A Practical Algorithm for Rendering All-Frequency Interreflections

KUN XU and YAN-PEI CAO and LI-QIAN MA
TNList, Department of Computer Science & Technology, Tsinghua University, Beijing
ZHAO DONG
Cornell University
RUI WANG
University of Massachusetts
and
SHI-MIN HU
TNList, Department of Computer Science & Technology, Tsinghua University, Beijing

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) and global photon maps 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 and caustics photon maps. In this paper, we present a practical algorithm for rendering interreflection effects at all frequency scales. Our method builds upon a Spherical Gaussian representation of the BRDF. Our main contribution is a novel mathematical development of the interreflection equation. This allows us to efficiently compute the one-bounce interreflection from a triangle to a shading point through an analytic formula combined with a piecewise linear approximation. We show through evaluation that this method is accurate for a wide range of BRDF parameters. Our second contribution is a hierarchical integration method to handle a large number of triangles with bounded error. Finally, we have implemented the algorithm on the GPU, achieving near-interactive rendering speed for a variety of scenes.

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

General Terms: Rendering

Additional Key Words and Phrases: Interreflections, Global Illumination, Spherical Gaussian, GPU

ACM Reference Format:

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

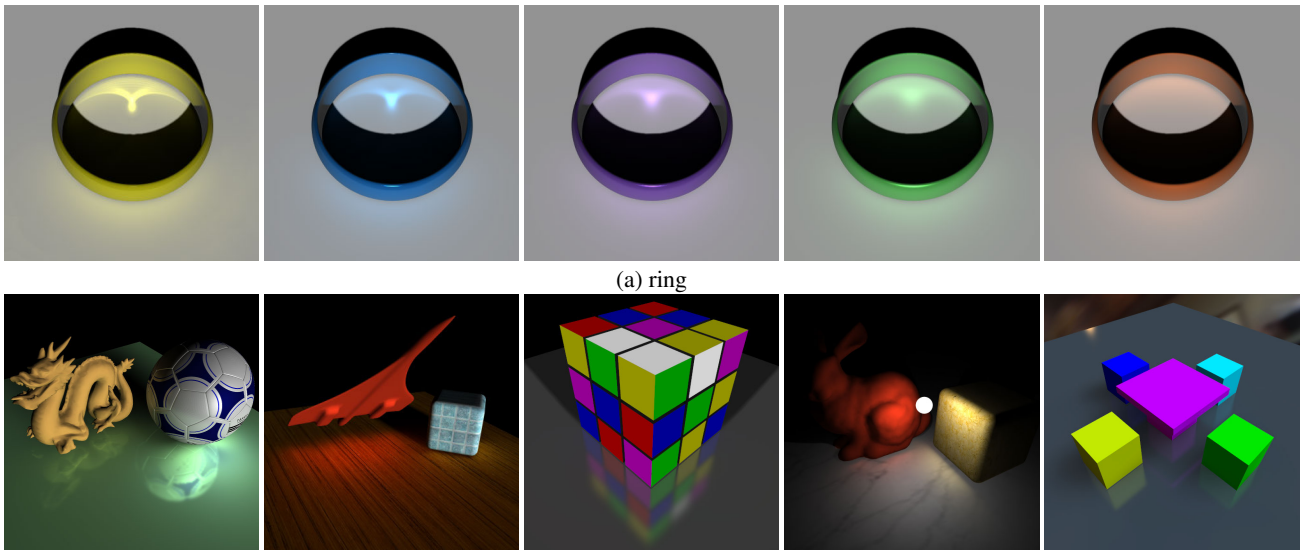
1. INTRODUCTION

Accurate rendering of interreflection (or indirect illumination) effects has been a long-standing challenge in computer graphics research, particularly when the materials can vary across a variety of different types, 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 specific range of materials. For example, methods based on Virtual Point Lights [Keller 1997] and global photon maps [Jensen 2001] 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. This works well with diffuse materials, due to their low-frequency and smooth filtering nature. However, for highly-glossy materials, these methods require a significantly larger number of discrete points, reducing the computation performance and increasing the storage size.

On the other hand, methods based on path tracing [Kajiya 1986] and caustics photon maps [Jensen 2001] are efficient for highly-glossy and specular materials, but the efficiency drops significantly when the materials become diffuse or semi-glossy. This is mainly because these methods stochastically trace light rays upon reflections or refractions from the materials. Therefore highly-glossy materials lead to lower variance in the computation, reducing the rendering noise; conversely, nearly diffuse materials lead to high variance and consequently increased rendering noise.

These limitations are fundamentally 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. This is a common issue in rendering research. Consequently, a scene that consists of mixed materials (i.e. at different frequency scales) often requires special care and the combination of several algorithms, each of which deals with a separate frequency range. This results in increased algorithm complexity, and makes it difficult to ensure all individual algorithms produce consistent results.

In this paper, we present a practical algorithm for rendering interreflection effects at all frequency scales. Our method builds upon a Spherical Gaussian (SG) representation of the BRDF [Wang et al. 2009]. By changing the support size of the Spherical Gaussian, this representation can faithfully reproduce BRDFs at a wide range of frequency (i.e. glossiness) levels. Our main contribution is a novel mathematical development of the interreflection equation. Specifi-



(a) ring (b) dragon (c) airplane (d) magic cube (e) bunny (f) table

Fig. 1: *Our algorithm achieves near-interactive rendering speed 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 interreflection effects, such as indirect highlight (b), diffuse reflection (c), glossy reflection (d), and 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.*

cally, by representing the BRDF and lighting both using SGs, we derive an analytic formula for the one-bounce interreflection from a triangle to a shading point, and accurately estimate the result using a piecewise linear approximation. We show through evaluation that this method performs well for a wide range of BRDF parameters, thanks to the analytic derivation.

In practical applications, however, we need to consider scenes that consist of more than a few triangles. To improve the efficiency of our algorithm, we present a hierarchical integration method to handle a large number of triangles. The hierarchical integration is computed with bounded error. Finally, we have implemented the algorithm on the GPU, achieving near-interactive rendering speed for a variety of scenes. Figure 1 shows several examples.

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 paper. We refer readers to [Ritschel et al. 2012] for a comprehensive survey. This section covers the most relevant work to ours. 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). Instant radiosity [Keller 1997] is a classic VPL-based technique. It creates VPLs by tracing paths from the primary lights, and uses shadow map algorithms to estimate the total illumination contribution from all VPLs to a shading point. The step of estimating the contributions from all VPLs is commonly known as *final gathering*. To achieve high-quality result, a large number of VPLs are usually necessary. Therefore a challenge is how to accelerate the final gathering. Lightcuts [Walter et al. 2005; Walter et al. 2006] constructs a hierarchical structure of VPLs to compute final gathering at sublinear cost. Row-column sampling [Hašan et al. 2007] and LightSlice [Ou and Pellacini 2011] 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 a different method for each component to account for detailed glossy interreflections. Recently, Walter et al. [2012] propose *bidirectional lightcuts* to reduce the bias in VPL-based rendering by introducing *virtual sensor points* on eye paths. While accurate, these VPL-based methods perform in offline speed, taking minutes or hours to run. In addition, highly glossy receivers typically pose a big challenge as they require a large number of VPLs.

Photon Mapping. 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 non-specular receivers using photon density estimation. By exploiting the GPU, photon mapping can achieve interactive performance [Purcell et al. 2003; Wang et al. 2009; Fabianowski and Dingliana 2009; McGuire and Luebke 2009; Hachisuka and Jensen 2010], including both caustics and indirect illumination. However, these methods usually require the reflectors to be either diffuse or specular. Semi-glossy reflectors lead to significantly increased computation cost.

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 a suitable basis set to exploit the low-dimensional structure of the matrices. PRT has been extended to achieve interreflection effects with dynamic BRDFs [Sun et al.

2007; Ben-Artzi et al. 2008; Cheslack-Postava et al. 2008] under the static scene assumption. Interreflections in dynamic scenes have also been studied [Iwasaki et al. 2007; Pan et al. 2007], but limited to low-frequency effects.

Interactive GI. Work on interactive global illumination (GI) has a rich history. Dachsbacher and Stamminger [2005] introduced *reflective shadow maps* (RSM) where pixels in the shadow map are considered indirect light sources. This method gathers low-resolution indirect lighting from the RSM and obtains high resolution result using screen-space interpolation. While interactive, it is limited to diffuse reflectors and ignores indirect shadows. Later, Dachsbacher and Stamminger [2006] presented a method to include non-diffuse reflectors by splatting the radiance contribution from each pixel. However, it only supports diffuse receivers. Ritschel et al. [2008] presented *imperfect shadow maps* to approximate indirect visibility for VPLs and achieved interactive GI using a small number of VPLs. It is limited to low-frequency reflection 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 using hierarchical point rasterization into a micro-buffer. The micro-buffer can be warped to account for BRDF importance. As a result it supports receivers with glossy BRDFs. However, the reflector is restricted to diffuse or low-frequency BRDFs. Laurijssen et al. [2010] proposed a method for interactively rendering 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 is limited to low-frequency effects. Finally, there are many image-based methods for efficiently rendering caustics [Wyman and Davis 2006; Shah et al. 2007], but these techniques are aimed 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 for generating the 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 glossy materials.

Spherical Gaussians (SGs). Spherical Gaussians (SGs) provide flexible support sizes and closed-formed 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. 2009]. Wang et al. [2009] approximate the normal distribution of a microfacet BRDF using sums of SGs. They demonstrated that such approximation is accurate for representing a wide range of parametric and measured BRDFs. They make use of this property to achieve real-time all-frequency rendering of dynamic BRDFs under the static scene assumption. However, this method only considers direct illumination, while our method focuses on interreflections.

3. BACKGROUND

In this section, we review the necessary background of spherical Gaussians (SGs) representations and BRDF approximations.

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

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

where unit vector $\mathbf{p}$, λ, and c represent the center direction, bandwidth, and the scalar coefficient of the SG. 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 are explained in details in the Appendix.

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

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

where $\mathbf{i}$, $\mathbf{o}$ are the incoming and outgoing directions; k_d, k_s are the diffuse and specular coefficients. We approximate the specular component (also called specular lobe) ρ_s using a single SG:

$$\rho_s(\mathbf{i},\mathbf{o}) \approx G(\mathbf{i};2(\mathbf{o}\cdot\mathbf{n})\mathbf{n}-\mathbf{o},\lambda_s)$$

where $\mathbf{n}$ is the surface normal, and λ_s is the bandwidth which controls the specular lobe size. Note that the diffuse component k_d can be treated as a special SG with a zero bandwidth: $k_d = G(\mathbf{i};2(\mathbf{o}\cdot\mathbf{n})\mathbf{n}-\mathbf{o},0,k_d)$. Therefore the BRDF defined in Eq. 2 can be re-written as the sum of two SGs:

$$\rho(\mathbf{i},\mathbf{o}) \approx \sum_{j=0}^{1} G\left(\mathbf{i};\mathbf{o}^j,\lambda^j,c^j\right) \tag{3}$$

where $\lambda^0 = 0$, $c^0 = k_d$, $\lambda^1 = \lambda_s$, $c^1 = k_s$, $\mathbf{o}^0 = \mathbf{o}^1 = 2(\mathbf{o}\cdot\mathbf{n})\mathbf{n}-\mathbf{o}$. Note that the above SG approximation is a simplified version of the model introduced by Wang et al. [2009]. As shown in their work, the specular component of commonly used parametric BRDFs, such as the Blinn-Phong and the Cook-Torrance models, can be accurately approximated by one single SG. Even for more sophisticated BRDFs, such as measured ones, a small number of SGs usually suffice to achieve highly accurate approximations.

4. ONE-BOUNCE INTERREFLECTION MODEL

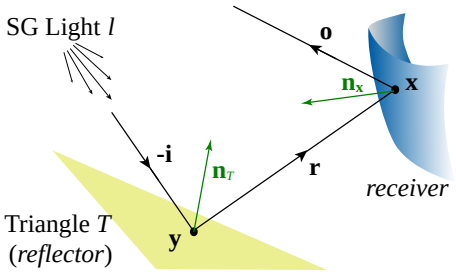


Fig. 2: *Light path of one-bounce interreflection.*

We start by deriving the one-bounce interreflection model for a single triangle reflector, as shown in Figure 2. Assuming a distant incident SG light l with emitted radiance $G(\mathbf{i};\mathbf{i}_l,\lambda_l)$ ($G_l(\mathbf{i})$ for short), given a triangle T with normal $\mathbf{n}_T$ (referred to as the *reflector*) and a shading point x with normal $\mathbf{n}_\mathbf{x}$ (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 (The incorporation of visibility will be explained later in Section 6). We also ignore texture data on the reflector, assuming that the reflector has a uniform BRDF. The incorporation of texture will be explained in Section 5. The one-bounce outgoing radiance from x towards $\mathbf{o}$ can then be computed as an integration over a spherical triangle:

$$L_{\mathrm{x}}(\mathbf{o}) = \int_{\Omega_T} L(\mathbf{r}) \rho_{\mathrm{x}}(-\mathbf{r}, \mathbf{o}) \max(-\mathbf{r} \cdot \mathbf{n}_{\mathrm{x}}, 0)\, \mathrm{d}\mathbf{r} \tag{4}$$

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 $\mathbf{n}_{\mathrm{x}}$ are the BRDF and normal direction at the receiver point x; $L(\mathbf{r})$ is the reflected radiance from y to x, defined as:

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

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 $L(\mathbf{r})$

In this subsection, we will explain how to evaluate the reflected radiance $L(\mathbf{r})$ defined in Eq. 5. First, we represent the BRDF ρ_T of the triangle T as a sum of SGs (as shown in Eq. 3): $\rho_T(\mathbf{i}, \mathbf{r}) \approx \sum_{j=0}^{1} G\left(\mathbf{i}; \mathbf{r}_T^j, \lambda_T^j, c_T^j\right)$. For denotation simplicity, in the following derivation, we omit the summation $\sum_{j=0}^{1}(\cdot)$ 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:

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

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

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

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. As shown in Appendix C, the integral of the product SG can be well approximated by a single SG:

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

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:

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

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

$$L(\mathbf{r}) \approx F(\mathbf{r}) G(\mathbf{r}; \mathbf{i}_R, \lambda_R) \tag{7}$$

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 $L(\mathbf{r})$ is evaluated as a linear sum of the product of a smooth function and an SG.

4.2 Evaluating the Interreflection Radiance $L_{\mathrm{x}}(\mathbf{o})$

Now we explain how to evaluate the one-bounce outgoing radiance $L_{\mathrm{x}}(\mathbf{o})$ 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): $\rho_{\mathrm{x}}(-\mathbf{r}, \mathbf{o}) \approx \sum_{j=0}^{1} G\left(\mathbf{r}; -\mathbf{o}_{\mathrm{x}}^j, \lambda_{\mathrm{x}}^j, c_{\mathrm{x}}^j\right)$. For denotation simplicity, we again omit the summation $\sum_{j=0}^{1}(\cdot)$ over index j below, and substitute the reflected radiance $L(\mathbf{r})$ (Eq. 7) into Eq. 4:

$$L_{\mathrm{x}}(\mathbf{o}) \approx \int_{\Omega_T} H(\mathbf{r}) G(\mathbf{r}; \mathbf{i}_R, \lambda_R) G(\mathbf{r}; -\mathbf{o}_{\mathrm{x}}, \lambda_{\mathrm{x}}, c_{\mathrm{x}})\, \mathrm{d}\mathbf{r}$$

where $H(\mathbf{r}) = F(\mathbf{r}) \cdot \max(-\mathbf{r} \cdot \mathbf{n}_{\mathrm{x}}, 0)$ is again a smooth function. Since the product of two SGs is still an SG (see Appendix B), the above Equation can be rewritten as:

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

where $G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h)$ is the product of the two SGs in the above Equation. (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 above equation as:

$$L_{\mathrm{x}}(\mathbf{o}) \approx H(\mathbf{r}'_h) \int_{\Omega_T} G(\mathbf{r}; \mathbf{r}_h, \lambda_h, c_h)\, \mathrm{d}\mathbf{r} \tag{8}$$

The representative direction $\mathbf{r}'_h$ 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 support. We adopt such an interpolation method because the optimal choice of the representative direction varies in different scenarios. For example, if the spherical triangle is much larger than the support of SG (e.g. SG just spans a small area inside the spherical triangle), the best choice of the representative direction is the SG center; Conversely, if the support of SG is much larger than the spherical triangle, the best choice is the triangle direction. The formula of $\mathbf{r}'_h$ is described in Appendix G.

The remaining question in Eq. 8 is how to evaluate the integral of an SG over a spherical triangle subtended by the planar triangle T. However, this integral does not have a closed-form solution. Fortunately, as shown in the following subsection, it can be reduced to a 1D integral, which can then be 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 $\int_{\Omega_T} G(\mathbf{v}; \mathbf{p}, \lambda)\, \mathrm{d}\mathbf{v}$. 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 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:

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

The plus/minus sign may vary depending on relative positions of the zenith point P and spherical triangle $\Omega_{\triangle ABC}$ (e.g. when P is inside $\Omega_{\triangle ABC}$, $\Omega_{\triangle ABC} = \Omega_{\triangle PBC} + \Omega_{\triangle PAB} + \Omega_{\triangle PCA}$). Without loss

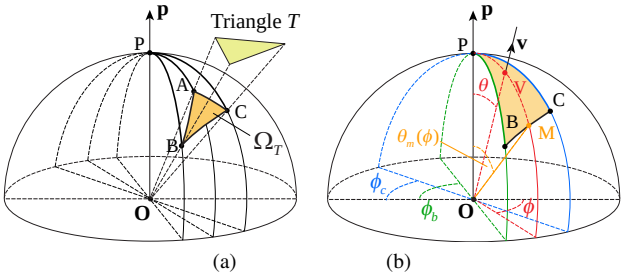


Fig. 3: *Integrating an SG over a spherical triangle Ω_T.*

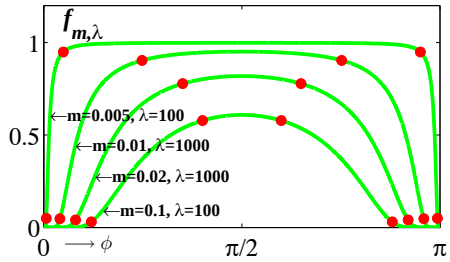


Fig. 4: *Plot of the 1D function $f_{m,\lambda}(\phi)$ and its non-uniform knots (i.e. sample points) at different parameter settings.*

of generality, let us consider how to evaluate the integral over spherical triangle $\triangle PBC$: $\int_{\Omega_{\triangle PBC}} G(\mathbf{v};\mathbf{p},\lambda)\,\mathrm{d}\mathbf{v}$. Using 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), 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 $\overset{\frown}{PB}$ and $\overset{\frown}{PC}$ are longitude arcs, we can rewrite the integral in spherical coordinates by integrating the azimuthal angle from ϕ_c to ϕ_b:

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

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 $\overset{\frown}{BC}$ and the longitude arc $\overset{\frown}{PV}$. It is easy to find out that the inner integral of polar angle θ has an analytic solution, so that the above equation can be rewritten as:

$$\int_{\Omega_{\triangle PBC}} G(\mathbf{v})\,\mathrm{d}\mathbf{v} = \int_{\phi_c}^{\phi_b}\left(-\frac{1}{\lambda}e^{\lambda(\cos\theta-1)}\Big|_0^{\theta_m(\phi)}\right)\mathrm{d}\phi$$
$$= \frac{1}{\lambda}\int_{\phi_c}^{\phi_b}\left(1-e^{\lambda(\cos\theta_m(\phi)-1)}\right)\mathrm{d}\phi$$

Through some further derivations using basic 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):

$$\cos\theta_m(\phi) = \sin(\phi+\phi_0)/\sqrt{m^2+\sin^2(\phi+\phi_0)} \tag{9}$$

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:

$$\int_{\Omega_{\triangle PBC}} G(\mathbf{v})\,\mathrm{d}\mathbf{v} = \frac{\phi_b-\phi_c}{\lambda} - \frac{1}{\lambda}\int_{\phi_c}^{\phi_b} e^{\lambda\left(\frac{\sin(\phi+\phi_0)}{\sqrt{m^2+\sin^2(\phi+\phi_0)}}-1\right)}\mathrm{d}\phi$$
$$= \frac{\phi_b-\phi_c}{\lambda} - \frac{1}{\lambda}\int_{\phi_1}^{\phi_2} f_{m,\lambda}(\phi)\,\mathrm{d}\phi \tag{10}$$

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

$$f_{m,\lambda}(\phi) = \exp\left[\lambda\left(\frac{\sin\phi}{\sqrt{m^2+\sin^2\phi}}-1\right)\right] \tag{11}$$

Thus we have simplified the original integral to a 1D integral through analytic development. Now, $\int f_{m,\lambda}(\phi)\,\mathrm{d}\phi$ does not have an analytic solution, so we need to evaluate it numerically. A straightforward solution is to precompute a 3D table of pre-integrated values, with respect to the three parameters (m,λ,ϕ). However, we found that the value of this function in fact changes rapidly when parameter m is very small, and thus using a precomputed table with finite resolutions can lead to severe artifacts. To address this issue, we introduce below a non-uniform piecewise linear approximation which works very well for accurate evaluation of $\int f_{m,\lambda}(\phi)\,\mathrm{d}\phi$.

4.4 Non-uniform Piecewise Linear Approximation

Figure 4 shows the plots of the 1D function $f_{m,\lambda}(\phi)$ under several different parameter settings. As shown in this figure, the function is defined in $\phi\in[0,\pi]$, is symmetric around $\phi=\pi/2$, and increases monotonically in the range $\phi\in[0,\pi/2]$. For some parameters m and λ, the function changes smoothly. However, for other parameters, particularly when $m\approx 0$, the function changes very sharply. We can approximate the function using piecewise linear approximation, but since the point where the sharp change happens varies, it is unsuitable to use a uniform sampling method. Instead, we propose a non-uniform sampling method as detailed below.

First, observing the overall shape of the function (Figure 4), we find 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 four knots (sample points) in the range $[0,\pi]$ to partition the function into 5 initial segments. Specifically, we pick two knots that have value $\eta\cdot f_{max}$ (note that since the function is symmetric, there are two points with the same value), and the other two that have value $(1-\eta)\cdot f_{max}$. Here $f_{max}=f(\phi=\pi/2)$ is the maximum value of the function, and η is a threshold value that we typically set to $\eta=0.05$. This is an empirically determined value that has worked well in practice. As shown in Figure 4, the knots are highlighted in red color. The knots capture the shape (structure) of the function well for different parameters of m and λ. 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 lies in the range $[\phi_1,\phi_2]$, we split it into two intervals $[\phi_1,\phi_k]$ and $[\phi_k,\phi_2]$, where ϕ_k is the knot position. Finally, we approximate the function $f_{m,\lambda}(\phi)$ in each interval using a uniformly sampled piecewise linear function with K partitions. In our implementation we set $K=3$. Due to the way the knots are selected, the function within each interval is very smooth, hence this method works quite well. Evaluation is presented in Section 7.2.

Summary. In this section, we derived the analytic formula for computing one-bounce interreflection from a triangle (Eq. 4) to a shading point due to a distance SG light. Furthermore, we demonstrated that this formula can be efficiently evaluated using a non-uniform piecewise linear approximation of a 1D function (Eq. 11).

5. HIERARCHICAL STRUCTURE

In the previous section, we derived an analytical formula for computing the interreflection from a single reflector triangle to a shading point. For simple scenes we can apply the formula to directly sum up the contribution from each triangle. 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], the Lightcuts [Walter et al. 2005] and the micro-rendering technique [Ritschel et al. 2009], we also propose a hierarchical method to efficiently sum up the contribution 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 to $[-1,1]$) and $\mathbf{n}$ is the triangle normal. m is a scalar weights that controls the relative importance of I and $\mathbf{n}$, and we usually set it to $m = 5$. Next, the splitting is computed using the principal direction rule. Specifically, at every 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 non-textured scene, each node stores the average center and normal of its triangles, as well as a bounding box, a normal cone, and the total triangle area. See Figure 5(a) for illustration. To deal with textured scene, each node additionally stores the average and the largest/smallest texture color values of all the textures mapped on its triangles. The average texture color value is used to modulate the radiance contribution of the node, while the largest/smallest texture color values are used to estimate the error bound. These stored terms are used later to estimate the reflected radiance from the node to a receiver point, and the associated error bound.

During the rendering stage, we employ a similar strategy as in the Lightcuts [Walter et al. 2005] to efficiently evaluate the one-bounce interreflections. For each receiver point, we start from an initial *cut* that contains only the root node of the binary tree, then iteratively refine it. At each iteration, we pick the node with the highest estimated error and replace it by its two child nodes. The iteration stops either if the largest estimated error 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*. The number of nodes in the cut is referred to as *cut size*. Figure 5(b) shows an illustration. Note that when reaching the leaf node 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 non-textured scenes since our derivations in

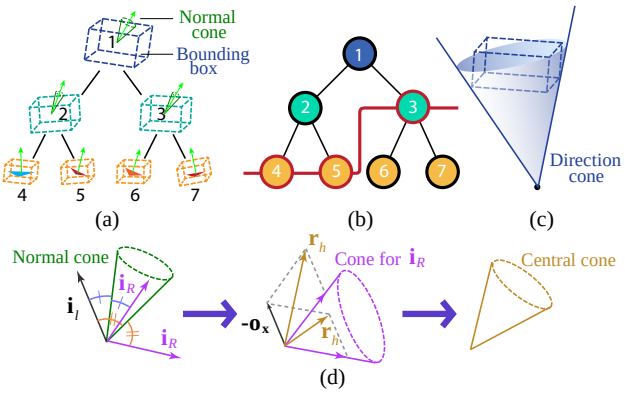


Fig. 5: *Hierarchical structure and error bound estimation. (a) a binary tree of triangles showing the bounding box and normal cone stored at each node; (b) an example of reflector cut; (c) direction cone; (d) computing central cone from the normal cone.*

Section 4 assumes that the reflector triangle has a uniform BRDF across it without texture variations. Hence, when dealing with leaf node with a textured triangle, we still compute its error bound using its stored texture info. If the estimated 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.

Next, we will explain in detail how to evaluate the estimated one-bounce radiance reflected by a node (in Section 5.1) and the associated error bound (in 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 $G(\mathbf{i};\mathbf{p}_l,\lambda_l)$ from node N. To begin, we denote the center position of node N as I_N, its average triangle normal as $\mathbf{n}_N$, and its triangle area as Δ_N. Then, inspired by the micro-rendering technique [Ritschel et al. 2009], the node N is treated as an "implicit" plane with center position I_N, normal direction $\mathbf{n}_N$, area Δ_N, and average texture color $\bar{t}_N$. Hence, we can reuse Eq. 8 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:

$$L_x(\mathbf{o}) \approx \bar{t}_N \cdot H(\mathbf{r}'_h) \int_{\Omega_N} G(\mathbf{r};\mathbf{r}_h,\lambda_h,c_h)\,\mathrm{d}\mathbf{r} \qquad (12)$$

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 :

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

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 V_{Ω_N} using an SG: $V_{\Omega_N}(\mathbf{r}) \approx G(\mathbf{r};\mathbf{r}_N,\lambda_N,c_N)$ ($G_N(\mathbf{r})$ 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 bandwidth and coefficient are determined by preserving the function energy and variance (refer to Appendix E): $\lambda_N = 4\pi/\|\Omega_N\|$, $c_N = 2$. Here $\|\Omega_N\|$ is the solid angle computed as:

$$\|\Omega_N\| \approx (\Delta_N \cdot \max(\mathbf{r}_N \cdot \mathbf{n}_N, 0))/d_N^2 \tag{13}$$

where d_N is the distance from the receiver point x to node center I_N.

Hence, the integral of the SG over Ω_N can be re-written as:

$$\int_{\Omega_N} G(\mathbf{r};\mathbf{r}_h,\lambda_h,c_h)\,\mathrm{d}\mathbf{r} \approx \int_{\Omega} G(\mathbf{r};\mathbf{r}_h,\lambda_h,c_h)\,G_N(\mathbf{r})\,\mathrm{d}\mathbf{r} \tag{14}$$

Substituting Eq. 14 into Eq. 12, the one-bounce interreflected radiance $L_x(\mathbf{o})$ can be easily computed since the product integral of two SGs has an analytic solution.

Summary. By approximating the binary function V_{Ω_N} using an SG, we can efficiently compute Eq. 12 using an analytic solution. An alternatively approach is to use a representative value of the binary function at each tree node, as in the Lightcuts method. We employ the SG approximation because it leads to an analytic solution and is thus less prone to numerical sampling artifacts. However, we note that this approximation will produce large errors in the case of integrating an SG with small support size over nodes with large solid angles. Thus we need to estimate the error bound of this approximation to ensure the accuracy. If the estimated error of the current node is larger than a predefined threshold, we replace it by its two child nodes thus reducing the approximation error.

5.2 Estimating the Error Bound

Given the bounding box and the normal cone of a node, we aim to evaluate the error bound of computing $L_x(\mathbf{o})$ using Eq. 12 and Eq. 14. Denote 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}. It is obvious that the error can be conservatively bounded by $(t_{max}\cdot g_{max}\cdot\|\Omega\|_{max} - t_{min}\cdot g_{min}\cdot\|\Omega\|_{min})$. The bounds on texture color values $[t_{min}, t_{max}]$ are stored within each node. In following, we explain how to estimate the bounds on the solid angles and the SG function values.

Bounds on the solid angle. Based on Eq. 13, we can compute the bounds on the solid angle $\|\Omega_N\|$ by estimating the bound of the distance d_N and that 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 the distance d_N from the receiver point x to the node. We denote the bounds as $d_{\min}$ and $d_{\max}$. 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 the two cones. We denote the lower and upper bounds of the angle as $\theta_d^{\min}$ and $\theta_d^{\max}$, respectively. Hence, the lower and upper bounds of the solid angle $\|\Omega_N\|$ can be computed as: $\|\Omega\|_{min} = \Delta_N \cdot \cos\theta_d^{\max}/d_{\max}^2$, $\|\Omega\|_{max} = \Delta_N \cdot \cos\theta_d^{\min}/d_{\min}^2$.

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 $\mathbf{r}_h$ (in Eq. 8): $\mathbf{r}_h$ is calculated from another direction $\mathbf{i}_R$, while $\mathbf{i}_R$ is computed using the normal $\mathbf{n}_N$. 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 $\theta_r^{\min}$ and $\theta_r^{\max}$, 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 DETAILS

The implementation of our algorithm consists of an initialization step which loads the scene and builds the binary tree structure, and a run-time rendering step which implements the one-bounce interreflection algorithm.

Algorithmic Pipeline. 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 for that model since it is more efficient to iterate through each triangle of the model during rendering. Then, for each tree node, we evaluate its visibility, the details of which are provided below. This initialization step only takes a few seconds and does not need to be performed again unless if a model deforms.

During the run-time rendering step, we first evaluate the direct illumination in a separate pass. To compute interreflections, we need to determine a reflector cut (see 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 their cut merging algorithm. This cut merging algorithm makes use of the per-vertex cut stored at the nearby vertices around a pixel. Finally, the interreflection is computed for each shading pixel using the merged reflector cut in a pixel shader.

Lights. Our method supports all-frequency incident lighting by nature, because Spherical Gaussian has varying support size. Following [Wang et al. 2009], 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. 2009], 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: $L(\mathbf{i}) \approx G\left(\mathbf{i};(\mathbf{l}-\mathbf{y})/\|\mathbf{l}-\mathbf{y}\|, 4\|\mathbf{l}-\mathbf{y}\|^2/r^2, 2c\right)$.

Visibility. For direct illumination, we adopt the variance shadow maps (VSM) [Donnelly and Lauritzen 2006] to generate shadows for each SG light. For indirect illumination, 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 sample 16 points uniformly on the reflector triangle and query into the VSM of the SG light to retrieve the shadow values. The average shadow value of the sample points is stored as the direct visibility for each reflector

triangle. To compute indirect visibility, we adopt a variant of imperfect shadow maps (ISM) [Ritschel et al. 2008]. Specifically, during scene loading step, we select 200 random sample points on each model to represent the virtual light positions to capture the ISM at run-time. For each node in the binary tree of the model, its closest three sampled virtual lights are stored. Then, the center position of the node is projected onto the triangle plane spanned by the node's three samples, and the weight of each sampled light is computed as the barycentric coordinate of the projected position. During the run-time rendering stage, the ISM with a resolution 128×128 is generated for each sampled virtual light. Next, during the evaluation of indirect illumination, after the reflector cut is determined, the visibility for each tree node is interpolated by the shadow values from the ISMs, using the three closest sampled virtual lights and the associated weights. The visibility approximation will be evaluated in Section 7.2.

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, we manage several (in our implementation, 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 from the first queue if it is non-empty, otherwise pick 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 positions, normals, BRDFs, 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 non-linear piecewise linear approximation (in Section 4) for reflector triangles is used for leaf nodes, while the node approximation (Eq. 14) is used for interior nodes. The contribution of each node is modulated by the direct and indirect visibility described earlier. While direct visibility is obtained as a pass-in parameter of each node, indirect visibility is obtained by interpolating from the query shadow values using the ISMs.

7. RESULTS AND COMPARISONS

This section presents our results. Performance data is reported on a PC with Intel Xeon 2.27G CPU and 8 GB 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 1% for all examples. The performance data is reported in Table I. For validation, we examine the accuracy of our mathematical model, as well as compare our results to the VPL-based method [Keller 1997], photon mapping [Jensen 2001], and the micro-rendering method [Ritschel et al. 2009].

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

Table I. : *Performance of the results shown in this paper. From left to right, we give the name of the scene, the number of faces, average cut size, overall fps, final shading time and and cut selection time.*

7.1 Results

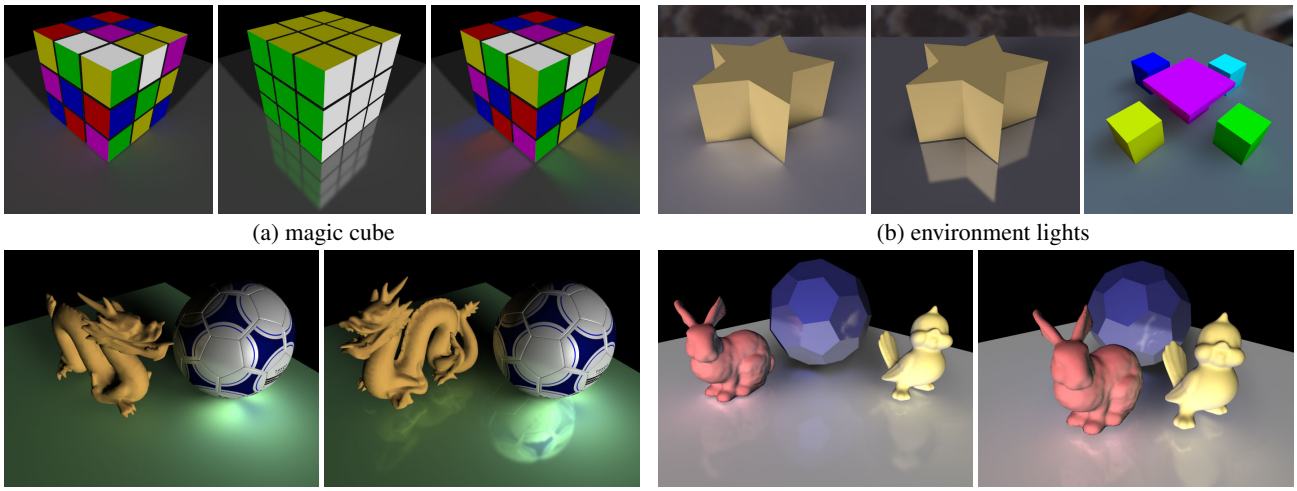
Simple Scenes. We first verify our method using simple scenes containing a couple hundred of triangles. Figure 6(a) shows results of a magic cube scene with varying BRDFs on the cube and on the plane. In Figure 6(b), we show the results of a star scene and a kitchen scene under different environment lights represented by 10 SGs. Note that in these examples, we 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.

Complex 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 effects on the plane for all examples. Further, our method achieves smooth and coherent transitions (e.g. without flickering) while changing BRDF parameters (see the supplemental video). Thus it is a single algorithm that can reproduce all-frequency reflection effects.

Figure 1(b,e) and Figure 6(b-d) demonstrate our method under different types of lights, including a local light (Figure 1(e)), SG lights with small support (Figure 1(c)) and large support (Figure 6(c,d), as well as environment lights (Figure 6 (b)).

Textured Scenes. Figure 7 shows results of complex textured scenes. Note that our method performs well for such scenes with 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 seconds. Clearly, the bottleneck lies in the per-pixel final shading stage, especially for complex scenes. Considering the sponza scene in Figure 7(b), the final shading stage takes 81% of the time on average, while other steps, including the cut selection, ISMs generation and direct lighting, take a total of 19% of the overall cost. This is because the average cut size increases with the number the faces. In the final shading step, which is implemented in fragment shader, for each pixel we need to loop all the nodes in its cut to compute the final shading. However, such a shading computation adapts with the fast evolving GPU and can be greatly accelerated by next-gen many-core hardware.



(a) magic cube (b) environment lights

(c) dragon (d) bunny & tweety

Fig. 6: *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.*

(a) kitchen (b) sponza

Fig. 7: *Rendering results of complex textured scenes.*

7.2 Comparisons

Accuracy of our model. In Figure 8, we show results of our one-bounce interreflection model using piecewise linear approximation with different number of piecewise intervals (in Section 4.4). We further compare our results with results generated by a VPL-based method [Keller 1997] and the path traced reference. The test scene consists of a single equilateral triangle placed perpendicular to the receiver plane, and a directional light at a 45 degree angle to the ground plane. The ground (receiver) plane has a Lambertain BRDF and we only show the interreflection (i.e. no direct illumination) from the triangle to the plane in this figure.

In Figure 8, 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 under all tested BRDFs. In contrast, the VPL-based method (column (d)) with 256 VPLs works well for only 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 must be significantly increased for highly-glossy BRDFs.

Accuracy of visibility approximation. Recall that we randomly sample some (typically 200) virtual lights (VLs) on each model and generates one 128×128 ISM for each VL. Visibility between the receiver point and each node/triangle is computed by interpolating the queried shadow values from three nearest VLs. Hence, the number of VLs affects the accuracy of the indirect visibility. In Figure 9, we evaluate the indirect visibility approximation used in our method. Notice that our result with 200 VLs (Figure 9 (c))

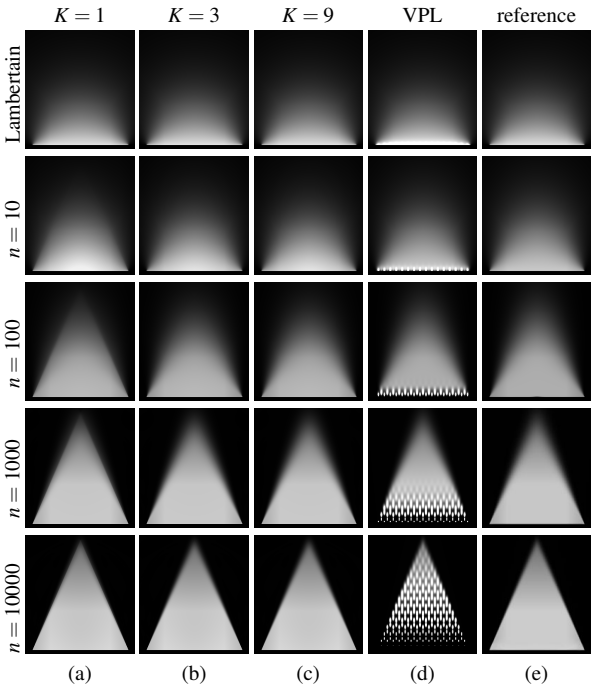


Fig. 8: *Comparisons of one-bounce interreflection with varying BRDFs. In the top row, the reflector triangle is diffuse; from row 2 to row 5, the reflector triangle has a Blinn-Phong BRDF with glossy shininess of $n = 10, 100, 1000, 10000$ respectively. Columns (a), (b) and (c) show results generated 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.*

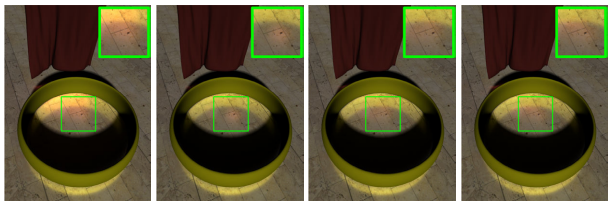


Fig. 9: *Comparison of visibility using different number of virtual lights. (a) gives a result without considering indirect visibility; (b),(c) gives our results with virtual light number as 100 and 200, respectively; (d) is the reference. Notice that, by setting virtual light number as 200, our result matches reference very well.*

matches reference image (Figure 9 (d)) very well. As demonstrated in [Ritschel et al. 2008], ISMs can handle low-frequency indirect shadows well but have difficulties with high-frequency shadows due to the low resolution of ISMs and low VL sampling rate. Therefore, our method inherits the same limitations and only handles low-frequency indirect visibility. However, we claim that handling indirect visibility is not our major contribution and we aims at supporting all-frequency BRDFs. The problem of handling high-frequency indirect visibility itself is an open research problem in rendering. When a better solution becomes available in the future, we can easily incorporate it into our implementation.

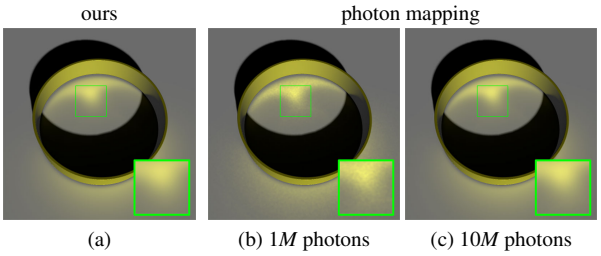


Fig. 10: *Comparison of caustics effects with photon mapping .*

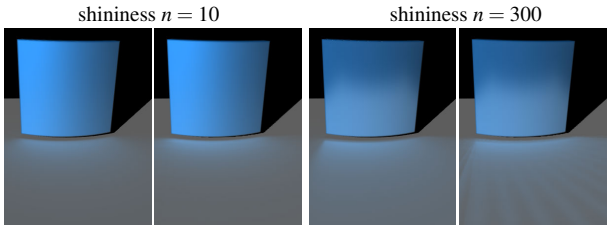


Fig. 11: *Comparison to micro-rendering [Ritschel et al. 2009].*

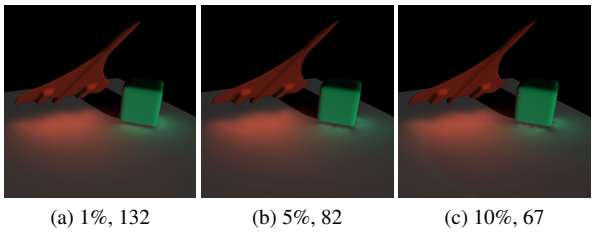


Fig. 12: *Comparison of our results using different error bound threshold. The threshold and average cut size are given in the caption of each subfigure.*

Comparison to photon mapping. In Figure 10, we compare our method with photon mapping [Jensen 2001] for generating caustics. The scene consists of a directional light, a reflective ring, which has a Blinn-Phong BRDF with shininess $n = 100$, and a Lambertian plane. Figure 10 (b) gives the result of photon mapping using $1M$ photons, which exhibits noticeable noises in the caustics area. The reason is that while photon mapping is highly efficient for rendering caustics from perfectly specular reflectors, it is not so efficient when the material becomes glossy, as shown in this example. This is mainly due to the stochastic sampling effects when photons are reflected from a glossy surface. As a result, glossy reflectors require a much large number of photons to achieve high quality. As shown in Figure 10(c), increasing the number of photons to $10M$ removes most of noise, but doing so also significantly reduces rendering performance. In contrast, our method handles the glossy reflector very efficiently as shown in Figure 10(a). We achieve the same quality as Figure 10(c) at near-interactive frame rates.

Comparison to micro-rendering. In Figure 11, we compare our method to micro-rendering [Ritschel et al. 2009]. The size of micro-buffer is set to be 24×24. The scene consists of a curved glossy plate placed on top of a Lambertian plane. In micro-rendering, importance sampling is performed in the final gathering step using BRDF importance, 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. The BRDF of the reflector node is ignored during cut selection, and hence it can ignore the importance of nodes with glossy BRDF, which are responsible for strong interreflections. In contrast, our cut selection scheme takes into account not only the subtended solid angle but also the BRDF of both receiver and reflector, and we use the estimated error bound to ensure accuracy. Moreover, our method estimates the contribution of each node using integration instead of a single point sample, leading to more robustness. As shown in Figure 11, 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 "speckle" artifacts. In contrast, our method still renders high-quality interreflections without noticeable artifacts.

Error bound threshold. We further evaluate our choice of the error bound threshold. Since smaller thresholds result in better quality but slower frame rates, a trade-off should be considered. Figure 12 shows the results using different thresholds, including 1%, 5% and 10%. Using a larger threshold such as 5% gives better performance (refer to Figure 12 (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 results in the demos are generated by setting the error bound threshold to 1%.

8. LIMITATIONS AND DISCUSSIONS

Multiple-bounce interreflections. One major limitation of our method is that it currently only supports *one-bounce* interreflections. However, our analytic 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. 14) 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 again be represented as an SG of $\mathbf{r}_2$. Hence, the outgoing radiance from the receiver point can once 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 sub-linear cost by employing a multi-dimensional reflector cut, similar to [Walter et al. 2006]. A complete study of this method to handle multiple-bounce interreflections remains our future work.

9. CONCLUSION AND FUTURE WORKS

To summarize, we have presented a practical algorithm for rendering one-bounce interreflections with all-frequency BRDFs. Our method builds upon a Spherical Gaussian representation of the BRDF 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 non-uniform piecewise linear approximation. To handle scenes containing a large number of triangles, we propose a hierarchical integration method with error bounds that take into account the BRDFs at both the reflector and receiver. We have presented evaluations using a wide frequency range of BRDFs, from purely diffuse to highly specular. 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), are all unified and supported in a single algorithm.

In future work, we would like to extend our method to handle multi-bounce interreflections. This may be accomplished by recursively applying our one-bounce algorithm and employing a multidimensional reflector cut as described in Section 8. We are also interested in extending our method to handle high dimensional textures, such as bidirectional texture functions. Another direction for future work is to improve the speed of our algorithm, such as by employing ManyLods [Hollander et al. 2011] to further exploit the spatial and temporal coherence in rendering.

REFERENCES

BEN-ARTZI, A., EGAN, K., DURAND, F., AND RAMAMOORTHI, R. 2008. A precomputed polynomial representation for interactive BRDF editing with global illumination. *ACM Transactions on Graphics 27,* 2, 13:1–13:13.

CHEN, M. AND ARVO, J. 2000. Theory and application of specular path perturbation. *ACM Transactions on Graphics 19,* 4, 246–278.

CHESLACK-POSTAVA, E., WANG, R., AKERLUND, O., AND PELLACINI, F. 2008. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Transactions on Graphics 27,* 5, 128:1–128:10.

DACHSBACHER, C. AND STAMMINGER, M. 2005. Reflective shadow maps. In *Proceedings of I3D*. 203–231.

DACHSBACHER, C. AND STAMMINGER, M. 2006. Splatting indirect illumination. In *Proceedings of I3D*. 93–100.

DAVIDOVIČ, T., KŘIVÁNEK, J., HAŠAN, M., SLUSALLEK, P., AND BALA, K. 2010. Combining global and local virtual lights for detailed glossy illumination. *ACM Transactions on Graphics 29,* 6, 143:1–143:8.

DONNELLY, W. AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of I3D*. 161–165.

FABIANOWSKI, B. AND DINGLIANA, J. 2009. Interactive global photon mapping. *Computer Graphics Forum 28,* 4, 1151–1159.

HACHISUKA, T. AND JENSEN, H. W. 2010. Parallel progressive photon mapping on gpus. In *Siggraph Asia Sketches*. 54:1–54:1.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A rapid hierarchical radiosity algorithm. In *Proceedings of SIGGRAPH*. 197–206.

HAŠAN, M., KŘIVÁNEK, J., WALTER, B., AND BALA, K. 2009. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics 28,* 5, 143:1–143:6.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics 26,* 3, 26:1–26:10.

HOLLANDER, M., RITSCHEL, T., EISEMANN, E., AND BOUBEKEUR, T. 2011. Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum 30,* 4, 1233–1240.

IWASAKI, K., DOBASHI, Y., YOSHIMOTO, F., AND NISHITA, T. 2007. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proceedings of EGSR*. 35–44.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd.

KAJIYA, J. T. 1986. The rendering equation. In *Proceedings of SIGGRAPH*. 143–150.

KELLER, A. 1997. Instant radiosity. In *Proceedings of SIGGRAPH*. 49–56.

KOBBELT, L. 2000. $\sqrt{3}-$subdivision. In *Proceedings of SIGGRAPH*. 103–112.

LAURIJSSEN, J., WANG, R., DUTRÉ, P., AND BROWN, B. 2010. Fast estimation and rendering of indirect highlights. *Computer Graphics Forum 29,* 4, 1305–1313.

LOOS, B. J., ANTANI, L., MITCHELL, K., NOWROUZEZAHRAI, D., JAROSZ, W., AND SLOAN, P.-P. 2011. Modular radiance transfer. *ACM Transactions on Graphics 30,* 6, 178:1–178:10.

MCGUIRE, M. AND LUEBKE, D. 2009. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of High Performance Graphics*. 77–89.

MITCHELL, D. AND HANRAHAN, P. 1992. Illumination from curved reflectors. In *Proceedings of SIGGRAPH*. 283–291.

OFEK, E. AND RAPPOPORT, A. 1998. Interactive reflections on curved objects. In *Proceedings of SIGGRAPH*. 333–342.

OOSTEROM, A. V. AND STRACKEE, J. 1983. The solid angle of a plane triangle. *IEEE Transactions on Bio-medical Engineering 30,* 2, 125–126.

OU, J. AND PELLACINI, F. 2011. Lightslice: matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics 30,* 6, 179:1–179:8.

PAN, M., WANG, R., LIU, X., PENG, Q., AND BAO, H. 2007. Precomputed radiance transfer field for rendering interreflections in dynamic scenes. *Computer Graphics Forum 26,* 3, 485–493.

PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In *Proceedings of Graphics Hardware*. 41–50.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics 28,* 5, 132:1–132:8.

RITSCHEL, T., GROSCH, T., DACHSBACHER, C., AND KAUTZ, J. 2012. State of the art in interactive global illumination. *Computer Graphics Forum 31*, to appear.

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics 27,* 5, 129:1–129:8.

ROGER, D. AND HOLZSCHUCH, N. 2006. Accurate specular reflections in real-time. *Computer Graphics Forum 25,* 3, 293–302.

SHAH, M., KONTTINEN, J., AND PATTANAIK, S. 2007. Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics 13,* 2, 272–280.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21,* 3, 527–536.

SUN, X., ZHOU, K., CHEN, Y., LIN, S., SHI, J., AND GUO, B. 2007. Interactive relighting with dynamic brdfs. *ACM Transactions on Graphics 26,* 3, 27:1–27:10.

TSAI, Y.-T. AND SHIH, Z.-C. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics 25,* 3, 967–976.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. *ACM Transactions on Graphics 25,* 3, 1081–1088.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics 24,* 3, 1098–1107.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Transactions on Graphics 31,* 4, 59:1–59:11.

WALTER, B., ZHAO, S., HOLZSCHUCH, N., AND BALA, K. 2009. Single scattering in refractive media with triangle mesh boundaries. *ACM Transactions on Graphics 28,* 3 (July), 92:1–92:8.

WANG, J., REN, P., GONG, M., SNYDER, J., AND GUO, B. 2009. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Transactions on Graphics 28,* 5, 133:1–133:10.

WANG, R., WANG, R., ZHOU, K., PAN, M., AND BAO, H. 2009. An efficient gpu-based approach for interactive global illumination. *ACM Transactions on Graphics 28,* 3, 91:1–91:8.

WYMAN, C. AND DAVIS, S. 2006. Interactive image-space techniques for approximating caustics. In *Proceedings of I3D*. 153–160.

XU, K., MA, L.-Q., REN, B., WANG, R., AND HU, S.-M. 2011. Interactive hair rendering and appearance editing under environment lighting. *ACM Transactions on Graphics 30,* 6, 173:1–173:10.

APPENDIX

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

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

B. Product of two SGs. Defining 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 two SGs is still an SG, given by $G(\mathbf{v};\mathbf{p}_1,\lambda_1)\cdot G(\mathbf{v};\mathbf{p}_2,\lambda_2) = G(\mathbf{v};\mathbf{p}_3,\lambda_3,c_3)$, where:

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

C. Dot product of two SGs. Defining 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 dot product of the two SGs is (using properties A & B):

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

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 above dot product can be approximated as:

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

The term $e^{-2\lambda_3}$ is usually very small (since λ_3 is relatively large), and hence it can be safely omitted in above equation. The above result can also be written as $\frac{2\pi}{\lambda_3}G\left(\mathbf{p}_1;\mathbf{p}_2,\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}\right)$, showing that if treating $\mathbf{p}_1$ (or $\mathbf{p}_2$) as an independent variable, the result of dot product can be still approximated as an SG of $\mathbf{p}_1$ (or $\mathbf{p}_2$). The accuracy of the dot product approximation depends on how large the bandwidth λ_3 is. It is valid in most cases (error $< 1\%$ when $\lambda_3 > 2.3$). This approximation will produce large error only when both SGs

are very large (e.g. a diffuse BRDF with a wide blue sky). Such cases can be avoided by restricting the bandwidth of the SG light.

D. Dot product 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 dot product 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. 2009; Xu et al. 2011]:

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

This approximation works well in our experience. For better accuracy, piecewise linear approximation of the smooth function [Xu et al. 2011] can be alternatively used.

E. Approximating a spherical region as an SG. Given a spherical region Ω_N, and a binary function $V_{\Omega_N}(\mathbf{v})$ that valued 1 when $\mathbf{v}\in N$ and valued 0 elsewise. Approximating the binary function as an SG: $V_{\Omega_N}(\mathbf{r}) \approx G(\mathbf{v};\mathbf{p}_N,\lambda_N,c_N)$, the central direction $\mathbf{p}_N$ is directly set to be the center of the spherical region Ω_N, and the bandwidth λ_N and coefficient c_N are obtained by preserving function energy and variance. Obviously, when approximating the spherical region N as a spherical disk, the energy (solid angle) and variance of the binary function $V_{\Omega_N}(\mathbf{v})$ are Ω_N and $\|\Omega_N\|^2/(2\pi)$, respectively. For SGs, it is also known that:

$$\int G_N(\mathbf{v})\,\mathrm{d}\mathbf{v} \approx 2\pi c_N/\lambda_N,\ \int G_N(\mathbf{v})\cdot(\mathbf{v}-\mathbf{p}_N)^2\,\mathrm{d}\mathbf{v} \approx 4\pi c_N/\lambda_N^2$$

Solving the above 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 [Oosterom and Strackee 1983], the solid angle $\|\Omega_T\|$ of a spherical triangle Ω_T is given by $\|\Omega_T\| = 2\cdot\arctan(N,M)$,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 $\mathbf{p}_i$ $(1\le i\le 3)$ are three unit directions from the sphere origin to the three vertices of the spherical triangle; $(\mathbf{p}_2\times\mathbf{p}_3)$ denotes the cross product of the two vectors. The center of the spherical triangle $\mathbf{p}_T$ is approximated as the average of these three directions: $\mathbf{p}_T \approx (\mathbf{p}_1+\mathbf{p}_2+\mathbf{p}_3)/\|\mathbf{p}_1+\mathbf{p}_2+\mathbf{p}_3\|$.

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 for querying constant values of other smooth functions. Approximating the spherical triangle as an SG $G(\mathbf{v};\mathbf{p}_T,\lambda_T,c_T)$ (in Appendix E), 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 Eq. 9. As shown in Figure 13, 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 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 $\mathbf{s}$ as ϕ_0, and the length of line segment $P'N$ as m. It is trivial that $|P'M'| = |P'N|/\sin(\angle P'M'B') = m/\sin(\phi+\phi_0)$. So that $\tan\theta_m = |P'M'|/|P'O| = |P'M'| = m/\sin(\phi+\phi_0)$, 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

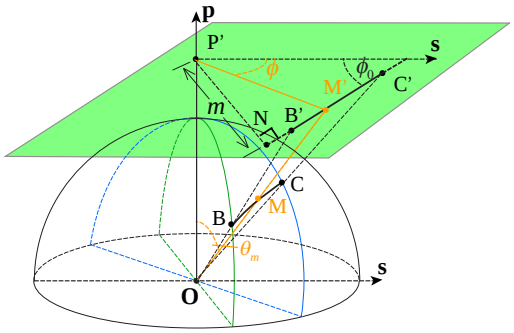


Fig. 13: *Illustration for calculating* θ_m.

both of m and ϕ_0 are geometric properties only depended on the positions of B and C, making them easy for computation.