DOI: 10.1111/j.1467-8659.2012.03052.x
EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Real-time Rendering of Dynamic Scenes under All-frequency Lighting using Integral Spherical Gaussian

Kei Iwasaki[1] Wataru Furuya[1] Yoshinori Dobashi[2] Tomoyuki Nishita[3]

[1] Wakayama University [2] Hokkaido University [3] The University of Tokyo
[1] iwasaki@sys.wakayama-u.ac.jp [2]doba@ime.ist.hokudai.ac.jp [3] nis@is.s.u-tokyo.ac.jp

Figure 1: *Real-time rendering of fully dynamic scenes with all-frequency lighting and highly specular BRDFs.*

Abstract

We propose an efficient rendering method for dynamic scenes under all-frequency environmental lighting. To render the surfaces of objects illuminated by distant environmental lighting, the triple product of the lighting, the visibility function and the BRDF is integrated at each shading point on the surfaces. Our method represents the environmental lighting and the BRDF with a linear combination of spherical Gaussians, replacing the integral of the triple product with the sum of the integrals of spherical Gaussians over the visible region of the hemisphere. We propose a new form of spherical Gaussian, the integral spherical Gaussian, that enables the fast and accurate integration of spherical Gaussians with various sharpness over the visible region on the hemisphere. The integral spherical Gaussian simplifies the integration to a sum of four pre-integrated values, which are easily evaluated on-the-fly. With a combination of a set of spheres to approximate object geometries and the integral spherical Gaussian, our method can render object surfaces very efficiently. Our GPU implementation demonstrates real-time rendering of dynamic scenes with dynamic viewpoints, lighting, and BRDFs.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Color, shading, shadowing, and texture

1. Introduction

Real-time rendering of fully dynamic scenes illuminated by environmental lighting, where the viewpoint, lighting, and BRDFs can be changed and objects can move and deform at run-time, is beneficial in many applications such as lighting/material design, animation, and games. For photorealistic rendering under environmental lighting, the triple product of the environmental lighting, the BRDF, and the visibility function is integrated. Integrating the triple product is computationally expensive, especially for dynamic scenes with highly-specular BRDFs, since the visibility function is high-frequency and BRDFs have sharp peaks; moreover, both of these vary at run-time due to the dynamic viewpoint, changing BRDFs and objects. To achieve real-time performance, previous methods tackled this problem by restricting the methods to static scenes [SKS02, NRH03, TS06, GKD07, WRG*09], low-frequency lighting [RWS*06, NKF09], or moderately specular reflectance [ADM*08, RGK*08]. Real-time rendering of fully dynamic scenes under all-frequency lighting still remains a challenging task.

To address this, we propose an efficient rendering method for dynamic scenes illuminated by all-frequency environmental lighting. Our method uses spherical Gaussians (SGs) that are identical to isotropic Gaussians [GKD07]. SGs are also a type of spherical radial basis functions (SRBFs) [TS06]. For diffuse BRDFs, our method represents the environmental lighting with a small number of SGs. For specular BRDFs, our method uses prefiltered environment maps with SGs of different sharpness, which makes it possible to deal with dynamic, highly-specular BRDFs. Both calculations require the integral of the product of the SGs and the visibility, which makes the rendering difficult and limits the previous method [WRG*09] to static scenes. We propose a novel calculation method, *integral spherical Gaussian (ISG)*, for integrating SGs with arbitrary sharpness over the visible region on the hemisphere at each shading point. SGs with various sharpness are pre-integrated over different integration intervals of the elevation and azimuthal angles. The integrated values are fitted to a simple function of the elevation angle, azimuthal angle, and the sharpness of the SG, enabling direct evaluation on the GPU. Therefore, the integral spherical Gaussians do not need huge precomputed data nor precomputational time, which are required for previous methods [NRH03, TS06]. The integral of the SG over the visible region on the hemisphere is efficiently calculated using the pre-integrated SG and a set of spheres approximating the object geometries. Furthermore, our method utilizes a hierarchical representation of spheres for scalability and level of detail. All the computations are performed in parallel on the GPU.

Our method contributes the following:

- Real-time rendering of fully dynamic scenes under all-frequency lighting with highly specular BRDFs.
- An efficient calculation method to integrate various SGs over the visible region on the hemisphere using the integral spherical Gaussian, which is easy to implement and therefore amenable to GPUs.
- No huge amount of precomputed data for the visibility function nor the BRDFs is needed, which allows the user to change the object geometries and BRDFs at run-time.

2. Previous Work

Precomputation based methods: Sloan et al. proposed a Precomputed Radiance Transfer (PRT) method to render static scenes illuminated by low-frequency lighting environments [SKS02]. Several methods have been proposed to render all-frequency shadows using Haar wavelets [NRH03, NRH04, WTL06] or spherical radial basis functions [TS06]. To render highly specular materials, Green et al. proposed a real-time rendering method by approximating BRDFs with isotropic Gaussian functions [GKD07]. Sun et al. developed an interactive relighting method for dynamic BRDFs with indirect illumination [SZC*07]. Wang et al. proposed a real-time rendering method for dynamic, spatially-varying BRDFs [WRG*09]. Lam et al. proposed a multiscale spherical radial basis function that represents all-frequency lighting, and achieved real-time rendering with its GPU-implementation [LHLW10]. Although these methods can render scenes illuminated by environmental lighting in real-time, they are limited to static scenes.

Kautz et al. proposed an interactive rendering method for self-shadows of dynamic objects using hemispherical rasterization [KLA04]. Sloan et al. proposed a local deformable PRT method using zonal harmonics [SLS05]. Ren et al. proposed a real-time rendering method for dynamic scenes under low-frequency lighting [RWS*06]. Sloan et al. extended this method to calculate the indirect illumination for dynamic scenes [SGNS07]. These two methods approximate the object shape with a set of spheres and calculate the occlusion due to the set in the Spherical Harmonics (SH) domain using a SH exponentiation technique. Nowrouzezahrai et al. presented a real-time shadowing method with arbitrary BRDFs for animated models [NKF09]. These methods, however, cannot be applied to an all-frequency lighting environment.

Zhou et al. proposed shadow fields that record shadowing effects due to a rigid object around the object and achieved real-time rendering of dynamic scenes for low-frequency lighting and interactive rendering for all-frequency lighting [ZHL*05]. Tamura et al. extended precomputed shadow fields to record shadowing effects adaptively and achieved real-time rendering on the GPU [TJCN06]. Sun et al. proposed a generalized wavelet product integral method to render dynamic scenes of rigid objects under all-frequency lighting [SM06]. Xu et al. proposed a real-time rendering method of dynamic scenes under all-frequency lighting using spherical piecewise constant functions [XJF*08]. These methods, however, cannot deal with deformable objects.

Shadow map based methods: Krivanek et al. proposed filtered importance sampling and rendered dynamic scenes using the approximate visibility calculated from shadow maps [KC08]. This method, however, utilizes a small number of shadow maps, which may cause incorrect visibilities. Annen et al. proposed a real-time rendering method for dynamic scenes under all-frequency lighting [ADM*08]. This method approximates the environmental lighting with a set of area light sources, and render shadows from these light sources using a convolution shadow map method. This method, however, does not integrate the product of the BRDF and the incident radiance. Ritschel et al. proposed imperfect shadow maps to calculate the indirect illumination for dynamic scenes [RGK*08]. They also demonstrated that these imperfect shadow maps can be applied to environment map lighting. Hollander et al. proposed the concept of 'ManyLoDs' to accelerate the many-view rasterization process [HREB11]. These methods determine the light positions to render shadow maps by using importance sampling of the environment maps, and ignore the importance

of the material BRDFs. Therefore, these methods are unsuitable for rendering highly specular materials.

3. Overview

Our method focuses on fast rendering of dynamic scenes with direct illumination. The outgoing radiance $L(\mathbf{x},\mathbf{\Omega}_o)$ at point $\mathbf{x}$ in the outgoing direction $\mathbf{\Omega}_o$ is calculated by the following equation.

$$L(\mathbf{x},\mathbf{\Omega}_o)=\int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)V(\mathbf{x},\mathbf{\Omega}_i)\ \ (\mathbf{x},\mathbf{\Omega}_i,\mathbf{\Omega}_o)\max(0,\mathbf{n}(\mathbf{x})\cdot\mathbf{\Omega}_i)d\mathbf{\Omega}_i, \tag{1}$$

where $\mathbb{S}^2$ denotes the unit sphere in $\mathbb{R}^3$, $\mathbf{\Omega}_i$ is the incident direction, $L(\mathbf{\Omega}_i)$ is the distant lighting represented by the environment maps, $V(\mathbf{x},\mathbf{\Omega}_i)$ is the visibility function at $\mathbf{x}$, is the BRDF, and $\mathbf{n}(\mathbf{x})$ is the normal at $\mathbf{x}$. To simplify the notation, we omit $\mathbf{x}$ in the following. Our method represents the BRDF with $(\mathbf{\Omega}_i,\mathbf{\Omega}_o)=k_d+k_s\ _s(\mathbf{\Omega}_i,\mathbf{\Omega}_o)$ where k_d is a diffuse term and $k_s\ _s(\mathbf{\Omega}_i,\mathbf{\Omega}_o)$ is a specular term. By substituting this in Eq. (1), $L(\mathbf{\Omega}_o)$ can be calculated from the sum of the diffuse component L_d and the specular component $L_s(\mathbf{\Omega}_o)$ as follows:

$$\begin{aligned} L(\mathbf{\Omega}_o) &= k_d L_d + k_s L_s(\mathbf{\Omega}_o), \\ L_d &= \int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)V(\mathbf{\Omega}_i)\max(0,\mathbf{\Omega}_i\cdot\mathbf{n})d\mathbf{\Omega}_i, \qquad (2)\\ L_s(\mathbf{\Omega}_o) &= \int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)V(\mathbf{\Omega}_i)\ _s(\mathbf{\Omega}_i,\mathbf{\Omega}_o)\max(0,\mathbf{\Omega}_i\cdot\mathbf{n})d\mathbf{\Omega}_i. (3)\end{aligned}$$

To calculate L_d, our method approximates $L(\mathbf{\Omega}_i)$ and the cosine term $\max(0,\mathbf{n}\cdot\mathbf{\Omega}_i)$ with the sum of spherical Gaussians. A spherical Gaussian (SG) $G(\mathbf{\Omega}_i;\ ,\ ,\mu)$ is represented by the following equation:

$$G(\mathbf{\Omega}_i;\ ,\ ,\mu)=\mu e^{\ (\mathbf{\Omega}_i\cdot\ -1)}, \tag{4}$$

where the unit vector is the lobe axis, is the lobe sharpness, and μ is the lobe amplitude that consists of RGB components. Our method represents the environmental lighting L as the sum of SGs; $L(\mathbf{\Omega}_i)\approx\sum_{k=1}^{K} G(\mathbf{\Omega}_i;\ _k,\ _k,\mu_k)$ where K is the number of SG lobes. The cosine term is also approximated by a single SG as $G(\mathbf{\Omega}_i;\mathbf{n},\ _c,\mu_c)$. By substituting these terms into Eq. (2), L_d is calculated by the following equation.

$$L_d\approx\sum_{k=1}^{K}\int_{\mathbb{S}^2} G(\mathbf{\Omega}_i;\ _k,\ _k,\mu_k)G(\mathbf{\Omega}_i;\mathbf{n},\ _c,\mu_c)V(\mathbf{\Omega}_i)d\mathbf{\Omega}_i. \tag{5}$$

The product of two SGs can be represented with a single SG:

$$G(\mathbf{\Omega}_i;\ ,\ ,\mu)=G(\mathbf{\Omega}_i;\ _k,\ _k,\mu_k)G(\mathbf{\Omega}_i;\mathbf{n},\ _c,\mu_c), \tag{6}$$

where the lobe sharpness is $\|\ _k\ _k+\ _c\mathbf{n}\|$, the lobe axis is $(\ _k\ _k+\ _c\mathbf{n})/\ $, and the amplitude μ is $\mu_k\mu_c e^{\ -\ _k-\ _c}$. The diffuse component L_d is calculated by integrating the product of the visibility function $V(\mathbf{\Omega}_i)$ and the spherical Gaussian $G(\mathbf{\Omega}_i;\ ,\ ,\mu)$. For fully dynamic scenes where the lighting and viewpoint change, and objects move and deform, $V(\mathbf{\Omega}_i)$ and $G(\mathbf{\Omega}_i;\ ,\ ,\mu)$ change and cannot be precomputed.

To calculate the visibility function and integrate the product at run-time, our method approximates the object geometry by a set of spheres. This makes it possible to calculate the solid angles of occluded regions for the SGs efficiently. To calculate the integral of the product of the visibility function and the SG, our method extends the summed area table (SAT) [Cro84] (or integral image [VJ01]) over the 2D image domain to the SG over the 2D domain on a unit sphere. We call this *integral spherical Gaussian* (ISG) (Section 4). The integral of the product of the visibility function and arbitrary SGs is efficiently calculated using the ISG.

To calculate the specular component L_s, our method represents the BRDF and the cosine term as the sum of SGs. As [WRG*09], the BRDF $(\mathbf{\Omega}_i,\mathbf{\Omega}_o)$ can be calculated from the sum of the SGs as $(\mathbf{\Omega}_i,\mathbf{\Omega}_o)\approx\sum_{j=1}^{J} G(\mathbf{\Omega}_i;\ _j,\ _j,\mu_j)$, where J is the number of SGs for , and the lobe axis $_j$ depends on the outgoing direction $\mathbf{\Omega}_o$. By substituting this into Eq. (3), $L_s(\mathbf{\Omega}_o)$ can be rewritten as:

$$L_s(\mathbf{\Omega}_o)\approx\sum_{j=1}^{J}\int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)G(\mathbf{\Omega}_i;\ _j,\ _j,\mu_j)G(\mathbf{\Omega}_i;\mathbf{n},\ _c,\mu_c)V(\mathbf{\Omega}_i)d\mathbf{\Omega}_i. \tag{7}$$

Since the product of two SGs is also represented with a single SG $G_j(\mathbf{\Omega}_i)=G(\mathbf{\Omega}_i;\ '_j,\ '_j,\mu'_j)$, $L_s(\mathbf{\Omega}_o)$ can be rewritten as:

$$L_s(\mathbf{\Omega}_o)\approx\sum_{j=1}^{J}\int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)G_j(\mathbf{\Omega}_i)V(\mathbf{\Omega}_i)d\mathbf{\Omega}_i. \tag{8}$$

Our method approximates the integral of the triple product as:

$$L_s(\mathbf{\Omega}_o)\approx\sum_{j=1}^{J}\int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)G_j(\mathbf{\Omega}_i)d\mathbf{\Omega}_i\frac{\int_{\mathbb{S}^2} G_j(\mathbf{\Omega}_i)V(\mathbf{\Omega}_i)d\mathbf{\Omega}_i}{\int_{\mathbb{S}^2} G_j(\mathbf{\Omega}_i)d\mathbf{\Omega}_i}. \tag{9}$$

The numerator $\int_{\mathbb{S}^2} G_j(\mathbf{\Omega}_i)V(\mathbf{\Omega}_i)d\mathbf{\Omega}_i$ can be calculated in the same way as L_d and the denominator $\int_{\mathbb{S}^2} G_j(\mathbf{\Omega}_i)d\mathbf{\Omega}_i$ can be calculated analytically as $\frac{2}{\ _j}(1-e^{-\ '_j})$. The integral of the product of L and G is calculated as a 3D function $\Gamma_L(\ ,\ ,\ )$:

$$\int_{\mathbb{S}^2} L(\mathbf{\Omega}_i)G(\mathbf{\Omega}_i;\ ,\ ,\mu)d\mathbf{\Omega}_i=\mu\Gamma_L(\ ,\ ,\ ). \tag{10}$$

Our method precomputes $\Gamma_L(\ ,\ ,\ )$ and stores the data as prefiltered environment maps for various lobe sharpness as [WRG*09]. Therefore, our method can change the BRDFs (i.e. the SGs) at run-time. Since our method integrates the product of the BRDF approximated by the SGs and the lighting, our method can represent highly specular materials.

In summary, efficient computation of the integral of the product of the SGs and the visibility function is the key element for both diffuse and specular components to achieve real-time rendering of fully dynamic scenes. We propose a

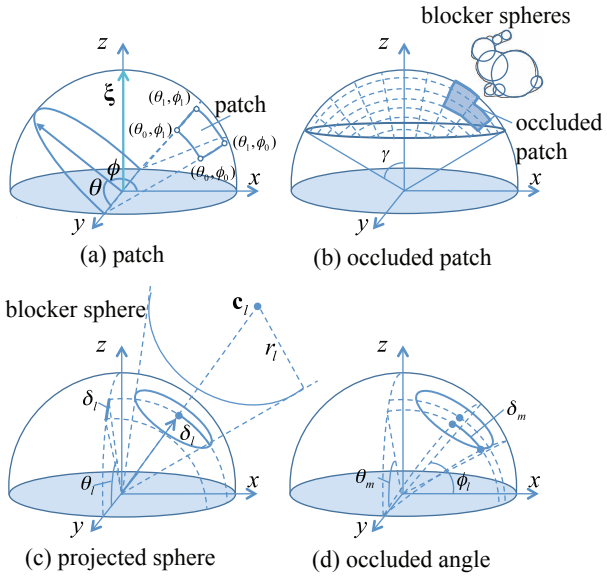


Figure 2: *Local coordinate system for integral spherical Gaussian and patch.*

fast calculation method of the integral of the product in the next Section.

4. Integral Spherical Gaussian

We now focus on the calculation of the integral of the product of the visibility function and the SGs. Our method sets the local coordinate system so that the axis is set to the z axis as shown in Fig. 2, and the direction $\boldsymbol{\Omega}$ is parameterized by two angles $\in [0,]$ and $\in [0,]$ as:

$$\boldsymbol{\Omega}(,) = (\sin \cos , \cos , \sin \sin) , \qquad (11)$$

where is the angle between $\boldsymbol{\Omega}$ and the y axis, and is the angle between $\boldsymbol{\Omega}$ and the xy plane. We define the integral spherical Gaussian $S(, ,)$ as

$$S(, ,) = \int_0 \int_0 G(\boldsymbol{\Omega}(', '); \mathbf{z}, , 1) \sin ' d ' d ' . \qquad (12)$$

We also define the normalized integral spherical Gaussian $\hat{S}(, ,)$ as:

$$\hat{S}(, ,) = S(, ,) / S(, ,) = \frac{S(, ,)}{(2 /)(1 - e^{- })} . \qquad (13)$$

Fig. 3 shows normalized ISGs for different values of lobe sharpness . Our method approximates the normalized ISG with the following function $f(, ,)$ consisting of two sigmoid functions:

$$f(, ,) = \frac{1}{1 + e^{-g( )(- /2)}} \frac{1}{1 + e^{-h( )(- /2)}}, \qquad (14)$$

where g and h are polynomial functions of lobe sharpness . Our method fits g and h to fourth degree polynomials

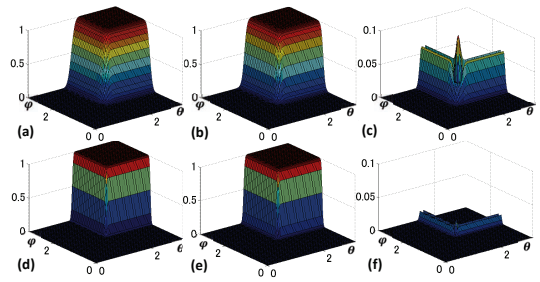


Figure 3: *3D plots of (a)$\hat{S}(, , 100)$, (b) $f(, , 100)$ and (c) difference between (a) and (b) , and (d)$\hat{S}(, , 1000)$, (e)$f(, , 1000)$, and (f) difference between (d) and (e). Relative squared error between (a) and (d) is 0.061%, and 0.0013% between (b) and (e). Maximum error between (a) and (b) is 0.0966 and that between (d) and (e) is 0.0251.*

for various sharpness (from 1 to 10000) using the least squares method. The polynomials of g and h are described in the Appendix. Consequently, ISG $S(, ,)$ is calculated by $\frac{2 }{ }(1 - e^{- }) f(, ,)$.

Our method calculates the integral of the product of the SGs and the visibility function by subtracting the integral of the SGs over the occluded region $\bar{\Omega}$ on the hemisphere from the ISG $S(, ,)$ as follows:

$$\int_{\mathbb{S}^2} G(\boldsymbol{\Omega}_i; \mathbf{z}, , 1) V(\boldsymbol{\Omega}_i) d\boldsymbol{\Omega}_i = S(, ,) - \int_{\bar{\Omega}} G(\boldsymbol{\Omega}_i; \mathbf{z}, , 1) d\boldsymbol{\Omega}_i .$$

To simplify the explanation, we first describe the integral of the SG over the occluded region $\bar{\Omega}$ bounded by four directions $\boldsymbol{\Omega}(0, 0)$, $\boldsymbol{\Omega}(0, 1)$, $\boldsymbol{\Omega}(1, 0)$, and $\boldsymbol{\Omega}(1, 1)$ as shown in Fig. 2(a). The area bounded by these four corner directions is referred to as the *patch* $P(0, 1, 0, 1)$.

Then the integral of the SG over the patch $P(0, 1, 0, 1)$ is efficiently calculated by:

$$\int_0^1 \int_0^1 G(\boldsymbol{\Omega}(', '); \mathbf{z}, , 1) \sin ' d ' d '$$
$$= S(1, 1,) - S(0, 1,) - S(1, 0,) + S(0, 0,).$$

4.1. Calculation of Occluded Patches

With our method, the integral of the SG over the occluded regions due to multiple overlapping blocker spheres can be calculated using ISG. To compute the integral efficiently, our method discretizes the surface on the hemisphere into small patches adaptively based on the lobe sharpness . Firstly, we calculate the limiting angle Γ that satisfies $\frac{1}{1 + e^{-g( )(\Gamma - /2)}} > $, where the threshold is specified by the user. In our implementation, we set to 0.01 and obtain good results. Then our method uniformly subdivides the surface on the hemisphere around the z axis within the limiting angle Γ as shown in Fig. 2(b).

For each blocker sphere, the patches overlapped by the

projected blocker sphere are calculated and considered as occluded. The occluded patches are determined by using the four angles of the patches. Let $\mathbf{c}_l$ and r_l be the center and the radius of l-th blocker sphere in the local coordinate system (see Fig. 2(c)). We calculate the angles $\quad_l$ and $\quad_l$ for $\mathbf{c}_l$, and the angle $\quad_l$ that corresponds to the radius of the projected blocker sphere as $\quad_l = \sin^{-1}(r_l/||\mathbf{c}_l||)$. Then the occluded region in the direction is calculated as $\quad_l - \quad_l \leq \quad \leq \quad_l + \quad_l$. For the m-th discretized angle $\quad_m$ as shown in Fig. 2(d), the occluded region in the direction is $\quad_l - \quad_m \leq \quad \leq \quad_l + \quad_m$, where $\quad_m$ is calculated as $\quad_m = \frac{\sqrt{\quad_l^2 - (\quad_l - \quad_m)^2}}{\sin \quad_m}$. A patch whose four angles overlap these regions is determined as occluded.

Finally, the ISGs of the occluded patches are calculated and subtracted from $\frac{2}{\quad}(1 - e^{-\quad})$, resulting in the integral of the product of the visibility function and the SG. For narrow SG lobes, since the limiting angle is very small, the discretized patches are very small and this provides good accuracy for discretization. For broader SG lobes, the SG changes smoothly and large patches are sufficient to approximate occluded regions.

4.2. Discussion

To integrate the product of the SG and the visibility, the previous method [WRG*09] proposed a spherical signed distance function (SSDF) $V^d(\quad)$. The SSDF $V^d(\quad)$ returns the signed angular distance between the lobe axis and the nearest boundary direction. The visible region is approximated as the region where the angle in Fig. 2 is larger than $\quad/2 - V^d(\quad)$. Then the integral of the product of the visibility function and the SG is calculated by integrating the SG over the visible region $\geq \quad/2 - V^d(\quad)$. That is, the integral calculation using the SSDF can be viewed as a special case of our ISG as $S(\quad, \quad/2 - V^d(\quad), \quad)$. Although the SSDF representation can render realistic shadows in the case where the occluded region widely covers the y axis (i.e. in Fig. 2 is nearly), the SSDF can overestimate the occluded region by assuming that is . Fig. 4 shows a comparison between our method using an ISG, the SSDF, and a ray tracing method. As shown in Fig. 4, shadow regions computed using the SSDF are large compared to the reference solution, and is overall dark due to overestimation of the invisible region. The ISG can calculate shadows more accurately, and can produce plausible shadows matching the reference solution. Note that computing SSDF is time-consuming even for an object approximated by spheres, especially for the SGs whose lobe axis direction is invisible (occluded). In this case, we need to shoot many rays to search for the nearest visible direction. This process is too expensive to compute at runtime for dynamic scenes.

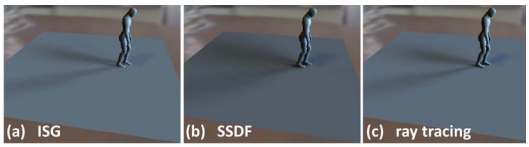


Figure 4: *Comparison of shadows using ISG and SSDF. To compute ISG and SSDF, a sphere set that approximates the character geometry is used. To render (c) using ray tracing, the character geometry itself is used to compute the visibility. The environment map is approximated by 9 SGs.*

5. Implementation Details

Our method calculates the outgoing radiance $k_d L_d + k_s L_s(\boldsymbol{\Omega}_o)$ by approximating the object geometries in the scene with a set of spheres. The computational cost of the diffuse component L_d for each point is $O(KN_s)$ and that of the specular component L_s is $O(JN_s)$, where K is the number of SGs to represent the lighting L, J is the number of SGs to represent the BRDF , and N_s is the number of spheres. As described in [WRG*09], K and J are small enough to produce plausible lighting (e.g. $K \leq 10$ and $J \leq 7$ for typical scenes). Therefore the performance of our method mainly depends on the number of spheres.

To render dynamic scenes efficiently, our method represents the spheres using a bounding volume hierarchy (BVH). A set of spheres that approximates each object geometry is prepared using [WZS*06] or [BO04]. These spheres are considered as leaf nodes and parent nodes are constructed that bound spheres of child nodes. In our implementation, a binary tree structure is used to represent the BVH. The BVH is constructed using [LGS*09] whose complexity is $O(N_s)$ and therefore this method scales well for many spheres.

To compute the integral of the product of the SG with lobe axis and the visibility, our method traverses the binary tree of spheres. As described in the previous section, our method only considers the spheres that intersect a cone defined by the limiting angle Γ centered on the lobe axis as shown in Fig. 5. From the root node, our method calculates the cut of the binary tree as follows. For each node, our method first determines whether the bounding sphere corresponds to the node intersecting the cone or not. If the sphere does not intersect the cone, the node and its descendant nodes are culled. If the sphere intersects the cone, the projected sphere in the local coordinate system is calculated. If the projected sphere is larger than the threshold specified by the user, our method traverses the child nodes. Otherwise, the patches that overlap the projected sphere are calculated and considered as occluded.

Although sphere approximation can accelerate the rendering, it is difficult to approximate thin shells (e.g. clothes) using spheres. To handle thin shells, our method projects triangle meshes instead of spheres, calculating the occluded patches by rasterizing the projected triangle meshes similar to [KLA04].

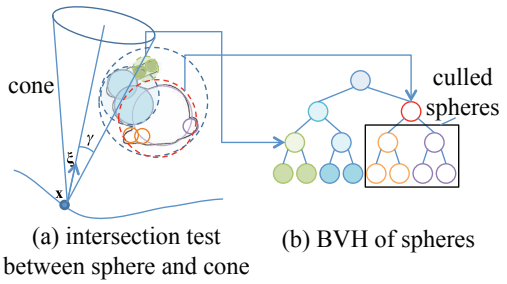


Figure 5: *Intersection tests between spheres and cones using BVH. Dashed (solid) spheres in (a) represent inner (leaf) nodes of BVH in (b). Shaded spheres in (a) intersect the cone. Since red dashed sphere does not intersect the cone, the descendant nodes are culled.*

Our method utilizes two methods to calculate the outgoing radiance, the per-vertex shadowing method and the per-pixel shadowing method. For the per-vertex shadowing method, the integral of the product of the SG and the visibility is calculated at each vertex. Then the integral is interpolated and shading is performed at each pixel using a fragment program. For the per-pixel shadowing method, our method calculates the integral at each pixel using CUDA. To do this, our method first renders the scene into the framebuffer object to store the position and normal at each pixel. Then the buffer is transferred to CUDA and the integration of the SG is performed at each pixel.

6. Results

We have implemented our rendering algorithm on a standard PC with an Intel Corei7 Extreme 965 CPU and a GeForce GTX 580 GPU. The statistics of examples in this paper are listed in Table. 1. The SGs that approximate the environmental lighting $L(\boldsymbol{\Omega}_i)$ are calculated using [TS06]. Our method represents isotropic BRDFs such as the Blinn-Phong BRDF model and the Cook-Torrance BRDF model with 1 SG, and the anisotropic BRDF (Ashikhmin-Shirely BRDF) with 7 SGs. The precomputation time and the data size for Γ_L are 10 min and 6MB, respectively.

Fig. 6 shows images of a Buddha model; (a), (b) with shadows and (c) without shadows. As shown in Fig. 6(a), our method can render self-shadows onto the pedestal. Fig. 6(b) shows the shadows due to the Buddha model with specular reflections on the floor. Fig. 7 shows still images of a chess scene that consists of 32 chess pieces and a board. The BRDFs of the chess pieces can be changed from diffuse to highly specular interactively at run-time. Our method discretizes the hemisphere into 100 to 400 patches in these examples. Fig. 8 shows still images of six deformable characters. Since our method does not require precomputed visibility data, our method can render deformable objects in real-time. Fig. 9 shows still images of rigid body simulations. The interactions between 8 bunny models and 8 chess pieces are simulated using NVIDIA PhysX. Fig. 10 shows still images of simulations of interactions between a cloth and 4 bunny models. As shown in Fig. 10, our method can deal with thin shells (e.g. the cloth). The shadows due to the cloth are rendered by projecting triangle meshes of the cloth and calculating the overlapped occluded patches.

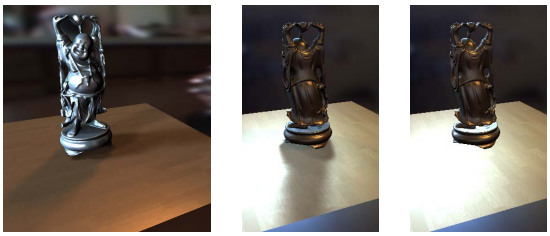
Fig. 11 shows images rendered using (a) our method, (b) a ray tracing method, and (c), (d) shadow map methods. The BRDF of the bunny model is the Blinn-Phong BRDF whose glossiness parameter is 1000. As shown in Fig. 11, the image rendered using our method matches that rendered using the ray tracing method. Fig. 11(c) is rendered using 30 shadow maps which is equal to the number of shadow maps used in [ADM*08]. The lighting directions of the shadow maps are calculated using importance sampling of the environment maps. As shown in Figs. 11, the specular reflections of a highly specular BRDF cannot be rendered using only 30 shadow maps. Fig. 11(d) is rendered using 1024 shadow maps which is equal to the number of shadow maps in [RGK*08], but the specular reflection is still different from Fig. 11(b). The computational times for Fig. 11 are (a) 14.5ms, (b) 1044sec, (c) 23.0ms, and (d) 262ms, respectively. Similar to Fig. 11(c), the convolution shadow map method [ADM*08] does not integrate the product of the BRDF and the incident radiance. Therefore, this method fails to capture highly specular reflections. As described in [RGK*08], the imperfect shadow map method can render the scene (Fig. 11 in [RGK*08]) in 25ms, while the rendering time using 1024 classic shadow maps is 405ms. Therefore, the estimated time to render Fig. 11(d) is about 16.2ms using imperfect shadow maps, comparable to our method. However, our method can render specular reflections better as shown in Fig. 11 (especially as highlighted in the red frame).

Fig. 12 shows a comparison between our method and a ray tracing method. Figs. 12(a) and (d) are rendered by using our method, Figs. 12(b) and (e) are rendered by using a ray tracing method, and Figs. 12(c) and (f) are difference images scaled by 4. Figs. 12 (b) and (d) are rendered by tracing $6 \times 64 \times 64$ rays per pixel to sample the incident radiance. The RMS errors in (c) and (f) are 3.2% and 4.6%, respectively. As shown in Fig. 12, our result matches the reference solution very well and any differences are subtle.

Although our method can render convincing results, it also has several limitations. Firstly, our method handles only direct illumination. This may be addressed by combining [LWDB10] with our method, but this remains as a future work. For almost ideally-specular surfaces, the circle boundaries due to the sphere set approximation may be visible. This may be relaxed by increasing the number of spheres approximating the geometries at the cost of decreasing the rendering performance. However, as shown in Fig. 12, our

Table 1: *Rendering performance. #vertices and #spheres represent the number of vertices and spheres, respectively. fps represents the rendering frame rate for fully dynamic scenes (dynamic lighting, objects, BRDFs, viewpoint). The image resolutions are* 640×480 *except for Fig. 11(*400×400*). Per-vertex shadowing is used except for Fig. 12 in which per-pixel shadowing is performed.*

Figure	# vertices	# spheres	fps
Fig. 6	57K	46	40fps
Fig. 7	101K	306	7-9fps
Fig. 8	27K	192	33-47fps
Fig. 9	66K	192	13-60fps
Fig. 10	25K	216	2.5-3fps
Fig. 11	18K	53	69fps
Fig. 12	18K	53	10fps



(a) (b) (c)

Figure 6: *Buddha scene (a) viewed from front, (b) viewed from back with shadows, and (c) without shadows.*

method can render plausible hard shadows comparable to the reference solution.

7. Conclusions and Future Work

We have proposed a real-time rendering method for fully dynamic scenes with highly specular BRDFs illuminated by all-frequency lighting. The key contribution of our method is an efficient method for calculating the integral of the product of various SGs and the visibility function. The integral of the product of arbitrary SGs and visibility is efficiently calculated using the sum of the integral spherical Gaussians which can be easily evaluated on-the-fly for various SG lobe sharpness. The integral of the triple product of the lighting, the visibility function and the BRDFs can be efficiently calculated using the integral spherical Gaussian.

In future work, we would like to extend our method to calculate indirect illumination.

Acknowledgments

This work was supported in part by Grant-in-Aid for Scientific Research(C) No. 20500082, and by the Japan Prize Foundation.

References

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3 (2008), 34:1–34:8. 1, 2, 6

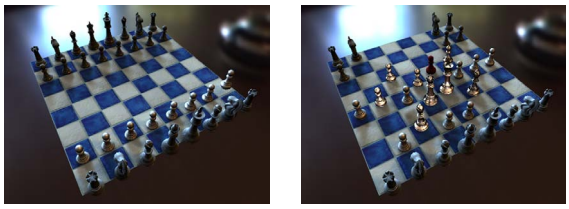


Figure 7: *Chess scene.*

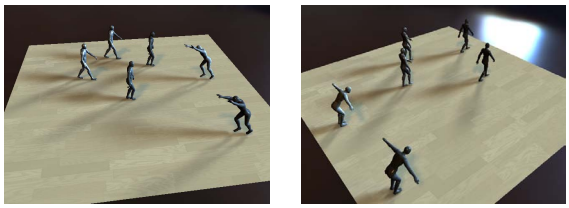


Figure 8: *Character scene.*

[BO04] BRADSHAW G., O'SULLIVAN C.: Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph. 23*, 1 (2004), 1–26. 5

[Cro84] CROW F. C.: Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph. 18* (1984), 207–212. 3

[GKD07] GREEN P., KAUTZ J., DURAND F.: Efficient reflectance and visibility approximations for environment map rendering. *Computer Graphics Forum 26*, 3 (2007), 495–502. 1, 2

[HREB11] HOLLANDER M., RITSCHEL T., EISEMANN E., BOUBEKEUR T.: Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum 30* (2011), 1233–1240. 2

[KC08] KRIVANEK J., COBERT M.: Real-time shading with filtered importance sampling. *Computer Graphics Forum 27*, 4 (2008), 1147–1154. 2

[KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical rasterization for self-shadowing of dynamic objects. In *Eurographics Symposium on Rendering* (2004), pp. 179–184. 2, 5

[LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast bvh construction on gpus. *Computer Graphics Forum 28*, 2 (2009), 375–384. 5

[LHLW10] LAM P.-M., HO T.-Y., LEUNG C.-S., WONG T.-T.: All-frequency lighting with multiscale spherical radial basis functions. *IEEE Transactions on Visualization and Computer Graphics 16*, 1 (2010), 43–56. 2

[LWDB10] LAURIJSSEN J., WANG R., DUTRÃL' P., BROWN B.: Fast estimation and rendering of indirect highlights. *Computer Graphics Forum 29*, 4 (2010), 1305–1313. 6

[NKF09] NOWROUZEZAHRAI D., KALOGERAKIS E., FIUME E.: Shadowing dynamic scenes with arbitrary brdfs. *Computer Graphics Forum 28*, 1 (2009), 249–258. 1, 2

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 3 (2003), 376–381. 1, 2

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. 23*, 3 (2004), 477–487. 2

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps

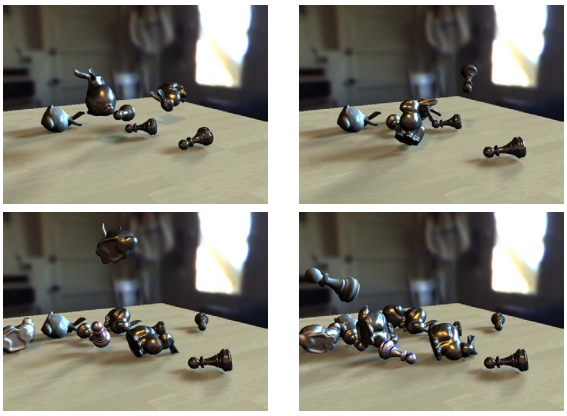


Figure 9: *Falling rigid objects scene.*

Figure 10: *Cloth scene.*

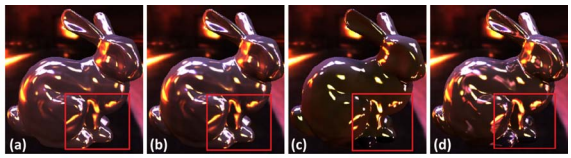


Figure 11: *Comparison of (a) our method (14.3ms), (b) ray tracing method (1044sec), (c) 30 shadow maps (23.0ms), and 1024 shadow maps (262ms). Shadow map resolutions of (c) and (d) are* 256×256.

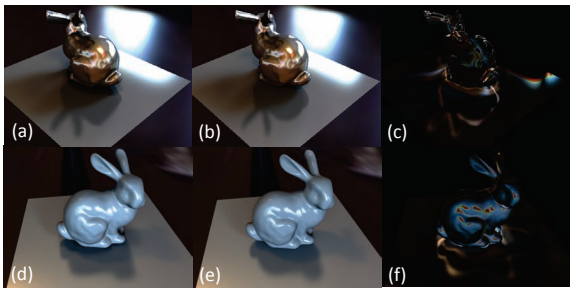


Figure 12: *Comparison of our method and ray tracing method. BRDFs of the bunny and floor for (a) and (b) are both Phong BRDF model (the glossiness parameter is 1000 and* $k_s = 0.5$*). For (d) and (e),* k_s *of the bunny is set to 0.01. (c) and (f) are difference images scaled by 4.*

for efficient computation of indirect illumination. *ACM Trans. Graph. 27*, 5 (2008), 129:1–129:8. 1, 2, 6

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Trans. Graph. 25*, 3 (2006), 977–986. 1, 2

[SGNS07] SLOAN P. P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics 2007* (2007), pp. 97–105. 2

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (2002), 527–536. 1, 2

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. *ACM Trans. Graph. 24*, 3 (2005), 1216–1224. 2

[SM06] SUN W., MUKHERJEE A.: Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Trans. Graph. 25*, 3 (2006), 955–966. 2

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic brdfs. *ACM Trans. Graph. 26*, 3 (2007), 27:1–27:10. 2

[TJCN06] TAMURA N., JOHAN H., CHEN B.-Y., NISHITA T.: A practical and fast rendering algorithm for dynamic scenes using adaptive shadow fields. *The Visual Computer* (2006), 702–712. 2

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25*, 3 (2006), 967–976. 1, 2, 6

[VJ01] VIOLA P., JONES M.: Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2001* (2001), pp. 511–518. 3

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. 28*, 5 (2009), 133:1–133:10. 1, 2, 3, 5

[WTL06] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of glossy objects. *ACM Trans. Graph. 25*, 2 (2006), 293–318. 2

[WZS*06] WANG R., ZHOU K., SNYDER J., LIU X., BAO H., PENG Q., GUO B.: Variational sphere set approximation for solid objects. *Vis. Comput. 22*, 9 (2006), 612–621. 5

[XJF*08] XU K., JIA Y.-T., FU H., HU S., TAI C.-L.: Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 454–467. 2

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph. 24*, 3 (2005), 1196–1201. 2

Appendix

The polynomial parameters for g and h are listed in the following:

$$g(\ )=g_4(\ /100)^4+g_3(\ /100)^3+g_2(\ /100)^2+g_1\ /100+g_0,$$
$$h(\ )=h_4(\ /100)^4+h_3(\ /100)^3+h_2(\ /100)^2+h_1\ /100+h_0,$$
$$(g_4,g_3,g_2,g_1,g_0)=(-2.6856e^{-6},7e^{-4},-0.0571,3.9529,17.6028),$$
$$(h_4,h_3,h_2,h_1,h_0)=(-2.6875e^{-6},7e^{-4},-0.0592,3.9900,17.5003).$$