency effects.

Interactive Global Illumination (GI). Dachsbacher and Stamminger [2005] introduced *Reflective Shadow Maps* (RSM) where pixels in the shadow map are considered as indirect light sources. This method gathers low-resolution indirect lighting from the RSM and obtains a high-resolution result using screen-space interpolation. While running in interactive framerates, it is limited to diffuse reflectors and ignores indirect shadows. Afterwards, Dachsbacher and Stamminger [2006] presented a method to include non diffuse reflectors by splatting the radiance contribution from each pixel, which only supports diffuse receivers. Ritschel et al. [2008] presented *imperfect shadow maps* to approximate indirect visibility from VPLs. However, it is limited to low-frequency effects. Later, Ritschel et al. [2009] introduced the micro-rendering technique for high-quality interactive GI. Final gathering at each shading point is efficiently computed by rasterizing the point-based hierarchy into a micro-buffer. The micro-buffer can be warped to account for BRDF

importance sampling. As a result it supports receivers with glossy BRDFs. Nevertheless, the reflector is restricted to contain either diffuse or low-frequency BRDFs. Laurijssen et al. [2010] proposed a method to interactively render indirect highlights, accounting for glossy-to-glossy paths. However, it is not suitable for diffuse receivers. Recently, Loos et al. [2011] presented modular radiance transfer for real-time indirect illumination, but this method only generates low-frequency effects. Crassin et al. [2011] achieved real-time one-bounce indirect lighting using voxel cone tracing, which organizes reflectors into a voxel octree, and approximates both the BRDF and incoming light radiance using Gaussian lobes. Final gathering at each pixel is performed by tracing a few cones towards the octree. This method is less precise compared to our method since it relies on a voxel-based representation instead of the original geometry, and is also inefficient for rendering caustics.

Finally, there are many image-based methods for efficiently rendering caustics [Wyman and Davis 2006; Shah et al. 2007], but these techniques are designed for perfectly specular reflectors.

Reflections/Refractions. Using Fermat's theorem, researchers have employed differential geometry [Mitchell and Hanrahan 1992] and Taylor expansion [Chen and Arvo 2000] to find reflection points on implicit curved reflectors. In addition, many methods [Ofek and Rappoport 1998; Roger and Holzschuch 2006] have been presented to generate specular reflections from triangle meshes. Walter et al. [2009] proposed an efficient method for finding refracted connecting paths from triangle meshes. However, these methods assume perfect reflection/refraction, and it is unclear how to extend them to handle nonspecular/glossy materials.

Spherical Gaussians (SGs). Spherical Gaussians (SGs) provide flexible widths and closed-form solutions for computing function products and integrals. Thus they have been widely adopted for representing spherical functions, such as environment lighting [Tsai and Shih 2006] and BRDFs [Wang et al. 2009a; Iwasaki et al. 2012a], and used for various applications including normal map filtering [Han et al. 2007], real-time rendering of rough refractions [de Rousiers et al. 2012] and translucent materials rendering [Yan et al. 2012]. Specifically, Wang et al. [2009a] approximate the normal distribution of a micro-facet BRDF using sums of SGs. They demonstrated that such an approximation is accurate for representing a wide range of parametric and measured BRDFs. By utilizing this property, real-time all-frequency rendering of static scenes with dynamic BRDFs is achieved. However, this method only considers direct illumination and ignores interreflections. By using a Summed-Area Table (SAT), Iwasaki et al. [2012b] introduced *Integral Spherical Gaussian (ISG)*, which can efficiently evaluate the integral of an SG over an axis-aligned spherical rectangle. Hence, using ISG, the product integral of the visibility function and an SG can be evaluated in real time, although for direct illumination only. Based on a piecewise linear approximation of SG, Wang et al. [2013] proposed an analytic method to evaluate the product integral of two SGs over a visible region. Recently, Xu et al. [2013] proposed *anisotropic spherical gaussians*, which extend SGs to efficiently represent anisotropic spherical functions while still retaining the desirable properties of SGs.

3. BACKGROUND AND ALGORITHM OVERVIEW

3.1 Background

We first review the necessary background of the Spherical Gaussians (SG) representation and BRDF approximation.

SG Definition. A Spherical Gaussian (SG) is a function of unit vector \mathbf{v} and is defined as

$$G(\mathbf{v}; \mathbf{p}, \lambda, c) = c \cdot e^{\lambda(\mathbf{v} \cdot \mathbf{p} - 1)}, \quad (1)$$

where unit vector \mathbf{p} , λ , and c represent the center direction, sharpness, and the scalar coefficient of the SG, respectively. We refer to the *width* of an SG as the inverse of the sharpness. For simplicity, we denote $G(\mathbf{v}) = G(\mathbf{v}; \mathbf{p}, \lambda, c)$ and $G(\mathbf{v}; \mathbf{p}, \lambda) = G(\mathbf{v}; \mathbf{p}, \lambda, 1)$. SGs have many known properties. For example, the integral of SG has analytic solutions, and the product of two SGs is still an SG. These properties are explained in detail in the Appendix.

BRDF Approximation. A BRDF is commonly represented as the sum of a diffuse component and a specular component:

$$\rho(\mathbf{i}, \mathbf{o}) = k_d + k_s \rho_s(\mathbf{i}, \mathbf{o}), \quad (2)$$

where \mathbf{i} , \mathbf{o} are the incoming and outgoing directions; k_d , k_s are the diffuse and specular coefficients. As demonstrated in Wang et al. [2009a], the specular component can be well approximated by a sum of SGs:

$$k_s \rho_s(\mathbf{i}, \mathbf{o}) \approx \sum_{j=1}^n G(\mathbf{i}; \mathbf{o}^j, \lambda^j, c^j),$$

where \mathbf{o}^j (which is calculated as $2(\mathbf{o} \cdot \mathbf{n}^j)\mathbf{n}^j - \mathbf{o}$), λ^j , and c^j are the center, sharpness, and coefficient of the j -th SG, respectively, and \mathbf{n}^j is the center of the j -th SG in the Normal Distribution Function (NDF) approximation [Wang et al. 2009a]. Note that the diffuse component k_d can be treated as a special SG with zero sharpness: $k_d = G(\mathbf{i}; 2(\mathbf{o} \cdot \mathbf{n})\mathbf{n} - \mathbf{o}, 0, k_d)$. Therefore the BRDF defined in Eq. (2) can be rewritten as the sum of $(n + 1)$ SGs:

$$\rho(\mathbf{i}, \mathbf{o}) \approx \sum_{j=0}^n G(\mathbf{i}; \mathbf{o}^j, \lambda^j, c^j), \quad (3)$$

where $\lambda^0 = 0$, $c^0 = k_d$, and $\mathbf{o}^0 = 2(\mathbf{o} \cdot \mathbf{n})\mathbf{n} - \mathbf{o}$. As shown in Wang et al. [2009a], the specular component of commonly used parametric BRDFs (e.g., the Blinn-Phong and the Cook-Torrance models) can be accurately approximated by one single SG. For anisotropic or measured BRDFs, a small number of SGs (typically 3–7) suffice to achieve accurate approximations.

3.2 Algorithm Overview

Our method aims at accurately and efficiently computing interreflections with all-frequency BRDFs represented by SGs.

In Section 4, we describe how to efficiently compute one-bounce interreflections from a single triangle (reflector) to a single shading point (receiver). We assume the lighting is distant, and can be represented using an SG (referred to as an *SG light*). We also represent the BRDFs of both reflector and receiver using SGs as described in Eq. (3). We then derive a novel analytic formula, combined with a piecewise linear approximation, to accurately compute the rendering integral of interreflections. This SG-based formula forms the foundation of our method.

Using the newly derived formula alone to compute interreflection, however, would only work for simple scenes with a small number of triangles. For complex scenes containing more than thousands of triangles, brute-force iteration over all triangles to evaluate the formula would be too costly. To address this issue, we propose a hierarchical integration method to handle complex scenes. Specifically, we organize all triangles into a binary tree, where each tree node represents a subset of triangles. We further derive formulas to efficiently compute the one-bounce interreflection reflected by a

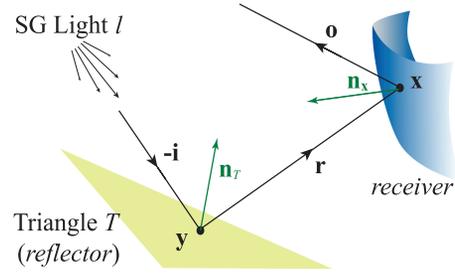


Fig. 2. Light path of one-bounce interreflection.

node (i.e., a subset of triangles), and to estimate the error bound of this approximation. Guided by the error bound, for each shading point, we find a *reflector cut* of the binary tree to approximate the sum of interreflections from all triangles. The reflector cut is obtained through iterative refinement until the largest error on the cut is small enough or the number of nodes exceeds a predefined threshold. The hierarchical integration scheme is explained in detail in Section 5.

Note that all preceding computations assume no occlusion between reflectors and receivers. To account for indirect visibility, we use a variant of imperfect shadow maps [Ritschel et al. 2008] to compute the average visibility from a node to a shading point. This is explained in Section 6.1. More implementation details will be discussed in Section 6.2.

4. ONE-BOUNCE INTERREFLECTION MODEL

We start by deriving the one-bounce interreflection model for a single triangle reflector under distant lighting, as shown in Figure 2. Assuming the incident distant lighting l can be represented by an SG in the form of $G(\mathbf{i}; \mathbf{i}_l, \lambda_l)$ ($G_l(\mathbf{i})$ for short), given a triangle T with normal \mathbf{n}_T (referred to as the *reflector*) and a shading point x with normal \mathbf{n}_x (referred to as the *receiver* point), we aim to compute the outgoing radiance from x to the view direction \mathbf{o} due to the reflection of the SG light l from triangle T towards x . Note that all directions are defined in the global frame.

To simplify the derivation, for now we assume there is no occlusion between the light, the reflector, and the receiver (how to deal with indirect visibility will be discussed in Section 6.1). We also ignore texture data on the reflector, assuming that the reflector has a uniform BRDF. How to incorporate texture will be explained in Section 5. The one-bounce radiance from x towards \mathbf{o} can then be computed as an integration over a spherical triangle:

$$L_x(\mathbf{o}) = \int_{\Omega_T} L(\mathbf{r}) \rho_x(-\mathbf{r}, \mathbf{o}) \max(-\mathbf{r} \cdot \mathbf{n}_x, 0) d\mathbf{r}, \quad (4)$$

where \mathbf{r} is the direction from a point y on the reflector triangle T to x , and the integral is over the spherical triangle Ω_T subtended by T ; ρ_x and \mathbf{n}_x are the BRDF and normal direction at the receiver point x ; $L(\mathbf{r})$ is the reflected radiance from y to x , defined as

$$L(\mathbf{r}) = \int_{\Omega} G_l(\mathbf{i}) \rho_T(\mathbf{i}, \mathbf{r}) \max(\mathbf{i} \cdot \mathbf{n}_T, 0) d\mathbf{i}, \quad (5)$$

where ρ_T , \mathbf{n}_T are the BRDF and normal of triangle T , respectively, and $G_l(\mathbf{i}) = G(\mathbf{i}; \mathbf{i}_l, \lambda_l)$ is the incident SG light as described before.

4.1 Evaluating the Reflected Radiance $L(\mathbf{r})$

To evaluate the reflected radiance $L(\mathbf{r})$ defined in Eq. (5), we first represent the BRDF ρ_T of the triangle T as a sum of SGs (as shown in

Eq. (3)): $\rho_T(\mathbf{i}, \mathbf{r}) \approx \sum_{j=0}^n G(\mathbf{i}; \mathbf{r}_T^j, \lambda_T^j, c_T^j)$. For notation simplicity, in the following, we omit the summation $\sum_{j=0}^n (\cdot)$ over index j and rewrite the BRDF approximation as $\rho_T(\mathbf{i}, \mathbf{r}) \approx G(\mathbf{i}; \mathbf{r}_T, \lambda_T, c_T)$ ($G_T(\mathbf{i})$ for short). This yields

$$L(\mathbf{r}) \approx \int_{\Omega} G_l(\mathbf{i}) G_T(\mathbf{i}) \max(\mathbf{i} \cdot \mathbf{n}_T, 0) d\mathbf{i}.$$

Then, since the product of $G_l(\mathbf{i})$ and $G_T(\mathbf{i})$ is still an SG (see Appendix B), and also the cosine factor $\max(\mathbf{i} \cdot \mathbf{n}_T, 0)$ is very smooth, we assume that the cosine factor is a constant and pull it out of the integral [Wang et al. 2009a]; Xu et al. [2011] (see Appendix D):

$$L(\mathbf{r}) \approx \max(\mathbf{i}_T(\mathbf{r}) \cdot \mathbf{n}_T, 0) \int_{\Omega} G_l(\mathbf{i}) \cdot G_T(\mathbf{i}) d\mathbf{i}, \quad (6)$$

where $\mathbf{i}_T(\mathbf{r}) = (\lambda_l \mathbf{i}_l + \lambda_T \mathbf{r}_T) / \|\lambda_l \mathbf{i}_l + \lambda_T \mathbf{r}_T\|$ is the center direction of the product SG. The product integral of two SGs can further be well approximated by a single SG (see Appendix C):

$$\int_{\Omega} G_l(\mathbf{i}) \cdot G_T(\mathbf{i}) d\mathbf{i} \approx c_R(\mathbf{r}) \exp(\lambda_R(\mathbf{r}_T \cdot \mathbf{i}_l - 1)), \quad (7)$$

where $c_R(\mathbf{r}) = 2\pi c_T / \|\lambda_l \mathbf{i}_l + \lambda_T \mathbf{r}_T\|$, $\lambda_R = \lambda_T \lambda_l / (\lambda_T + \lambda_l)$. Besides, the dot product $\mathbf{r}_T \cdot \mathbf{i}_l$ satisfies:

$$\mathbf{r}_T \cdot \mathbf{i}_l = (2(\mathbf{r} \cdot \mathbf{n}_T) \mathbf{n}_T - \mathbf{r}) \cdot \mathbf{i}_l = \mathbf{r} \cdot (2(\mathbf{i}_l \cdot \mathbf{n}_T) \mathbf{n}_T - \mathbf{i}_l) = \mathbf{r} \cdot \mathbf{i}_R,$$

where $\mathbf{i}_R = 2(\mathbf{i}_l \cdot \mathbf{n}_T) \mathbf{n}_T - \mathbf{i}_l$. Hence the integral of the product SG can be rewritten as

$$\begin{aligned} \int_{\Omega} G_l(\mathbf{i}) \cdot G_T(\mathbf{i}) d\mathbf{i} &\approx c_R(\mathbf{r}) \exp(\lambda_R(\mathbf{r} \cdot \mathbf{i}_R - 1)) \\ &= c_R(\mathbf{r}) G(\mathbf{r}; \mathbf{i}_R, \lambda_R). \end{aligned}$$

By substituting the previous equation to Eq. (6), the reflected radiance $L(\mathbf{r})$ can finally be evaluated as

$$L(\mathbf{r}) \approx F(\mathbf{r}) G(\mathbf{r}; \mathbf{i}_R, \lambda_R), \quad (8)$$

where $F(\mathbf{r}) = c_R(\mathbf{r}) \max(\mathbf{i}_T(\mathbf{r}) \cdot \mathbf{n}_T, 0)$ is a function much smoother than the SG. Thus the reflected radiance $L(\mathbf{r})$ can be efficiently approximated as a linear sum of the product of a smooth function and an SG (Eq. (8)). Note that the main approximation here is the product integral approximation in Eq. (7), that is, approximating the product integral of two SGs again using an SG. This approximation will produce large error only when the sharpness of both SGs is small. Such cases can be avoided by restricting the sharpness of the SG light. Detailed derivation and analysis of the product integral approximation can be found in Appendix C.

4.2 Evaluating the Interreflection Radiance $L_x(\mathbf{o})$

To finally determine the one-bounce outgoing radiance $L_x(\mathbf{o})$ defined in Eq. (4), similar to before, we represent the BRDF ρ_x at the receiver point x as a sum of SGs (as shown in Eq. (3)): $\rho_x(-\mathbf{r}, \mathbf{o}) \approx \sum_{j=0}^n G(\mathbf{r}; -\mathbf{o}_x^j, \lambda_x^j, c_x^j)$. For denotation simplicity, we again omit the summation $\sum_{j=0}^n (\cdot)$ over index j in what follows and substitute the reflected radiance $L(\mathbf{r})$ (Eq. (8)) into Eq. (4):

$$L_x(\mathbf{o}) \approx \int_{\Omega_T} H(\mathbf{r}) G(\mathbf{r}; \mathbf{i}_R, \lambda_R) G(\mathbf{r}; -\mathbf{o}_x, \lambda_x, c_x) d\mathbf{r},$$

where $H(\mathbf{r}) = F(\mathbf{r}) \cdot \max(-\mathbf{r} \cdot \mathbf{n}_x, 0)$ is again a smooth function. Since the product of two SGs is still an SG, the preceding equation can be rewritten as

$$L_x(\mathbf{o}) \approx \int_{\Omega_T} H(\mathbf{r}) G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) d\mathbf{r},$$

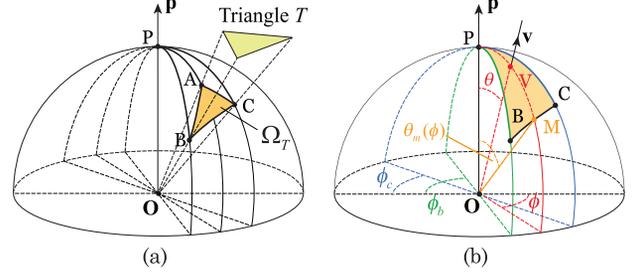


Fig. 3. Integrating an SG over a spherical triangle Ω_T .

where $G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h)$ is the product of the two SGs (the formulas for \mathbf{r}_h, λ_h and c_h can be found in Appendix B). Since function H is intrinsically smooth, we can pull it out of the integral and rewrite the aforesaid equation as

$$L_x(\mathbf{o}) \approx H(\mathbf{r}'_h) \int_{\Omega_T} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) d\mathbf{r}. \quad (9)$$

The representative direction \mathbf{r}'_h , which is used for querying the constant value of function H , is set to be a linear interpolation of the SG's center direction \mathbf{r}_h and the direction from the origin to the center of the spherical triangle Ω_T . The interpolation weight is determined by the area size of Ω_T and SG width, which determines the spanned area of SG. Such an interpolation method is attributed to the fact that the optimal choice of the representative direction varies in different scenarios. For example, if the area size of Ω_T is much larger than the area spanned by SG (e.g., SG just spans a small area inside the spherical triangle), the best choice of the representative direction is the SG center; and conversely, if the spanned area of SG is much larger, the best choice is the triangle direction. The formula of \mathbf{r}'_h is described in Appendix G.

The remaining question in Eq. (9) is how to evaluate the integral of an SG over a spherical triangle subtended by the planar triangle T . Under normal circumstances, this integral does not have a closed-form solution. Fortunately, as shown in Section 4.3, this integral can be reduced to a 1D integral, which can then be accurately evaluated using a piecewise linear approximation.

4.3 Integrating an SG over a Spherical Triangle

Given an SG $G(\mathbf{v}; \mathbf{p}, \lambda)$, we want to derive how to integrate it over a spherical triangle Ω_T , in other words, compute $\int_{\Omega_T} G(\mathbf{v}; \mathbf{p}, \lambda) d\mathbf{v}$. As shown in Figure 3(a), the local frame is defined by setting the SG center direction \mathbf{p} as the zenith direction. Denote the origin of the unit sphere as O and the zenith point as P , and the vertices of the spherical triangle Ω_T as A , B , and C , respectively. It is obvious that the spherical triangle Ω_T satisfies

$$\Omega_T = \Omega_{\Delta ABC} = \Omega_{\Delta PBC} - \Omega_{\Delta PAB} - \Omega_{\Delta PCA}.$$

The plus/minus sign may vary depending on the relative positions of the zenith point P and spherical triangle $\Omega_{\Delta ABC}$ (e.g., when P is inside $\Omega_{\Delta ABC}$, $\Omega_{\Delta ABC} = \Omega_{\Delta PBC} + \Omega_{\Delta PAB} + \Omega_{\Delta PCA}$). Without loss of generality, we first consider how to evaluate the integral over spherical triangle ΔPBC : $\int_{\Omega_{\Delta PBC}} G(\mathbf{v}; \mathbf{p}, \lambda) d\mathbf{v}$. Relying on the coordinate system defined in Figure 3, we can rewrite the integral using the spherical coordinates to simplify the derivation.

As shown in Figure 3(b), we denote the azimuthal angles of point B and C as ϕ_b and ϕ_c , respectively, the intersection point of an arbitrary direction \mathbf{v} with the unit sphere as V , the polar and azimuthal angle of \mathbf{v} as (θ, ϕ) . Since both arcs PB and PC are

longitude arcs, we can rewrite the integral in spherical coordinates by integrating the azimuthal angle from ϕ_c to ϕ_b

$$\begin{aligned} \int_{\Omega_{\Delta PBC}} G(\mathbf{v}) d\mathbf{v} &= \int_{\phi_c}^{\phi_b} \left(\int_0^{\theta_m(\phi)} G(\mathbf{v}; \mathbf{p}, \lambda) \sin \theta d\theta \right) d\phi \\ &= \int_{\phi_c}^{\phi_b} \left(\int_0^{\theta_m(\phi)} e^{\lambda(\cos \theta - 1)} \sin \theta d\theta \right) d\phi, \end{aligned}$$

where $\theta_m(\phi)$ is the maximal allowed polar angle when the azimuthal angle is ϕ . As shown in Figure 3(b), it is the polar angle of the intersection point M of arc \widehat{BC} and the longitude arc \widehat{PV} . It is easy to find out that the inner integral of polar angle θ has an analytic solution, so that the previous equation can be rewritten as

$$\begin{aligned} \int_{\Omega_{\Delta PBC}} G(\mathbf{v}) d\mathbf{v} &= \int_{\phi_c}^{\phi_b} \left(-\frac{1}{\lambda} e^{\lambda(\cos \theta - 1)} \Big|_0^{\theta_m(\phi)} \right) d\phi \\ &= \frac{1}{\lambda} \int_{\phi_c}^{\phi_b} (1 - e^{\lambda(\cos \theta_m(\phi) - 1)}) d\phi. \end{aligned}$$

Through some further derivations using geometric properties, we found that the cosine of the maximal allowed polar angle $\cos \theta_m(\phi)$ can be written as (proof can be found in Appendix H):

$$\cos \theta_m(\phi) = \sin(\phi + \phi_0) / \sqrt{m^2 + \sin^2(\phi + \phi_0)}, \quad (10)$$

where the two parameters ϕ_0 and m can be calculated through the spherical coordinates of the two vertices B and C . Putting everything together, the integral of $G(\mathbf{v})$ can be rewritten as

$$\begin{aligned} \int_{\Omega_{\Delta PBC}} G(\mathbf{v}) d\mathbf{v} &= \frac{\phi_b - \phi_c}{\lambda} - \frac{1}{\lambda} \int_{\phi_c}^{\phi_b} e^{\lambda \left(\frac{\sin(\phi + \phi_0)}{\sqrt{m^2 + \sin^2(\phi + \phi_0)}} - 1 \right)} d\phi \\ &= \frac{\phi_b - \phi_c}{\lambda} - \frac{1}{\lambda} \int_{\phi_1}^{\phi_2} f_{m,\lambda}(\phi) d\phi, \end{aligned} \quad (11)$$

where $\phi_1 = \phi_c + \phi_0$, $\phi_2 = \phi_b + \phi_0$, and the 1D function $f_{m,\lambda}(\phi)$ is defined as

$$f_{m,\lambda}(\phi) = \exp \left[\lambda \left(\frac{\sin \phi}{\sqrt{m^2 + \sin^2 \phi}} - 1 \right) \right]. \quad (12)$$

Thus the original double integral has been simplified to a 1D integral through analytical derivations. Since $\int f_{m,\lambda}(\phi) d\phi$ does not have an analytic solution, we need to evaluate it numerically. A straightforward solution is to precompute a 3D table of preintegrated values, with respect to the three parameters (m, λ, ϕ) . However, the value of $\int f_{m,\lambda}(\phi) d\phi$ in fact changes rapidly when parameter m is very small, and thus using a precomputed table with finite resolution can lead to severe artifacts. To address this issue, shortly we introduce a nonuniform piecewise linear approximation, which works very well for accurate evaluation of $\int f_{m,\lambda}(\phi) d\phi$.

Figure 4 shows the plots of the 1D function $f_{m,\lambda}(\phi)$ under different parameter settings. Observing the overall shape of the function, it is noticeable that the function curve changes smoothly with respect to some parameters while sharply with respect to other parameters. Hence, we can approximate $f_{m,\lambda}(\phi)$ by a nonuniform piecewise linear function. First, we empirically find 4 knots (as highlighted in red in Figure 4) to partition the function in the range $\phi \in [0, \pi]$ into 5 initial segments. The details of how to find the knots are given in Appendix I. Next, we split the integral range $[\phi_1, \phi_2]$ into a few intervals using the selected knots as splitting points. For example, if no knot lies in the range $[\phi_1, \phi_2]$, no splitting is needed; if one knot at ϕ_k lies in the range $[\phi_1, \phi_2]$, we split it into two intervals $[\phi_1, \phi_k]$ and $[\phi_k, \phi_2]$. Finally, we approximate the function in each

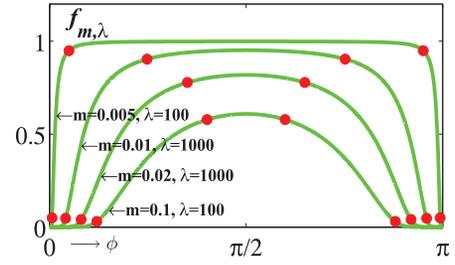


Fig. 4. Plot of the 1D function $f_{m,\lambda}(\phi)$ and its nonuniform knots (i.e., sample points) at different parameter settings.

interval using a uniformly sampled piecewise linear function with K partitions. We set $K = 3$ in the implementation. Due to the way the knots are selected, the function within each interval is very smooth. Hence, such an approximation works very well, and the related evaluation is presented in Section 7.1.

Summary. In this section, we have derived a formula for computing one-bounce interreflection from a triangle to a shading point due to a distant SG light (Eq. (4)). Further, we demonstrated that this formula can be efficiently evaluated using a nonuniform piecewise linear approximation of a 1D function (Eq. (12)).

5. HIERARCHICAL STRUCTURE

Utilizing the previously derived formula, for simple scenes with small number of triangles, we can directly sum up the interreflection contributions from all triangles to a shading point. However, the cost is linear to the number of triangles, thus this approach would perform poorly for complex scenes with many triangles.

Inspired by previous hierarchical integration methods, such as hierarchical radiosity [Hanrahan et al. 1991], Lightcuts [Walter et al. 2005] and the micro-rendering technique [Ritschel et al. 2009], we propose a hierarchical scheme to efficiently sum up the contributions from all triangles in the scene. Specifically, as shown in Figure 5(a), we organize the scene triangles into a binary tree. The leaf nodes are individual triangles, and the interior nodes are subsets of triangles, each owning the triangles that belong to its two child nodes. The binary tree is built in a top-down fashion during the scene initialization step. Starting from the root node, which owns all the triangles, each node is recursively split into two child nodes until reaching the leaf node.

To define the splitting criterion, we first define a 6D feature $(I, m\mathbf{n})$ for each triangle, where I is the triangle center (the scene's bounding box is normalized into the range $[-1, 1]$) and \mathbf{n} is the triangle normal. m is a scalar weight that controls the relative importance of I and \mathbf{n} , and we usually set $m = 5$. Next, to split a node, we compute the principal direction (using PCA) of the 6D features of all triangles belonging to that node, and perform a median split along the principal direction. This ensures that the splitting is done along the direction of maximum variance. The result of the median split produces two child nodes with equal number of triangles. For nontextured scenes, each node stores the average center and normal of its triangles, as well as the bounding box, the bounding normal cone, and the total triangle area, as shown in Figure 5(a). To deal with textured scenes, each node additionally stores the average and the largest/smallest texture color values of its triangles, which are obtained by enumerating over the pixels covered by the projected triangles in the texture space. All these stored terms are used later to approximate the reflected radiance from the node to a receiver

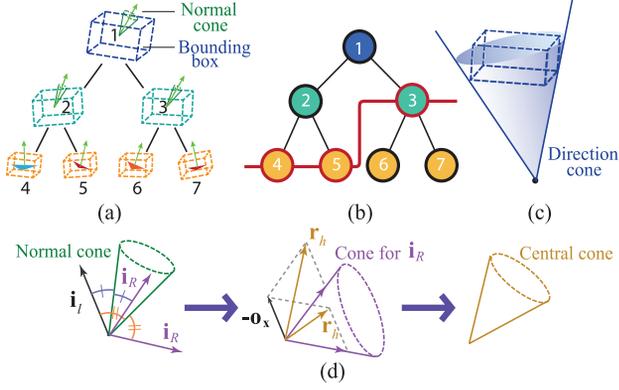


Fig. 5. Hierarchical structure and error bound estimation. (a) a binary tree of triangles showing the bounding box and normal cone stored at each node; (b) an example of reflector cut; (c) direction cone; (d) computing central cone from the normal cone.

point (Section 5.1), and to evaluate the associated upper bound of its error (Section 5.2).

During the rendering stage, we employ a similar strategy as in the Lightcuts [Walter et al. 2005] to efficiently evaluate the one-bounce interreflections, the pseudocode of which is given in Algorithm 1. For each receiver point, we start from an initial *cut* that contains only the root node of the binary tree, and then iteratively refine it. At each iteration, we pick the node with the highest error bound and

ALGORITHM 1: Pseudocode of interreflection computation using reflector cut

```

for each shading pixel  $x$  do
  compute radiance  $L_R$  from root node  $R$  (Sec. 5.1);
  compute error bound  $E_R$  for root node  $R$  (Sec. 5.2);
  initialize the reflector cut  $C$  containing only root node  $R$ ;
  initialize the total unshadowed radiance  $L_u = L_R$ ;
  while  $cut\ size < 1000$  do
    pick the node  $N$  with the largest error bound  $E_N$ ;
    if  $E_N < 1\% \times L_u$  then break remove node  $N$  from
    cut  $C$ , and update  $L_u = L_u - L_N$ ;
    for each child node  $M$  of node  $N$  do
      if  $M$  is leaf node then
        compute  $L_M$  from triangle  $M$  (Sec. 4);
      else
        compute  $L_M$  from node  $M$  (by Sec. 5.1);
      end
      insert node  $M$  to cut  $C$ , and update
       $L_u = L_u + L_M$ ;
      compute error bound  $E_M$  for node  $M$  (Sec. 5.2);
    end
  end
  Initialize the shadowed interreflected radiance  $L_s = 0$ ;
  for each node  $N$  in the cut  $C$  do
    query the indirect visibility of node  $N$ ;
    modulate radiance from  $N$  by visibility and add to
     $L_s$ ;
  end
end

```

replace it by its two child nodes. The iteration stops either when the largest error bound falls below a threshold (in our case, 1% of the total estimated reflected radiance), or a predefined maximum number of nodes in the generated cut is reached (1000 in our case). When the iteration terminates, we refer to the resulting cut as the *reflector cut* (Figure 5(b)), and the number of nodes in the reflector cut as *cut size*. Note that if any leaf node is reached during the iteration, we can directly use the interreflection model for a single triangle reflector to accurately evaluate its one-bounce contribution. However, this strategy only works well for nontextured scenes since our derivations in Section 4 assume that the reflector triangle has a uniform BRDF across it without texture variations. Hence, for any leaf node with a textured triangle, we still compute its error bound using its stored texture information. If the error bound is larger than a predefined threshold, we subdivide this triangle into 4 subtriangles using $\sqrt{3}$ -subdivision [Kobbelt 2000], and use the 4 subtriangles to further evaluate one-bounce contributions. Such a *dynamic subdivision* strategy can be applied iteratively until the estimated error bounds of the subdivided triangles are sufficiently small. For a subdivided triangle, the terms such as its averaged center and bounding box are calculated on-the-fly, the largest/smallest texture values are inherited from its parent, and the averaged texture value is simply set as the texture value at the subdivided triangle center.

Next, we will explain in detail how to evaluate the estimated one-bounce radiance reflected by a node (Section 5.1) and the associated error bound (Section 5.2).

5.1 Estimating the Interreflected Radiance

Given a node N and a receiver point x , we estimate the radiance from x towards the view direction \mathbf{o} , due to the reflection of an SG light $G(\mathbf{i}; \mathbf{p}_i, \lambda_i)$ from node N . To begin, we denote the center position of node N as I_N , its average triangle normal as \mathbf{n}_N , its triangle area as Δ_N , and its average texture color value as \bar{t}_N . We can reuse Eq. (9) derived for a single triangle, only changing the integration area from a spherical triangle Ω_T to a spherical region Ω_N spanned by all the triangles belonging to node N .

$$L_x(\mathbf{o}) \approx \bar{t}_N \cdot H(\mathbf{r}'_h) \int_{\Omega_N} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) d\mathbf{r}. \quad (13)$$

However, since the shape of the spherical region Ω_N is unknown (it is a spherical region subtended by a set of triangles), we cannot directly apply the piecewise linear approximation described in Section 4. To address this issue, we rewrite the integral of an SG over spherical region Ω_N as the SG multiplied by a binary mask integrated over the whole sphere:

$$\int_{\Omega_N} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) d\mathbf{r} = \int_{\Omega} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) V_{\Omega_N}(\mathbf{r}) d\mathbf{r},$$

where Ω denotes the whole sphere, $V_{\Omega_N}(\mathbf{r})$ is a binary function that indicates if a direction \mathbf{r} is inside Ω_N . We further approximate the binary mask V_{Ω_N} using an SG: $V_{\Omega_N}(\mathbf{r}) \approx G(\mathbf{r}; \mathbf{r}_N, \lambda_N, c_N)$ (G_N for short). The center direction \mathbf{r}_N is set to be the unit direction from node center I_N to the receiver point x ; the sharpness and coefficient are determined by preserving the function energy and variance (see Appendix E): $\lambda_N = 4\pi / \|\Omega_N\|$, $c_N = 2$. Here $\|\Omega_N\|$ is the solid angle computed as

$$\|\Omega_N\| \approx (\Delta_N \cdot \max(\mathbf{r}_N \cdot \mathbf{n}_N, 0)) / d_N^2, \quad (14)$$

where d_N is the distance from the receiver point x to node center I_N . Hence, the one-bounce interreflected radiance $L_x(\mathbf{o})$ in Eq. (13)

can be approximated as

$$L_x(\mathbf{o}) \approx \bar{I}_N \cdot H(\mathbf{r}'_h) \int_{\Omega} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h) G_N(\mathbf{r}) d\mathbf{r}. \quad (15)$$

Note that, in the preceding equation, the product integral of two SGs has an analytic solution (see Appendix C) and hence $L_x(\mathbf{o})$ can be easily computed.

Summary. By approximating the binary function V_{Ω_N} using an SG, we can efficiently approximate the one-bounce interreflected radiance $L_x(\mathbf{o})$ by an analytic solution (Eq. (15)). However, we note that this approximation will produce large errors in the case of integrating SGs with small widths over nodes with large solid angles. Thus we need to estimate the error bound of this approximation to ensure its accuracy. If the estimated error of the current node is larger than a predefined threshold, we replace it by its two child nodes to reduce the overall approximation error.

5.2 Estimating the Error Bound

Given the bounding box and the normal cone of a node, we aim to evaluate the upper bound of the error in computing $L_x(\mathbf{o})$ using Eq. (15). Denoting the largest and smallest values of the SG function $G(\mathbf{r}; \mathbf{r}_h, \lambda_h)$ in region Ω_N as g_{max} and g_{min} , the largest and smallest possible solid angle of $\|\Omega_N\|$ as $\|\Omega\|_{max}$ and $\|\Omega\|_{min}$, and the largest and smallest texture color values of node N as t_{max} and t_{min} , obviously the error can be conservatively bounded by $H(\mathbf{r}'_h) \cdot (t_{max} \cdot g_{max} \cdot \|\Omega\|_{max} - t_{min} \cdot g_{min} \cdot \|\Omega\|_{min})$. Note that the bounds on texture color values $[t_{min}, t_{max}]$ have already been stored in each node. In the following, we will explain how to more accurately estimate the bounds on the solid angles and the SG function values, respectively.

Bounds on the solid angle. Based on Eq. (14), we can compute the bounds on the solid angle $\|\Omega_N\|$ by estimating the bound of the distance d_N from the receiver point \mathbf{x} to the node, and the bound of the dot product $\mathbf{r}_N \cdot \mathbf{n}_N$. Given the bounding box of the node, as shown in Figure 5(c), it is trivial to compute the lower and upper bounds of d_N , which are denoted as d_{min} and d_{max} , respectively. The spherical region Ω_N spanned by the node as observed from the receiver point can be bounded by a cone, which is referred to as *direction cone*. Note that direction \mathbf{r}_N is also bounded within the direction cone. Since the normal direction \mathbf{n}_N is already bounded by the normal cone of the node, we can easily compute the lower and upper bounds of the angle between \mathbf{r}_N and \mathbf{n}_N using these two cones. We denote the lower and upper bounds of the angle as θ_d^{min} and θ_d^{max} , respectively. Hence, the lower and upper bounds of the solid angle $\|\Omega_N\|$ can be computed as: $\|\Omega\|_{min} = \Delta_N \cdot \cos \theta_d^{max} / d_{max}^2$, $\|\Omega\|_{max} = \Delta_N \cdot \cos \theta_d^{min} / d_{min}^2$. More details on how these bounds are derived can be found in Appendix J.

Bounds on the SG function values. Estimating the bounds on the SG function $G(\mathbf{r}; \mathbf{r}_h, \lambda_h)$ requires computing the bound of the dot product $(\mathbf{r} \cdot \mathbf{r}_h)$. The direction \mathbf{r} is naturally bounded by the direction cone, since it is restricted in the integral area Ω_N . To find a bounding cone for the SG central direction \mathbf{r}_h , recall the formula for defining \mathbf{r}_h (Eq. (9)): \mathbf{r}_h is calculated from another direction \mathbf{i}_R , while \mathbf{i}_R is computed using the normal \mathbf{n}_N ($\mathbf{i}_R = 2(\mathbf{i}_l \cdot \mathbf{n}_N) \mathbf{n}_N - \mathbf{i}_l$, see derivations before Eq. (8)). Hence, as shown in Figure 5(d), we can first determine a bounding cone for direction \mathbf{i}_R based on the normal cone, and then find a bounding cone for direction \mathbf{r}_h , which is referred to as *central cone*. Finally, the lower and upper bounds of the angle between direction \mathbf{r} and the SG central direction \mathbf{r}_h can be computed from the direction cone and the central cone,

which are denoted as θ_r^{min} and θ_r^{max} , respectively. Putting everything together, the lower and upper bounds of the SG values are: $g_{min} = \exp(\lambda_h(\cos \theta_r^{max} - 1))$, and $g_{max} = \exp(\lambda_h(\cos \theta_r^{min} - 1))$.

6. IMPLEMENTATION

6.1 Visibility

So far the interreflection computation described in Section 4 and Section 5 does not account for occlusion. In this section, we describe how to handle visibility properly in our method. We use the term *direct visibility* to denote the visibility from an SG light to the reflector, and *indirect visibility* to denote the visibility from the reflector to the receiver. To evaluate the direct visibility, we compute 16 sample points uniformly distributed on the reflector triangle and the shadow value for each point is computed by querying the Variance Shadow Maps (VSM) [Donnelly and Lauritzen 2006] of the SG light. The average shadow value of all the sample points is then stored as the direct visibility for each reflector triangle. As for the indirect visibility, we adopt a variant of the Imperfect Shadow Maps (ISM) [Ritschel et al. 2008]. Specifically, during the scene initialization stage, we select 200 random sample points on each model to represent the Virtual Light (VL) positions and capture ISMs for them at runtime. For each node in the binary tree of the model, its three closest VLs are computed and stored. Then for each node we need to find how its value will be linearly interpolated from the three closest VLs. To do so, we project the center of the node onto the triangle plane spanned by the three closest VLs, and compute the weights by the barycentric coordinates of the projected position. During the rendering stage, an ISM with resolution 128×128 is generated for each VL. Then, after the reflector cut is determined, the indirect visibility for each node in the cut is interpolated from the ISMs of its three closest VLs using the associated weights. This indirect visibility approximation will be evaluated in Section 7.1.

6.2 Implementation Details

Algorithmic Pipeline. The implementation of our algorithm consists of an initialization stage which loads the scene and builds the binary tree structure, and a runtime rendering stage which implements the one-bounce interreflection algorithm.

In the scene loading stage, we first build a binary tree for each model (i.e., a triangle mesh) following the algorithm described in Section 5. If the number of triangles in a model is less than 200, we skip the tree building step for that model since it is more efficient to just iterate over all triangles of the model to calculate indirect illumination. This initialization stage only takes a few seconds and does not need to be performed again unless a model deforms.

During the runtime rendering stage, we first evaluate the direct illumination in a separate pass. To compute interreflections, we need to determine a reflector cut (Section 5) for each shading pixel. We follow Cheslack-Postava et al. [2008] to compute a per-vertex reflector cut, which is then interpolated at each shading pixel using the cut merging algorithm, which makes use of the per-vertex cut stored at the nearby vertices around a pixel. Finally, we compute interreflection for each shading pixel using the merged reflector cut in a pixel shader.

Lights. Our method naturally supports all-frequency incident lighting, because spherical Gaussian has varying widths which can represent both high- and low-frequency lights. Following Wang et al. [2009a], different types of lights, including distant environment lights, distant area lights, and local spherical lights, can be

easily integrated into our algorithm. An environment light can be fitted into a small number of SG lights using the method presented in Tsai and Shih [2006]. As shown in Wang et al. [2009a], for a distant area light with direction \mathbf{p}_l , solid angle $\|\Omega_l\|$, and intensity c , the approximated SG light is given by: $L(\mathbf{i}) \approx G(\mathbf{i}; \mathbf{p}_l, 4\pi/\|\Omega_l\|, 2c)$. For a local spherical light located at position \mathbf{l} with radius r and intensity c , the approximated SG light towards surface position \mathbf{y} is given by: $L(\mathbf{i}) \approx G(\mathbf{i}; (\mathbf{l} - \mathbf{y})/\|\mathbf{l} - \mathbf{y}\|, 4\|\mathbf{l} - \mathbf{y}\|^2/r^2, 2c)$.

Cut Selection. To improve performance, we implement the per-vertex cut selection algorithm using CUDA. The cut selection algorithm requires a priority queue structure to store tree nodes, since it needs to pick the node with the largest error and to replace it with its two children. However, an accurate priority queue implementation using CUDA is not yet efficient, thus we employ an approximate priority queue. Specifically, in our implementation, we manage 5 ordinary queues with different priorities. The queue with the i -th priority stores nodes whose errors are larger than $2^{5-i}\%$. For example, the first queue stores nodes with errors larger than 16%, the second queue stores nodes with error larger than 8%, and so on. At each step, we pick the front node in the queue with the highest priority (e.g., pick node from the first queue if it is nonempty, otherwise pick a node from the second queue, and so on) and split it into two child nodes. The two child nodes are then pushed into the queues with the corresponding priorities according to their errors. Note that, for simple scenes without a tree structure, the per-vertex cut selection is not necessary.

Final Shading. The final one-bounce interreflection is evaluated in a pixel shader. At each pixel, the required data of its reflector cut, including position, normal, BRDF, and direct visibility of each cut node, as well as the SG lights, are stored in textures and passed into the pixel shader. Hence, the total contributions from all nodes in the cut to the pixel can be computed by iterating through every cut node. The nonlinear piecewise linear approximation (Section 4) for reflector triangles is used for leaf nodes, while the node approximation (Eq. (15)) is used for interior nodes. The contribution of each node is modulated by both direct and indirect visibility. The direct visibility is obtained as a pass-in parameter of each node, and the indirect visibility is computed by interpolating the queried shadow values using the ISMs.

7. EVALUATIONS AND RESULTS

In this section, we provide a comprehensive evaluation of both the accuracy and the performance of our method. Furthermore, we compare our method with the state-of-the-art interreflection rendering techniques to demonstrate its effectiveness and versatility.

7.1 Accuracy of Our Method

For validation, we first examine the accuracy of our one-bounce interreflection model (Section 4) in handling all-frequency isotropic BRDFs as well as anisotropic BRDFs. Next, we evaluate how the choices of the error bound threshold employed in the reflector cut (Section 5) impact the rendering accuracy. Finally, we examine the error of the visibility approximation (Section 6.1).

Accuracy of our interreflection model. In Figure 6, we show the results of our one-bounce interreflection model using piecewise linear approximation (Section 4.3) with varying glossiness of reflector BRDFs. We further compare our results with those generated by a VPL-based method [Keller 1997], and with path-traced reference. The test scene consists of a single equilateral triangle placed perpendicular to the receiver plane, and a directional light at

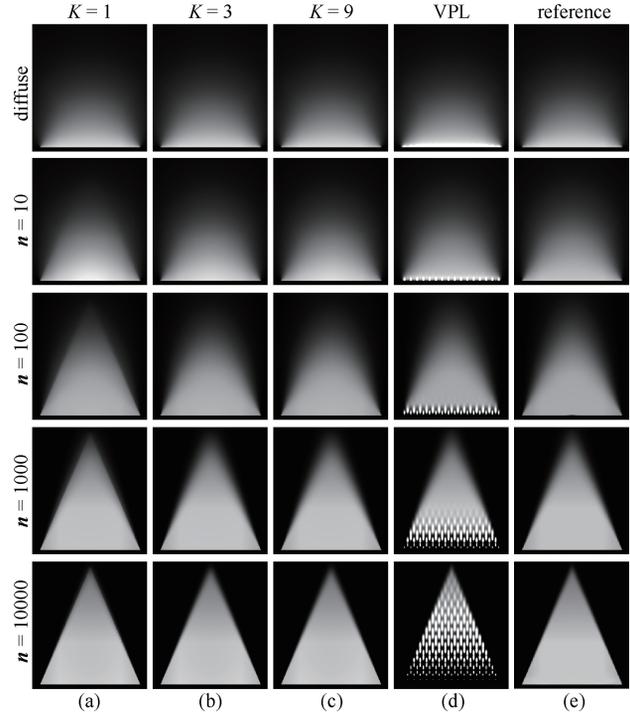


Fig. 6. Comparisons of one-bounce interreflection with varying glossy shininess of BRDFs. In the top row, the reflector triangle is diffuse; from row 2 to row 5, the reflector triangle has a Blinn-Phong BRDF with glossy shininess of $n = 10, 100, 1000, 10000$ respectively. Columns (a), (b), and (c) show results generated by our model with different number of piecewise linear partitions $K = 1, K = 3$, and $K = 9$; column (d) shows the results using VPL-based method with 256 VPLs; column (e) gives the ground-truth reference.

a 45 degree angle to the ground plane. The ground (receiver) plane has a Lambertian BRDF and we only show the interreflection (i.e., no direct illumination) from the triangle to the plane in this figure. From the top row to the bottom row, the BRDF of the reflector triangle varies from purely diffuse to highly specular. Note that our result with partition number $K = 3$ (column (b)) already matches the reference image (column (e)) very well for all tested BRDFs. In contrast, the VPL-based method (column (d)) with 256 VPLs only works well for low-frequency BRDFs, while producing severe artifacts when the shininess parameter $n \geq 10$. This is mainly due to the limited number of VPLs, which has to be enormously increased for highly glossy BRDFs.

In Figure 7, we further compare our results to path-traced reference with varying glossiness of both the reflector and receiver BRDFs. The scene contains a box placed on a plane with two SG lights. From left to right, the BRDF of the box changes from diffuse to highly glossy; while from top to bottom, the BRDF of the plane varies from diffuse to highly glossy. The comparison of our result (left) with the reference (right) clearly shows that the interreflection results (with all-frequency BRDFs) produced by our method are visually indistinguishable from the references.

We further verify the ability of our method in handling anisotropic BRDFs. As shown in Figure 8, the scene contains a diffuse plane and a disk with an Ashikhmin-Shirley anisotropic BRDF (anisotropic ratio of 4:1). Following Wang et al. [2009a], we approximate the anisotropic BRDF using 7 SGs. Figure 8(a) and (c) show

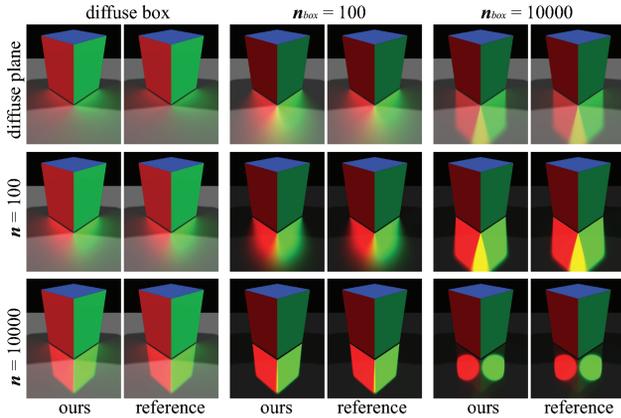


Fig. 7. Comparisons of one-bounce interreflection with varying glossiness of both the reflector and receiver BRDFs.

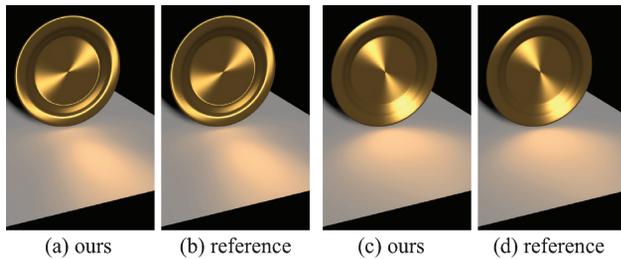


Fig. 8. Results with anisotropic BRDFs.

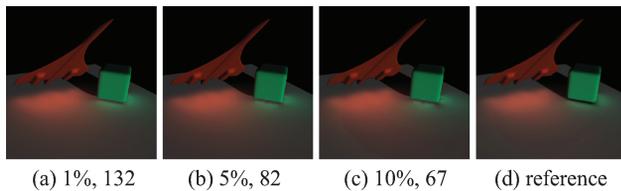


Fig. 9. Evaluation of the error bound threshold. The threshold and average cut size are given in the caption of each subfigure.

the rendering results with two different local frames of the disk, respectively; and the reference results are given accordingly in (b) and (d), respectively. Note that our results only have subtle differences from the references, which demonstrates that our method can deal with complex anisotropic BRDFs, as long as the BRDFs can be accurately approximated by a mixture of SGs.

Error bound threshold. We further evaluate our choice of the error bound threshold. Since smaller thresholds result in better quality but slower framerates, a trade-off should be made. Figure 9 shows the results using different thresholds, including 1%, 5%, and 10%. Using a larger threshold such as 5% gives better performance (refer to Figure 9(b), which has an average cut size of 82 and average fps of 2.3), but it also produces visible artifacts around the contact area between the box and the plane. Reducing the threshold to 1% eliminates these artifacts, but leads to larger average cut size of 132 and average fps of 1.2. In our experiments, we find that 1% is an optimal choice, and all the experimental results are generated by setting the error bound threshold to be 1%.

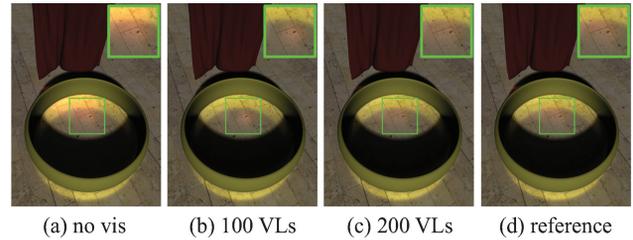


Fig. 10. Evaluation of visibility using different numbers of virtual lights. (a) is result without considering indirect visibility; (b), (c) are results with 100 and 200 virtual lights, respectively; (d) is the path-traced reference. Note that with 200 VLs our result matches the reference very well.

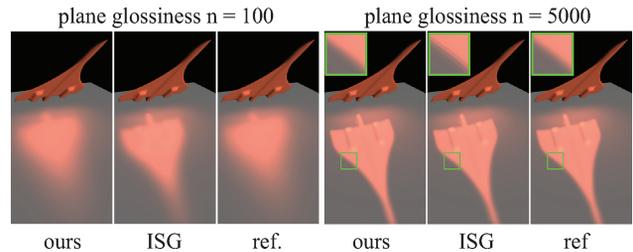


Fig. 11. Comparison to Integral Spherical Gaussian (ISG).

Visibility approximation. Figure 10 evaluates the indirect visibility approximation described in Section 6.1. Note that our result with 200 VLs (Figure 10(c)) matches the reference in (d) very well.

7.2 Comparisons to Other Methods

In this section, we compare our method to the state-of-the-art interreflection rendering methods, which include the Integral Spherical Gaussian (ISG) [Iwasaki et al. 2012b], photon mapping [Jensen 2001], micro-rendering [Ritschel et al. 2009], and also a GPU-based ray-tracer implemented with NVIDIA's Optix [Parker et al. 2010].

Comparison to ISG. We compare our piecewise linear approximation (Section 4.3) with ISG in integrating an SG over a spherical triangle. Note that ISG is not designed specifically for integrating SGs over spherical triangles. Thus we use ISG to compute the integration as follows: First, we discretize the hemisphere into small patches as done in Iwasaki et al. [2012b]; Then, we determine the patches that overlap with the given spherical triangle; Finally, the integral over the spherical triangle is approximated as the sum of the integral over all overlapping patches, where the integral of each patch is calculated by ISG. In Figure 11, we show the results of our method, ISG (with 1600 discretized patches), and reference with both high-frequency and low-frequency reflector BRDFs. ISG produces artifacts for high-frequency BRDFs due to discretization error. While it produces smooth results for low-frequency BRDFs, the difference to the ground truth is quite obvious. This is because the error of the sigmod function approximation used in ISG is large for SGs with low sharpness. In comparison, our method is accurate in both cases.

Comparison to photon mapping. In Figure 12, we compare our method with GPU-based photon mapping. The scene consists of a directional light, a reflective ring, which has a Blinn-Phong BRDF with glossiness $n = 100$, and a Lambertian plane. Figure 12(a) and (b) show our result and photon mapping result using $1M$ photons, respectively. Both results are generated in a

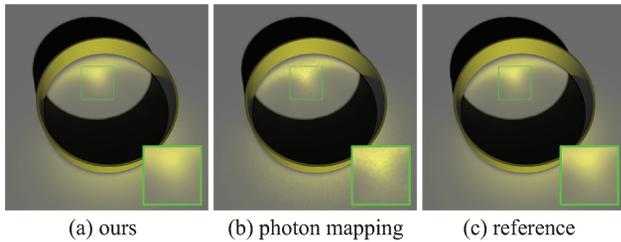


Fig. 12. Comparison with photon mapping on caustics effects.

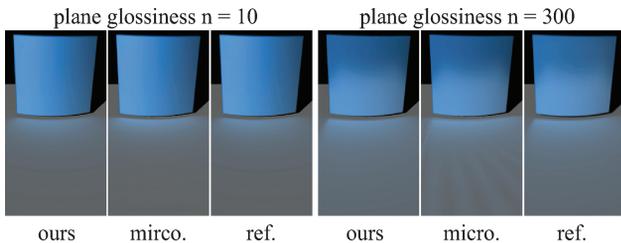


Fig. 13. Comparison to the micro-rendering method [Ritschel et al. 2009].

comparable time period (about 1 second). However, the result using photon mapping exhibits noticeable noise in the caustics region. The reason is that while photon mapping is highly efficient for rendering caustics from perfectly specular reflectors, it becomes inefficient when dealing with glossy materials, such as the BRDF of the ring in this example. This is mainly due to the slow convergence of the stochastic sampling when photons are reflected from a glossy surface. As a result, glossy reflectors require a much larger number of photons to achieve high-quality results. The reference image is shown in Figure 12(c), which is generated by increasing the number of photons to $10M$ to remove noise; but doing so also significantly reduces rendering performance. In contrast, our method, as shown in Figure 12(a), handles the glossy reflector at near-interactive frame-rates with much less noise.

Comparison to micro-rendering and GPU ray tracing. In Figure 13, we compare our method to micro-rendering [Ritschel et al. 2009] with approximately equal rendering time (both at about 0.5 fps). The size of micro-buffer is set to be 24×24 . The scene consists of a curved glossy plate placed on the top of a Lambertian plane. In micro-rendering, the BRDF importance sampling is performed in the final gathering step, and hence it can handle glossy receivers well. However, the micro-rendering method selects reflector nodes from a point hierarchy of scene sample points, in which only the subtended solid angle from each node is considered. Since the BRDF of the reflector node is ignored during cut selection, it may ignore some important nodes with glossy BRDFs, which are responsible for strong interreflections. In contrast, our cut selection scheme takes into account not only the subtended solid angle but also the BRDFs of both the receiver and reflector. Moreover, our method estimates the contribution of each node using integration instead of using a single point sample, leading to more robustness. As shown in Figure 13, when the shininess of the reflector is low at $n = 10$, both our method and micro-rendering generate smooth interreflections. However, when the shininess of the reflector increases to $n = 300$, micro-rendering produces “banding” artifacts. In contrast, our method still renders high-quality interreflections without noticeable artifacts.

Table I. Performance of the Results Shown in this Article

scene	#faces	avg. cut size	fps	shade. time	cut sel. time	fps 720p
magic cube	140	N/A	4.0	0.25s	N/A	1.4
table	104	N/A	1.0	1.0s	N/A	0.35
ring	21k	151	0.7	1.0s	0.4s	0.28
dragon	26k	258	0.4	1.8s	0.5s	0.16
airplane	20k	142	1.2	0.58s	0.19s	0.5
bunny & tweety	36k	316	0.3	2.5s	0.7s	0.12
kitchen	92k	489	0.12	6.9s	1.3s	0.04
sponza	143k	566	0.09	9.0s	2.0s	0.03

From left to right, we give the name of the scene, the number of faces, average cut size, overall fps, final shading time, cut selection time, and the fps for rendering with a 720p resolution (i.e., 1280×720).

Figure 14 gives equal-time comparisons of the results generated by our method, micro-rendering (with micro-buffer size 72×72), and GPU-based ray tracing (OptiX). Notice that these two scenes contain interreflection effects covering all-frequency ranges, including low-frequency interreflection (e.g., lily to ground), glossy reflection (e.g., ring to box, pillar to ground), indirect highlights (e.g., fountain to its bottom), and caustics (e.g., ring to ground). As shown in the close-up views in Figure 14, micro-rendering does not scale well for highly glossy reflectors and its results exhibit banding artifacts, while GPU-based ray tracing still requires more time to reduce the visible noises in the results. In comparison, our method achieves the best quality and the generated interreflections are comparable to the references.

7.3 Results

Our results are reported on a PC with an Intel Xeon 2.27G CPU, 8GB memory, and an NVIDIA Geforce GTX 690 graphics card. The algorithm is implemented using OpenGL 4.0 and CUDA 4.1. The image resolution for all rendered results is 640×480 . We set the partition number $K = 3$ and the error bound threshold to be 1% for all examples. The performance data is reported in Table I.

Simple Scenes. We first verify our method using simple scenes containing a couple hundred of triangles. Figure 15(a) shows results of a magic cube scene with varying BRDFs on the cube and on the plane. In Figure 15(b), we show the results of a star scene and a kitchen scene under different environment lights represented by 10 SGs. Note that these examples demonstrate the capability of our algorithm in producing all-frequency interreflection effects, including diffuse to diffuse, diffuse to specular, and specular to diffuse (i.e., caustics) effects.

More Scenes. Figure 1(a) shows the results of a ring with a Blinn-Phong BRDF changing from highly specular to purely diffuse. From left to right, the glossy shininess of the Blinn-Phong BRDF is set to $n = 10000, 1000, 300, 100$ respectively, and the last one is using a Lambertian BRDF. Note that our method produces convincing caustics on the plane for all examples. Further, our method achieves smooth and coherent transitions (i.e., without flickering) while changing BRDF parameters (see the supplemental video). Thus it is a single algorithm that can reproduce interreflections with all-frequency BRDFs.

Figure 1(c), (d), and (e) and Figure 15(b), (c), and (d) demonstrate our method under different types of lights, including a local light (Figure 1(e)), SG lights with small width (Figure 1(c)), and large width (Figure 15(c) and (d)), as well as environment lights (Figure 15(b)).

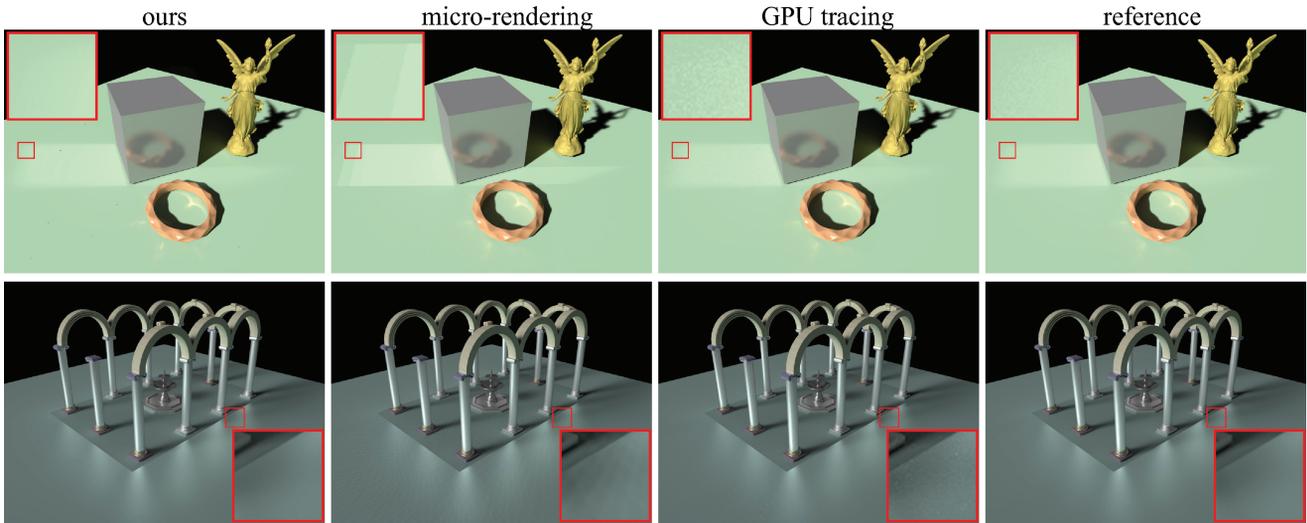


Fig. 14. Equal-time image comparison of our method, micro-rendering and GPU-based ray tracing. The last column shows reference images. For the lucy scene (top) containing 139K triangles, our method, micro-rendering and GPU-based ray tracing take 13.6s, 14.6s, and 14.0s, respectively; for the cloister scene (bottom) containing 118K triangles, our method, micro-rendering and GPU-based ray tracing take 12.2s, 13.0s, and 14.3s, respectively. The Lucy model is courtesy of the Stanford 3D Scanning Repository.

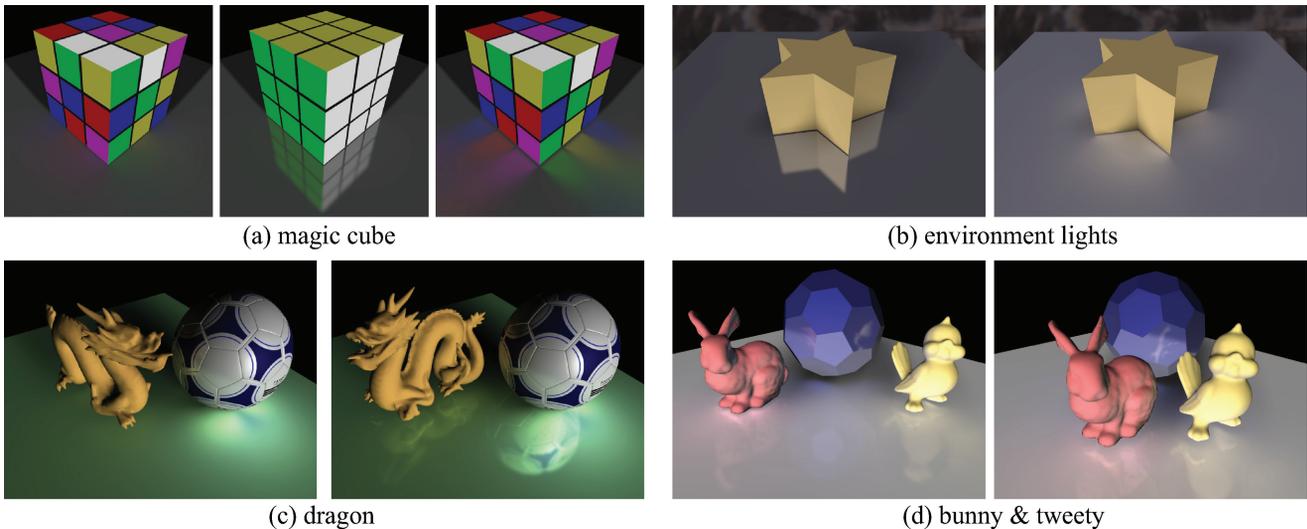


Fig. 15. Additional rendering results. (a) A magic cube model with various BRDF settings (left: plane and cube are both diffuse; middle: cube is diffuse, plane is specular; right: plane is diffuse, cube is specular). (b) A star model and a table scene under environment light. The environment lights are approximated using 10 SGs. (c) A dragon scene. (d) A “Bunny and Tweety” scene. The models in (c), (d) are courtesy of the Stanford 3D Scanning Repository.

Figure 16 shows results of more complex textured scenes. Our method performs well for these scenes that exhibit complex visibility and textures. Please refer to the accompanying video for additional results.

Performance. The performance data, including framerates, average cut size (i.e., the average number of nodes in the *reflector cut*), time for final shading, and time for cut selection, is shown in Table I. We do not list the time for direct lighting and ISMs generations since for all scenes it takes less than 0.1 second. Clearly, the bottleneck lies in the per-pixel final shading stage, especially for complex scenes. Considering the sponza scene in Figure 16(b), the final shading stage takes 81% of the time on average, while other

steps, including the cut selection, ISMs generation, and direct lighting, together only occupy 19% of the overall cost. This is because the average cut size increases with the number of faces. In the final shading step, which is implemented in a fragment shader, for each pixel we need to loop all the nodes in its cut to compute the final shading. Fortunately, this shading computation is easy to scale with parallel processing power, and can thus be greatly accelerated on modern GPUs.

8. DISCUSSIONS AND CONCLUSION

The scope of our method. The goal of our method is to provide a unified algorithm for rendering interreflections with



Fig. 16. Rendering results of complex textured scenes. The sponza scene is courtesy of Frank Meinl, Crytek.

all-frequency BRDFs. Admittedly, our method is less efficient than micro-rendering [Ritschel et al. 2009] or global photon mapping if the scene contains only diffuse or low-frequency BRDFs. On the other hand, our method is also slower than ray tracing in rendering scenes with only highly glossy or specular receivers. However, our method is more efficient than these methods in rendering scenes that simultaneously contain a wide frequency range of BRDFs, as demonstrated in Figure 14. Our method is capable of producing various types of interreflections including diffuse reflection (diffuse to diffuse), caustics (glossy to diffuse), glossy reflection (diffuse to glossy), and indirect highlights (glossy to glossy), in the same unified framework. This goal is shared with some recent works [Jakob and Marschner 2012; Hachisuka et al. 2012; Georgiev et al. 2012], which aimed to render more general light transport effects. Nevertheless, our focus is on providing fast rendering speed and interactive feedback by considering one-bounce interreflections with a wide frequency range of BRDFs.

Limitations. One major limitation of our method is that it currently only supports *one-bounce* interreflections. However, our interreflection model can in fact be extended to handle multiple bounces. Taking two-bounce interreflections as an example, the incident light bounces from a first triangle (referred to as reflector I), then bounces at a second triangle (referred to as reflector II), and finally arrives at a receiver point. Denote the direction from reflector I to reflector II as \mathbf{r}_1 , and the direction from reflector II to receiver point as \mathbf{r}_2 . By making use of the SG approximation formula (Eq. (15)) of the tree node to compute the one-bounce interreflection from reflector I to reflector II, the reflected radiance $L(\mathbf{r}_2)$ from reflector II to receiver can be represented as an SG of \mathbf{r}_2 . Hence, the outgoing radiance from the receiver point can again be approximated using our one-bounce model. The computational cost of computing the second bounce is $O(N^2)$ (where N is the number of reflector triangles), because we need to consider all possible combinations of reflector I and reflector II. However, the computational cost can be greatly reduced to be sublinear by employing a multidimensional reflector cut, similar to Walter et al. [2006]. A complete study of this method to handle multiple-bounce interreflections remains as our future work.

Another limitation is that our method only handles low-frequency indirect shadows. As demonstrated in Ritschel et al. [2008], ISMs

can handle low-frequency indirect shadows well but has difficulty dealing with high-frequency shadows due to the low-resolution ISMs and the low sampling rate of virtual lights. Hence, our method inherits the same limitations and only generates low-frequency indirect shadow. Nevertheless, unoccluded all-frequency interreflection itself is a research challenge, and our method provides a viable solution to efficiently simulate such effects.

Finally, it is worthwhile to address the scalability of the present method in rendering very complex scenes, for example, more complex geometries, highly anisotropic BRDFs, and high-frequency environment light. Without using a hierarchical structure, the computation cost would grow linearly with respect to the number of triangles, the number of SGs needed to represent BRDFs, and the number of SGs needed to approximate lighting. Fortunately, our hierarchical integration method can be easily extended to handle BRDFs or lighting represented by multiple SGs, and hence the computation cost only grows sublinearly.

Conclusion. To summarize, we present a practical algorithm for rendering one-bounce interreflections with all-frequency BRDFs. Our method builds upon a Spherical Gaussian representation of BRDFs and lighting. The core of our method is an efficient algorithm for computing one-bounce interreflection from a triangle to a shading point using an analytic formula combined with a nonuniform piecewise linear approximation. To handle scenes containing lots of triangles, we propose a hierarchical integration method with error bounds that take into account the BRDFs at both the reflector and receiver. The experimental results demonstrate that our method well supports a wide frequency range of BRDFs, from purely diffuse to highly specular, and render, important effects caused by different types of lighting paths, including diffuse to diffuse, diffuse to glossy, glossy to diffuse (i.e., caustics), and glossy to glossy (i.e., indirect highlights), in a single unified framework.

In the future, we would like to extend our method to handle multibounce interreflections. This may be accomplished by recursively applying our one-bounce algorithm and employing a multidimensional reflector cut. We are also interested in extending our method to handle bidirectional texture functions. Another direction is to improve the speed of our algorithm, such as employing ManyLODs [Hollander et al. 2011] to further exploit the spatial and temporal coherence in rendering.

APPENDIXES

A. INTEGRAL OF A SPHERICAL GAUSSIAN (SG)

It is straightforward to prove that the integral of an SG $G(\mathbf{v}; \mathbf{p}, \lambda)$ over the entire sphere is

$$\int_{\Omega} G(\mathbf{v}; \mathbf{p}, \lambda) d\mathbf{v} = \frac{2\pi}{\lambda} (1 - e^{-2\lambda}).$$

B. PRODUCT OF TWO SGS

Given two SGs $G(\mathbf{v}; \mathbf{p}_1, \lambda_1)$ and $G(\mathbf{v}; \mathbf{p}_2, \lambda_2)$, it is straightforward to prove that the product of them is still an SG, given by $G(\mathbf{v}; \mathbf{p}_3, \lambda_3)$. $G(\mathbf{v}; \mathbf{p}_2, \lambda_2) = G(\mathbf{v}; \mathbf{p}_3, \lambda_3, c_3)$, where

$$\lambda_3 = \|\lambda_1 \mathbf{p}_1 + \lambda_2 \mathbf{p}_2\|, \mathbf{p}_3 = \frac{\lambda_1 \mathbf{p}_1 + \lambda_2 \mathbf{p}_2}{\lambda_3}, \quad c_3 = e^{\lambda_3 - (\lambda_1 + \lambda_2)}.$$

C. PRODUCT INTEGRAL OF TWO SGS

Given two SGs $G(\mathbf{v}; \mathbf{p}_1, \lambda_1)$ and $G(\mathbf{v}; \mathbf{p}_2, \lambda_2)$, it is straightforward to prove that the product integral of them is

$$\begin{aligned} \int_{\Omega} G_1(\mathbf{v}) \cdot G_2(\mathbf{v}) d\mathbf{v} &= \int_{\Omega} G_3(\mathbf{v}) d\mathbf{v} \\ &= \frac{2\pi e^{\lambda_3 - (\lambda_1 + \lambda_2)}}{\lambda_3} (1 - e^{-2\lambda_3}), \end{aligned}$$

where $\lambda_3 = \|\lambda_1 \mathbf{p}_1 + \lambda_2 \mathbf{p}_2\| = \sqrt{\lambda_1^2 + \lambda_2^2 + 2\lambda_1 \lambda_2 (\mathbf{p}_1 \cdot \mathbf{p}_2)}$, which can be approximated by a first-order Taylor expansion of $\mathbf{p}_1 \cdot \mathbf{p}_2$

$$\lambda_3 \approx (\lambda_1 + \lambda_2) - \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} (1 - \mathbf{p}_1 \cdot \mathbf{p}_2).$$

Hence, the preceding product integral can be approximated as

$$\begin{aligned} \int_{\Omega} G_3(\mathbf{v}) d\mathbf{v} &\approx \frac{2\pi}{\lambda_3} (1 - e^{-2\lambda_3}) \exp\left(\frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} (\mathbf{p}_1 \cdot \mathbf{p}_2 - 1)\right) \\ &\approx \frac{2\pi}{\lambda_3} \exp\left(\frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} (\mathbf{p}_1 \cdot \mathbf{p}_2 - 1)\right). \end{aligned}$$

The term $e^{-2\lambda_3}$ is usually very small (since λ_3 is relatively large compared to λ_1 and λ_2), and hence it can be safely omitted in the previous equation. The preceding result can also be written as $\frac{2\pi}{\lambda_3} G(\mathbf{p}_1; \mathbf{p}_2, \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2})$, showing that if treating \mathbf{p}_1 (or \mathbf{p}_2) as an independent variable, the product integral result can still be approximated as an SG of \mathbf{p}_1 (or \mathbf{p}_2). The accuracy of this approximation depends on how large the sharpness values (λ_1 and λ_2) are. It is valid in most cases, and will produce large error only when the sharpness values of both SGs are small (e.g., a diffuse BRDF with a wide blue sky). Such cases can be avoided by restricting the sharpness of the SG light. We have done a quantitative analysis to measure the error of this approximation. The maximal absolute error is 0.017 and 6.6×10^{-4} when $\lambda_2 = 10$ and 50, respectively (no matter how large λ_1 is). Note that this approximation is also derived in a concurrent work by Iwasaki et al. [2012a].

D. PRODUCT INTEGRAL OF AN SG AND A SMOOTH FUNCTION

Given an SG $G(\mathbf{v}; \mathbf{p}, \lambda)$ and another spherical function $S(\mathbf{v})$, if $S(\mathbf{v})$ is very smooth, we can approximate the product integral by pulling $S(\mathbf{v})$ out of the integral, and approximating $S(\mathbf{v})$ using the value at

the SG center $S(\mathbf{p})$ [Wang et al. 2009a]

$$\int_{\Omega} G(\mathbf{v}) S(\mathbf{v}) d\mathbf{v} \approx S(\mathbf{p}) \int_{\Omega} G(\mathbf{v}) d\mathbf{v} = \frac{2\pi S(\mathbf{p})}{\lambda} (1 - e^{-2\lambda}).$$

This approximation works well in our experiments. If higher accuracy is desired, we can instead use a piecewise linear approximation of the smooth function [Xu et al. 2011].

E. APPROXIMATING A SPHERICAL REGION AS AN SG

Given a spherical region Ω_N , and a binary function $V_{\Omega_N}(\mathbf{v})$ whose value is 1 when $\mathbf{v} \in \Omega_N$ and 0 elsewhere, we can approximate the binary function as an SG: $V_{\Omega_N}(\mathbf{r}) \approx G(\mathbf{v}; \mathbf{p}_N, \lambda_N, c_N)$, where the center direction \mathbf{p}_N is directly set to be the center of the spherical region Ω_N , and the sharpness λ_N and coefficient c_N are obtained by preserving function energy and variance. Note that when approximating the spherical region Ω_N as a spherical disk, the energy (solid angle) and variance of the binary function $V_{\Omega_N}(\mathbf{v})$ should be $\|\Omega_N\|$ and $\|\Omega_N\|^2 / (2\pi)$, respectively. For SGs, it is also known that

$$\int_{\Omega} G_N(\mathbf{v}) d\mathbf{v} \approx 2\pi c_N / \lambda_N, \quad \int_{\Omega} G_N(\mathbf{v}) \cdot (\mathbf{v} - \mathbf{p}_N)^2 d\mathbf{v} \approx 4\pi c_N / \lambda_N^2.$$

Solving the aforesaid two equations, we get $\lambda_N \approx 4\pi / \|\Omega_N\|$, $c_N \approx 2$.

F. SOLID ANGLE AND CENTRAL DIRECTION OF A SPHERICAL TRIANGLE

As proved in Van Oosterom and Strackee [1983], the solid angle $\|\Omega_T\|$ of a spherical triangle Ω_T can be calculated as $\|\Omega_T\| = 2 \cdot \arctan(N, M)$, where

$$N = \mathbf{p}_1 \cdot (\mathbf{p}_2 \times \mathbf{p}_3), \quad M = 1 + \mathbf{p}_1 \cdot \mathbf{p}_2 + \mathbf{p}_2 \cdot \mathbf{p}_3 + \mathbf{p}_3 \cdot \mathbf{p}_1,$$

where \mathbf{p}_i ($1 \leq i \leq 3$) are three unit directions from the sphere center to the three vertices of the spherical triangle; $(\mathbf{p}_2 \times \mathbf{p}_3)$ denotes the cross-product of two vectors. The center of the spherical triangle \mathbf{p}_T is approximated as the average of these three directions: $\mathbf{p}_T \approx (\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3) / \|\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3\|$.

Hence, using the conclusion of Appendix E, the binary mask occupied by the spherical triangle can also be approximated as $V_{\Omega_T}(\mathbf{v}) \approx G(\mathbf{v}; \mathbf{p}_T, \lambda_T, c_T)$, where $\lambda_T = 4\pi / \|\Omega_T\|$, $c_T = 2$.

G. REPRESENTATIVE DIRECTION OF AN SG OVER A SPHERICAL TRIANGLE

Given an SG $G(\mathbf{v}; \mathbf{p}_1, \lambda_1)$, and a spherical triangle Ω_T , we usually need to determine a representative direction \mathbf{p}'_1 , which may be used as a point sample for querying constant values of other smooth functions. By approximating the spherical triangle as an SG $G(\mathbf{v}; \mathbf{p}_T, \lambda_T, c_T)$ (using the conclusions of Appendix E and F), the representative direction \mathbf{p}'_1 is set to be the center of the product SG of $G(\mathbf{v}; \mathbf{p}_1, \lambda_1)$ and $G(\mathbf{v}; \mathbf{p}_T, \lambda_T, c_T)$, which is

$$\mathbf{p}'_1 = (\lambda_1 \mathbf{p}_1 + \lambda_T \mathbf{p}_T) / \|\lambda_1 \mathbf{p}_1 + \lambda_T \mathbf{p}_T\|.$$

H. PROOF OF HOW θ_M IS DEFINED

As shown in Figure 17, we place a plane perpendicular to polar direction \mathbf{p} , with unit distance to sphere origin O . Denote the intersection point of polar direction \mathbf{p} to the plane as P' , and the projections of spherical points B , C , and M to the plane as B' , C' and M' , respectively. Denote the azimuthal angle of

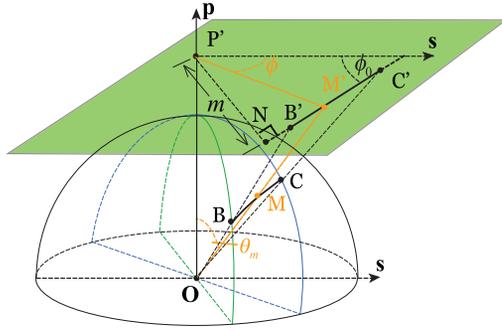


Fig. 17. Illustration for calculating θ_m .

point M' as ϕ . Draw a line $P'N$ perpendicular to line $B'C'$. Denote the angle between line $B'C'$ and the horizontal direction s as ϕ_0 , and the length of line segment $P'N$ as m . It is easy to see that $|P'M'| = |P'N|/\sin(\angle P'M'B') = m/\sin(\phi + \phi_0)$. So $\tan \theta_m = |P'M'|/|P'O| = |P'N|/\sin(\phi + \phi_0)$, and hence $\cos \theta_m = 1/\sqrt{\tan^2 \theta_m + 1} = \sin(\phi + \phi_0)/\sqrt{m^2 + \sin^2(\phi + \phi_0)}$. Note that both m and ϕ_0 are geometric properties that only depend on the positions of B and C , making them easy to compute.

I. FINDING KNOTS FOR THE 1D FUNCTION

$$F_{M,\lambda}(\phi)$$

By observing the overall shape of the function, we notice that it can be divided into 5 segments, including a smoothly ascending, a sharply ascending, a relatively flat, a sharply descending, and a smoothly descending segment. Hence, independent of the integration range $[\phi_1, \phi_2]$, we always find 4 knots in the range $\phi \in [0, \pi]$ to partition the function into 5 initial segments. Specifically, we pick two knots with value $\eta \cdot f_{max}$ (note that since the function is symmetric, there are two knots with the same value), and the other two with value $(1 - \eta) \cdot f_{max}$. Here $f_{max} = f(\phi = \pi/2)$ is the maximum value of the function, and η is an empirically determined threshold, and we typically set $\eta = 0.05$.

J. LOWER AND UPPER BOUNDS FOR SOLID ANGLES

For an infinitesimal area A , denote its size as $\Delta(A)$, its distance to the receiver point as $d(A)$, and the angle between its normal direction and the direction from the receiver point to A as $\theta_d(A)$, the solid angle subtended by A to a receiver point is $\Delta(A) \cos \theta_d(A)/d(A)^2$. Now, consider a node N that contains a set of triangles, the solid angle can be evaluated as an integral over all the area occupied by node N : $\|\Omega_N\| = \int_N \Delta(A) \cos \theta_d(A)/d(A)^2 dA$. It is easy to obtain the following inequality

$$\begin{aligned} \|\Omega_N\| &\geq \int_N \frac{\Delta(A) \cos \theta_d^{max}}{d_{max}^2} dA = \frac{\cos \theta_d^{max}}{d_{max}^2} \int_N \Delta(A) dA \\ &= \frac{\Delta_N \cdot \cos \theta_d^{max}}{d_{max}^2}, \end{aligned}$$

where θ_d^{max} is the largest possible value of $\theta_d(A)$, and d_{max} is the largest possible distance from the node to the receiver point. The upper bound of the solid angle can be obtained in a similar way.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments.

REFERENCES

- Aner Ben-Artzi, Kevin Egan, Fredo Durand, and Ravi Ramamoorthi. 2008. A precomputed polynomial representation for interactive BRDF editing with global illumination. *ACM Trans. Graph.* 27, 2, 13:1–13:13.
- Min Chen and James Arvo. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph.* 19, 4, 246–278.
- Ewen Cheslack-Postava, Rui Wang, Oskar Akerlund, and Fabio Pellacini. 2008. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph.* 27, 5, 128:1–128:10.
- Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and El-Mar Eisemann. 2011. Interactive indirect illumination using voxel cone tracing. *Comput. Graph. Forum* 30, 7, 1921–1930.
- Carsten Dachsbacher and Marc Stamminger. 2005. Reflective shadow maps. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'05)*. 203–231.
- Carsten Dachsbacher and Marc Stamminger. 2006. Splating indirect illumination. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'06)*. 93–100.
- Tomas Davidovic, Jaroslav Krivanek, Milos Hasan, Philipp Slusallek, and Kavita Bala. 2010. Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph.* 29, 6, 143:1–143:8.
- Charles De Rousiers, Adrien Bousseau, Kartic Subr, Nicolas Holzschuch, and Ravi Ramamoorthi. 2012. Real-time rendering of rough refraction. *IEEE Trans. Vis. Comput. Graph.* 18, 10, 1591–1602.
- William Donnelly and Andrew Lauritzen. 2006. Variance shadow maps. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'06)*. 161–165.
- Bartosz Fabianowski and John Dingliana. 2009. Interactive global photon mapping. *Comput. Graph. Forum* 28, 4, 1151–1159.
- Vclav Gassenbauer, Jaroslav Krivanek, and Kadi Bouatouch. 2009. Spatial directional radiance caching. *Comput. Graph. Forum* 28, 4, 1189–1198.
- Iliyan Georgiev, Jaroslav Krivanek, Tomas Davidovic, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6.
- Toshiya Hachisuka and Henrik Wann Jensen. 2010. Parallel progressive photon mapping on gpus. In *Siggraph Asia Sketches*. 54:1–54:1.
- Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph.* 31, 6.
- Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. Frequency domain normal map filtering. *ACM Trans. Graph.* 26, 3.
- Pat Hanrahan, David Salzman, and Larry Aupperle. 1991. A rapid hierarchical radiosity algorithm. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'91)*. 197–206.
- Milos Hasan, Jaroslav Krivanek, Bruce Walter, and Kavita Bala. 2009. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph.* 28, 5, 143:1–143:6.
- Milos Hasan, Fabio Pellacini, and Kavita Bala. 2007. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.* 26, 3, 26:1–26:10.
- Matthias Hollander, Tobias Ritschel, Elmar Eisemann, and Tamy Boubekeur. 2011. ManyLoDs: Parallel many-view level-of-detail selection for real-time global illumination. *Comput. Graph. Forum* 30, 4, 1233–1240.
- Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. 2012a. Interactive bi-scale editing of highly glossy materials. *ACM Trans. Graph.* 31, 6.
- Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. 2007. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. 35–44.

- Kei Iwasaki, Wataru Furuya, Yoshinori Dobashi, and Tomoyuki Nishita. 2012b. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. *Comput. Graph. Forum* 31, 727–734.
- Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.* 31, 4, 58:1–58:13.
- Henrik Wann Jensen. 2001. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd.
- James T. Kajiya. 1986. The rendering equation. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'86)*. 143–150.
- Leif Kobbelt. 2000. Root 3-subdivision. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*. 103–112.
- Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'97)*. 49–56.
- Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance caching for efficient global illumination computation. *IEEE Trans. Vis. Comput. Graph.* 11, 5, 550–561.
- Jurgen Laurijssen, Rui Wang, Phillip Dutre, and Benedict J. Brown. 2010. Fast estimation and rendering of indirect highlights. *Comput. Graph. Forum* 29, 4, 1305–1313.
- Bradford J. Loos, Lakulish Antani, Kenny Mitchell, Derek Nowrouzezahrai, Wojciech Jarosz, and Peter-Pike Sloan. 2011. Modular radiance transfer. *ACM Trans. Graph.* 30, 6, 178:1–178:10.
- Morgan McGuire and David Luebke. 2009. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the Conference on High Performance Graphics*. 77–89.
- Don Mitchell and Pat Hanrahan. 1992. Illumination from curved reflectors. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'92)*. 283–291.
- Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.* 22, 3, 376–381.
- Eyal Ofek and Ari Rappoport. 1998. Interactive reflections on curved objects. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'98)*. 333–342.
- Adriaan Van Oosterom and Jan Strackee. 1983. The solid angle of a plane triangle. *IEEE Trans. Bio-Med. Engin.* 30, 2, 125–126.
- Jiawei Ou and Fabio Pellacini. 2011. LightSlice: Matrix slice sampling for the many-lights problem. *ACM Trans. Graph.* 30, 6, 179:1–179:8.
- Minghao Pan, Rui Wang, Xinguo Liu, Qunsheng Peng, and Hujun Bao. 2007. Precomputed radiance transfer field for rendering interreflections in dynamic scenes. *Comput. Graph. Forum* 26, 3, 485–493.
- Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David Mcallister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A general purpose ray tracing engine. *ACM Trans. Graph.* 29, 4, 66:1–66:13.
- Timothy J. Purcell, Craig Donner, Mike Cammarano, Henrik Wann Jensen, and Pat Hanrahan. 2003. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware (HWWS'03)*. 41–50.
- Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global illumination with radiance regression functions. *ACM Trans. Graph.* 32, 4.
- Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph.* 28, 5, 132:1–132:8.
- Tobias Ritschel, Thorsten Grosch, Carsten Dachsbacher, and Jan Kautz. 2012. State of the art in interactive global illumination. *Comput. Graph. Forum* 31, 1, 160–188.
- Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.* 27, 5, 129:1–129:8.
- David Roger and Nicolas Holzschuch. 2006. Accurate specular reflections in real-time. *Comput. Graph. Forum* 25, 3, 293–302.
- Musawir A. Shah, Jaakko Kontinen, and Sumanta Pattanaik. 2007. Caustics mapping: An image-space technique for real-time caustics. *IEEE Trans. Vis. Comp.Graph.* 13, 2, 272–280.
- Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.* 21, 3, 527–536.
- Xin Sun, Kun Zhou, Yanyun Chen, Stephen Lin, Jiaoying Shi, and Baining Guo. 2007. Interactive relighting with dynamic brdfs. *ACM Trans. Graph.* 26, 3, 27:1–27:10.
- Yu-Ting Tsai and Zen-Chung Shih. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph.* 25, 3, 967–976.
- Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. 2006. Multidimensional lightcuts. *ACM Trans. Graph.* 25, 3, 1081–1088.
- Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A scalable approach to illumination. *ACM Trans. Graph.* 24, 3, 1098–1107.
- Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional lightcuts. *ACM Trans. Graph.* 31, 4.
- Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. 2009. Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph.* 28, 3, 92:1–92:8.
- Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. 2009a. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph.* 28, 5, 133:1–133:10.
- Rui Wang, Minghao Pan, Weifeng Chen, Zhong Ren, Kun Zhou, Wei Hua, and Hujun Bao. 2013. Analytic double product integrals for all-frequency relighting. *IEEE Trans. Vis. Comput. Graph.* 19, 7, 1133–1142.
- Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. 2009b. An efficient gpu-based approach for interactive global illumination. *ACM Trans. Graph.* 28, 3, 91:1–91:8.
- Chris Wyman and Scott Davis. 2006. Interactive image-space techniques for approximating caustics. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'06)*. 153–160.
- Kun Xu, Yun-Tao Jia, Hongbo Fu, Shi-Min Hu, and Chiew-Lan Tai. 2008. Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE Trans. Vis. Comput. Graph.* 14, 2, 454–467.
- Kun Xu, Li-Qian Ma, Bo Ren, Rui Wang, and Shi-Min Hu. 2011. Interactive hair rendering and appearance editing under environment lighting. *ACM Trans. Graph.* 30, 6, 173:1–173:10.
- Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic Spherical Gaussians. *ACM Trans. Graph.* 32, 6, 209:1–209:11.
- Ling-Qi Yan, Yahan Zhou, Kun Xu, and Rui Wang. 2012. Accurate translucent material rendering under spherical gaussian lights. *Comput. Graph. Forum* 31, 7, 2267–2276.
- Yubo Zhang, Zhao Dong, and Kwan-Liu Ma. 2013. Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE Trans. Vis. Comput. Graph.* 19, 8, 1317–1330.

Received October 2012; revised September 2013; accepted September 2013