

Cosine Lobes for Interactive Direct Lighting in Dynamic Scenes

Sylvain Meunier, Daniel Meneveaux, Lilian Aveneau, Djamchid Ghazanfarpour

▶ **To cite this version:**

Sylvain Meunier, Daniel Meneveaux, Lilian Aveneau, Djamchid Ghazanfarpour. Cosine Lobes for Interactive Direct Lighting in Dynamic Scenes. 2009. hal-00401728

HAL Id: hal-00401728
https://hal.science/hal-00401728

Submitted on 5 Jul 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cosine Lobes for Interactive Direct Lighting in Dynamic Scenes

S. Meunier[1], D. Meneveaux[1], L. Aveneau[1], D. Ghazanfarpour[2]

[1] Xlim laboratory, CNRS UMR 6172, University of Poitiers, France
[2] Xlim laboratory, CNRS UMR 6172, University of Limoges, France

Abstract

Many rendering systems rely on spherical harmonics or wavelets that provide a mean to solve the rendering equation using scalar products. In addition, these basis of functions may represent hemispherical lighting, visibility and reflectance with a small number of coefficients. However, their use requires the projection of each term of the rendering equation, which is computationally intensive at run-time (for instance with dynamic environments), and limits practically the number of basis functions used at the expense of precision. In this paper, we show how cosine lobes can also be used for representing each term of the rendering equation, without the drawbacks usually existing in the above basis of functions. Cosine lobe representations (such as Lambert, Phong or Lafortune models) have already been used intensively by many authors for their advantage of intuitively and compactly representing reflectance functions. Using this representation also for visibility and lighting, we explain how the rendering equation can be efficiently solved. As an application, we propose an interactive rendering system for direct lighting, naturally including soft shadows and spatially varying materials.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Bitmap and frame buffer operations I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

Several methods in the literature are proposed for producing physically-based images with lighting simulations, complex BRDF (bidirectional reflectance distribution function), natural lighting, a high number of objects and so on [PH04]. Unfortunately, producing realistic images at interactive frame rates in dynamic scenes remains a challenging task. Many authors have provided interesting solutions with the use of orthogonal basis functions such as spherical harmonics (SH) [SKS02], hemispherical harmonics (HSH) [Gau06] or Haar wavelets (HW) [NRH03]. This type of representation simplifies the rendering equation computation and replaces the continuous integration by products of coefficients. Nevertheless, the projection of visibility, BRDF and/or incoming radiance on the basis functions remains a costly process. This problem is often addressed offline, with pre-computed radiance transfer methods (PRT) [SKS02, NRH03], so as to provide real time rendering. However, PRT still is difficult to manage with fully dynamic scenes because the stored data increases rapidly, unless some parameters be fixed (e.g. viewpoint, objects, BRDF, etc.). Some applications compute only a few coefficients for ensuring interactivity rather than relying on pre-computed transfers [KLA04]. These methods inhibit all frequency lighting environments with SH and produce blocking artifacts with HW.

In this paper we propose a method for overcoming the limitations of previous methods, using the flexibility of spherical radial basis functions (SRBF) [TS06]. They are defined on the sphere, not necessarily uniformly distributed, invariant under rotation and spatially localized; they only require choosing an axis and some shape parameters which is much meaningful; the representation is compact (a small number of coefficients accurately represent data). Our approach is based on cosine lobes, providing a mean for using directly Phong and Lafortune BRDF models, with potentially spatially varying, anisotropic, retro-reflection and/or off-specular reflectance data [MLH02]. It provides practical solutions to represent many reflectance properties. Our contributions include:

- a method for solving rendering equation using a homogeneous representation based on cosine lobes;
- an acceleration system based on table fetching and linear interpolations;
- an application using cosine lobes for estimating shadows and direct lighting with various types of BRDF;
- an interactive rendering system providing all-frequency shadows and realistic materials in dynamic scenes.

This paper shows how the rendering equation can be efficiently solved when each term is represented by cosine lobes. Our application does not address diffuse interreflections that should be handled using existing approximations proposed in [SGNS07, GJW08].

The remaining of this paper is organized as follows: Section 2 reviews existing methods and highlights our contributions; Section 3 presents an overview of our approach. Section 4 details the use of cosine lobes for the rendering equation; Section 5 explains the representation of visibility, BRDF and incoming radiance using cosine lobe sums; Section 6 presents and discusses the performances provided by our method; Finally, Section 7 concludes and mentions future work.

2. Related Work

Let us consider the rendering equation :

$$L_o(x,\overrightarrow{v_o}) = \int_{\Omega+} f_r(x,\overrightarrow{v},\overrightarrow{v_o}) L_i(x,\overrightarrow{v})(\overrightarrow{n}\cdot\overrightarrow{v})d\overrightarrow{v} \quad (1)$$

$$= \int_{\Omega+} \tilde{f}_r(x,\overrightarrow{v},\overrightarrow{v_o}) L_i(x,\overrightarrow{v})d\overrightarrow{v} \quad (2)$$

where L_o denotes the outgoing radiance from a point x in direction $\overrightarrow{v_o}$; $\Omega+$ is the upper hemisphere with solid angles $d\overrightarrow{v}$ around directions $\overrightarrow{v}$; f_r is the BRDF and $\tilde{f}_r$ the BRDF multiplied by $(\overrightarrow{n}\cdot\overrightarrow{v})$; L_i is the incoming radiance at x from direction $\overrightarrow{v}$ and n the surface normal at x.

Let L_s be the environment lighting at a point x from any direction $\overrightarrow{v}$. When objects are located between x and the lighting environment, visibility has to be considered. Equation 3 can thus be rewritten :

$$L_o(x,\overrightarrow{v_o}) = \int_{\Omega+} \tilde{f}_r(x,\overrightarrow{v},\overrightarrow{v_o}) L_s(x,\overrightarrow{v}) V(x,\overrightarrow{v})d\overrightarrow{v} \quad (3)$$

V being the visibility term.

We aim at dealing with dynamic scenes (with potentially varying BRDF, moving lighting sources and objects), so that the triple product $\tilde{f}_r \cdot L_s \cdot V$ be solved at run-time. Several methods choose SH or HW basis of functions for representing $\tilde{f}_r$, L_s and V, such as [KLA04, ZHL*05, RWS*06, SGNS07, GJW08].

Generally speaking, orthogonal basis functions such as SH and HW provide a useful representation of functions on the sphere. Once the projection is performed, the integration becomes a dot product, allowing real-time or interactive rendering for dynamic scenes. However, all the coefficients have to be estimated even though some (potentially *many*) of them only contribute negligibly to lighting computations [KLA04].

Shadow fields [ZHL*05] tabulate SH or HW visibility representation in the space surrounding each blocker, requiring large tables and many triple product computations. SH exponentiation [RWS*06, SGNS07, GJW08] replaces the SH triple products by fast coefficients accumulation in log space. With bounding sphere sets, these methods are capable of handling more complex scenes and skinned objects. Nevertheless, SH limit the application to low frequency lighting environments. Kautz *et al*. [KLA04] propose a hemispherical rasterization of blocker geometry for estimating a SH representation of the *transfer function* ($\tilde{f}_r.V$ in this case). This method also uses SH with low memory requirements and no precomputation based on the scene geometry. [KLA04, ZHL*05, RWS*06, SGNS07, GJW08] are commonly denoted as *blocker accumulation* methods and imply visibility computations for each shaded point.

Kozlowski and Kautz [KK07] perceptually analyze possible simplifications for visibility. They propose a directional ambient occlusion method based on a piecewise constant approximation of V. SH are used to smooth the results and even with approximated shadows and highlights, images remain visually plausible. Green et al. [GKMD06, GKD07] use a similar simplification for shadows, associated with isotropic gaussian functions (SRBF) for representing BRDF and/or visibility; incoming radiance is represented by environment maps, pre-filtered using gaussian functions, allowing real time rendering. These methods have been designed for existing environment maps and do not allow interactive dynamic lighting.

SRBF are well known in the computer graphics community. For instance the generalized cosine lobe [LFTG97] and the isotropic Gaussian kernel [War92] are used to model BRDF, leading to a compact, expressive and physically plausible representation. Tsai and Shih [TS06] adapt the Abel-Poisson kernel to PRT. The compactness of SRBF leads to a more efficient compression scheme than the clustered principal component analysis proposed in [SHHS03] and provides faster rendering with more realistic results.

The attractiveness of SH/HW is practically limited by the projection cost for high-frequency environments or BRDF. For instance a small detail may require a lot of (sometimes null) coefficients with SH/HW when only one cosine lobe can be efficiently used. In addition, some useless computations have to be performed during the SH/HW projection, when null coefficients are produced. We propose to overcome this limitation with the use of an unfixed (contrary to [TS06]) set of cosine lobes for every term : visibility, lighting and reflectance.

3. Overview

We propose to represent each term of the triple product $\tilde{f}_r \cdot L_s \cdot V$ by cosine lobes in the rendering equation (Equation 3). The integration is performed based on the idea that the product of two lobes can be approximated by another lobe (Sections 4). We provide an interactive application dedicated to direct lighting (Section 5), where our approximation proves efficient with freely and dynamically distributed lobes, avoiding useless computations of null coefficients (Section 6).

We have implemented a blocker accumulation system that benefits from SRBF (with cosine lobes) advantages, while providing interactive rendering in dynamic scenes with all-frequency BRDF, visibility and incoming radiance. Therefore, scene geometry is simplified using sphere sets (blockers and light sources); from a given point, a sphere defines a cone, used for constructing a lobe.

The triple product $\tilde{f}_r \cdot L_s \cdot V$ cosine lobes are generated using:

- the BRDF (Lambert, Phong, Lafortune, etc.) representation for $\tilde{f}_r$ (see Section 5.1);
- a set of spherical light sources for L_s, with one lobe for each sphere (see Section 5.2);
- for V, a visibility mask is produced using the simplified objects geometry for constructing again spheres and thus cosine lobes (see section 5.3).

4. The Rendering Equation and Cosine Lobes

A cosine lobe function is written:

$$c(\overrightarrow{v}) = s\ \max(\overrightarrow{a}\cdot\overrightarrow{v},0)^e \quad (4)$$
$$= s\ \Psi(\overrightarrow{v}) \quad (5)$$

where s is the scaling factor, $\overrightarrow{a}$ is the axis and e is the lobe *thickness*.

Fixing x, $\overrightarrow{v_o}$ and assuming $\tilde{f}_r$, L_s and V approximated by any cosine lobe sums:

$$\tilde{f}_r(\overrightarrow{v}) = \sum_i \left(s_i\Psi_i(\overrightarrow{v})\right) \quad (6)$$
$$L_s(\overrightarrow{v}) = \sum_j \left(s_j\Psi_j(\overrightarrow{v})\right) \quad (7)$$
$$V(\overrightarrow{v}) = \sum_k \left(s_k\Psi_k(\overrightarrow{v})\right) \quad (8)$$

Note that $s_i\Psi_i$, $s_j\Psi_j$ and $s_k\Psi_k$ do generally not correspond to the same cosine lobes (even when $i=j=k$).

Equation 3 can be rewritten:

$$L_o = \int_\Omega \sum_i \left(s_i\Psi_i(\overrightarrow{v})\right)\sum_j \left(s_j\Psi_j(\overrightarrow{v})\right)\sum_k \left(s_k\Psi_k(\overrightarrow{v})\right)\, d\overrightarrow{v} \quad (9)$$

$$= \sum_i\sum_j\sum_k s_i s_j s_k \int_\Omega \Psi_i(\overrightarrow{v})\Psi_j(\overrightarrow{v})\Psi_k(\overrightarrow{v})\, d\overrightarrow{v} \quad (10)$$

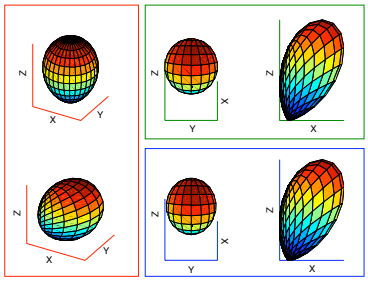


Figure 1: *Example of cosine lobe product. Cosine lobes in the left box are multiplied. The top-right box provides the resulting surface while the bottom-right box illustrates the (single) lobe corresponding to our approximation: the global shape and direction are preserved.*

However, the term $\int_\Omega \Psi_i(\overrightarrow{v})\Psi_j(\overrightarrow{v})\Psi_k(\overrightarrow{v})\, d\overrightarrow{v}$ cannot be precomputed because cosine lobes are not fixed. Thus, starting from equation 9:

$$L_o = \int_\Omega \sum_{i,j}\left[s_i\Psi_i(\overrightarrow{v})s_j\Psi_j(\overrightarrow{v})\right]\sum_k s_k\Psi_k(\overrightarrow{v})\, d\overrightarrow{v} \quad (11)$$

we propose to approximate $\left[s_i\Psi_i(\overrightarrow{v})s_j\Psi_j(\overrightarrow{v})\right]$ by a single cosine lobe $s_l\Psi_l(\overrightarrow{v})$:

$$L_o \approx \int_\Omega \sum_l s_l\Psi_l(\overrightarrow{v})\sum_k s_k\Psi_k(\overrightarrow{v})\, d\overrightarrow{v} \quad (12)$$
$$= \int_\Omega \sum_{l,k}\left[s_l\Psi_l(\overrightarrow{v})s_k\Psi_k(\overrightarrow{v})\right]\, d\overrightarrow{v} \quad (13)$$

The rendering equation finally becomes a sum of cosine lobe integrations:

$$L_o \approx \int_\Omega \sum_m s_m\Psi_m(\overrightarrow{v})\, d\overrightarrow{v} = \sum_m s_m \int_\Omega \Psi_m(\overrightarrow{v})\, d\overrightarrow{v} \quad (14)$$

Sections 4.1 and 4.2 describe how the product of two lobes is managed and how the cosine lobe integration can be precomputed.

4.1. Product

We propose to approximate the product of two cosine lobes c_1 and c_2 by another single cosine lobe c_r:

$$c_1(\overrightarrow{v})c_2(\overrightarrow{v}) \approx c_r(\overrightarrow{v}) \quad (15)$$
$$\approx s_1 s_2 p_r \max(\overrightarrow{a_r}\cdot\overrightarrow{v},0)^{e_r} \quad (16)$$

p_r is called the *partial* scaling factor. The parameters p_r, $\overrightarrow{a_r}$ and e_r depend on e_1, e_2 and α (the angle between $\overrightarrow{a_1}$ and $\overrightarrow{a_2}$). p_r, $\overrightarrow{a_r}$ and e_r are precomputed for a wide range of (e_1, e_2, α) triplets and store the result in a table. For maintaining directionality and shape for the approximated cosine lobe (see Figure 1), we choose to minimize the L_2 distance

between $c_1(\overrightarrow{v}) \cdot c_2(\overrightarrow{v})$ and $c_r(\overrightarrow{v})$:

$$\{p_r, \overrightarrow{a_r}, e_r\} = \underset{\{p_r,\overrightarrow{a_r},e_r\}}{\arg\min} \int_\Omega \left(c_1(\overrightarrow{v})c_2(\overrightarrow{v}) - c_r(\overrightarrow{v})\right)^2 d\overrightarrow{v} \tag{17}$$

Since solving this optimization problem for the parameters altogether is time-consuming and prone to numerical errors, we propose an approach in two steps: (i) finding the axis $\overrightarrow{a_r}$ and (ii) fitting the partial scaling factor p_r and the exponent e_r.

First step:

$$\{a_r\} = \underset{\{\overrightarrow{a_r}\}}{\arg\max}\, c_1(\overrightarrow{a_r})c_2(\overrightarrow{a_r}) \tag{18}$$

is done with the Nelder-Mead method. The algorithm is initialized with $\frac{\overrightarrow{a_1}+\overrightarrow{a_2}}{\|\overrightarrow{a_1}+\overrightarrow{a_2}\|}$ since $\overrightarrow{a_r}$ is necessary between $\overrightarrow{a_1}$ and $\overrightarrow{a_2}$.

Second step:

$$\{p_r, e_r\} = \underset{\{p_r,e_r\}}{\arg\min} \int_\Omega \left(c_1(\overrightarrow{v})c_2(\overrightarrow{v}) - c_r(\overrightarrow{v})\right)^2 d\overrightarrow{v} \tag{19}$$

is done with the Levenberg-Marquardt algorithm. e_r is initialized with $e_1 + e_2$ which is the solution for $\overrightarrow{a_1} = \overrightarrow{a_2}$ and p_r with 1.

In practice e_1 and e_2 are sampled in $[0, 100000]$ so that any scene type be handled. α is ranged in $[0, \pi]$. Our sampling strategy is based on the cosine lobe integration variation (see Figure 2). We pick 16 linearly spaced samples in each interval $[0,1], [1,10] \ldots, [10000, 100000]$, which is empirically an acceptable trade-off between storage, computing time and accuracy.

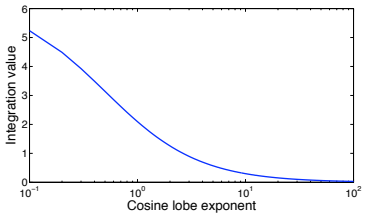


Figure 2: *Integration value according to cosine lobe exponents (logarithmic scale), with a scaling factor s set to 1. The low variation for high exponents allows a sparser sampling than with lower exponents.*

Such an approximation produces precise results compared to the actual lobe product integration: we have estimated the integration difference value between (i) the approximated lobe product in the table and (ii) the corresponding lobe product. The integration is processed using an adaptive Simpson quadrature with a tolerance set to 10^{-7}; the mean error is equal to 7.1379×10^{-4}, with a variance equal to 1.2091×10^{-8} ; the median error value is 0. Note that the maximum error (Figure 3) occurs with very low exponents e_1 and e_2, that are practically rarely used, even with Lafortune model. In addition, as explained in Section 4.3, cosine lobe exponents produced during rendering always increase.

4.2. Integration

Note that the cosine lobe integration value only depends on the exponent:

$$\int_\Omega c(\overrightarrow{v})\, d\overrightarrow{v} = \int_\Omega s\, \max(\overrightarrow{a} \cdot \overrightarrow{v}, 0)^e d\overrightarrow{v} \tag{20}$$

$$= s \int_\Omega \max(\overrightarrow{X} \cdot \overrightarrow{v}, 0)^e\, d\overrightarrow{v} \tag{21}$$

where c is an arbitrary lobe and $\overrightarrow{X}$ is any axis.

Since this integration is time-consuming, we choose to pre-compute a set of values for e in the range $[0, 100000]$ using the common adaptive Gauss quadrature. For limiting memory requirements, the table sampling strategy also relies on the cosine lobe integration variation (see Figure 2): the first samples are densely distributed while the last ones can be sparser; 128 linearly spaced samples are estimated in each interval $[0,1], [1,10] \ldots, [10000, 100000]$. At run-time, the rendering process uses linear interpolation; it is fast and the error remains low with this sampling strategy (see section 6).

4.3. Rendering

The shading of a 3D point seen from a given viewpoint is performed using only cosine lobe sums for BRDF, visibility and lighting. Equation 3 can be rewritten:

$$L_o = \int_{\Omega+} \sum_i^I \left(c_i(\overrightarrow{v})\right) \sum_j^J \left(c_j(\overrightarrow{v})\right) \sum_k^K \left(c_k(\overrightarrow{v})\right) c_{dot}(\overrightarrow{v}) d\overrightarrow{v} \tag{22}$$

where $c_{dot}(\overrightarrow{v})$ is the cosine lobe $\overrightarrow{n} \cdot \overrightarrow{v}$.

Using approximation 16, equation 22 becomes

$$L_o = \sum_m^{I \cdot J \cdot K} \int_{\Omega+} c_m(\overrightarrow{v}) d\overrightarrow{v} \tag{23}$$

Many cosine lobe products only negligibly contribute to the outgoing radiance L_o, for two main reasons: (i) when a product $c_1 \cdot c_2$ is approximated as a new cosine lobe c_r (see Equation 16), the exponent e_r is greater than e_1 and e_2 (see Figure 6); (ii) integration values of cosine lobes decrease when exponents increase. Thus, a negligible cosine lobe multiplied with another lobe is approximated by a negligible lobe.

When the product of two cosine lobe sums is expanded (see Equations 11 and 13), we propose to check the integration value of each approximated cosine lobe product. If the integration value is smaller than a given threshold t_i, the new

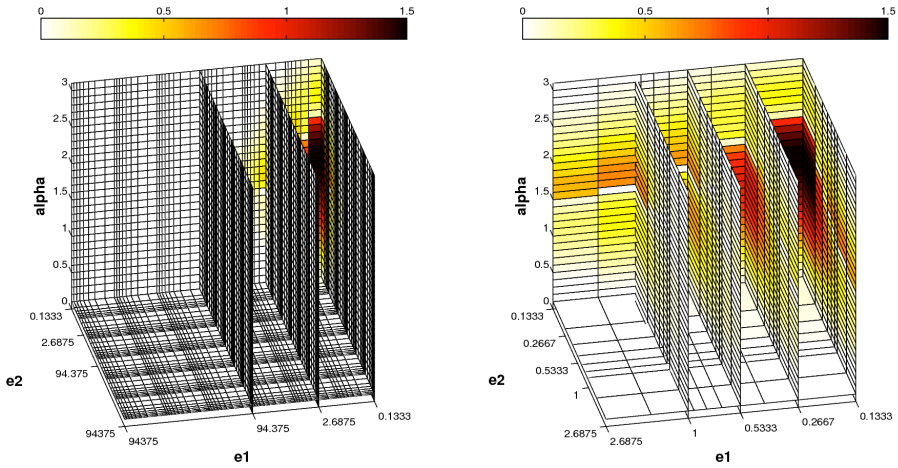


Figure 3: *Lobe product approximation (Section 4.1) error: e_1 and e_2 correspond to the lobes exponents and α is the angle between the lobes axis. The error is defined as the absolute difference between the integration of $c_1 \cdot c_2$ and c_r. The error value is represented by a color; the white regions correspond to a low error; the black regions to a high error. An important error ($\approx 50\%$) is noticeable for e_1 and e_2 between 0 and 1 approximatively and an angle α around $\pi/2$. Left: full representation for e_1 and e_2 in range $[0..100000]$; Right: zoom on the region corresponding to the highest error.*

cosine lobe can be neglected (removed from the expanded sum).

Figure 4 presents a statistical distribution of lobe products integration values (10 million of randomly generated products). This Figure shows that even a small threshold avoids many lobes computations.

In our application, a single threshold t_i is set manually by the user for all shaded points (see Figure 5); other strategies with varying thresholds depending on time constraints or the scene configuration could also be investigated.

In addition, given a product of several lobes, when the first one can be neglected, all the remaining computations can obviously be avoided. Consequently, the ordering of computations affects the number of products actually performed. Section 5.4 describes the ordering strategy we propose for our application.

5. Cosine Lobes for Direct Lighting

This section describes an application that uses our representation for direct lighting. We discuss cosine lobes construction for each term of the integral. Once lobes are defined, the computations are performed using the precomputed products and integration described above.

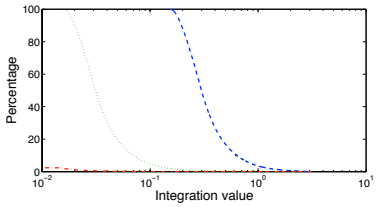


Figure 4: *Percentage of lobe products integration (ordinate) smaller than or equal to a given value. The dashed curve represents integrated products in a low frequency environment (exponents $\in [0,10]$); the dotted curve corresponds to a higher frequency environment (exponents $\in [0,100]$) and the last curve to exponents $\in [0,1000]$. In practice, exponents often grow up to 15000.*

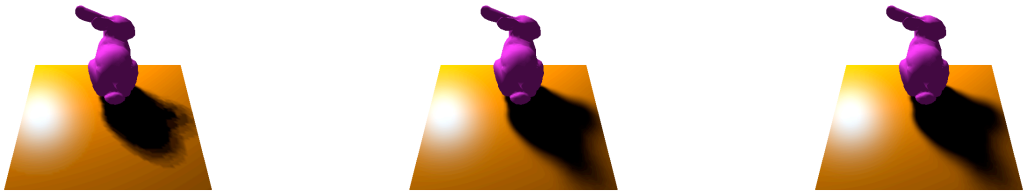
5.1. Review of Cosine Lobe-Based BRDF Models

Phong and Lafortune models [Pho75, LFTG97] rely on cosine lobes for compactly representing BRDF. This section expresses those models according to our formalism.

Phong Model

$$\tilde{f_r}(\overrightarrow{v}) = k_d(\overrightarrow{n} \cdot \overrightarrow{v}) + k_s \max(\overrightarrow{r} \cdot \overrightarrow{v}, 0)^{e_s} \quad (24)$$

$$= c_d(\overrightarrow{v}) + c_s(\overrightarrow{v}) \quad (25)$$



$t_i = 5 \times 10^{-2}$ $t_i = 1 \times 10^{-3}$ $t_i = 0$

Figure 5: *The scene is rendered using several values of t_i. Visibility is represented with a high number of thin cosinus lobes (small contributions) in penumbra, while umbra require a small number of thick lobes. Consequently, when t_i is too high, shadows become aliased.*

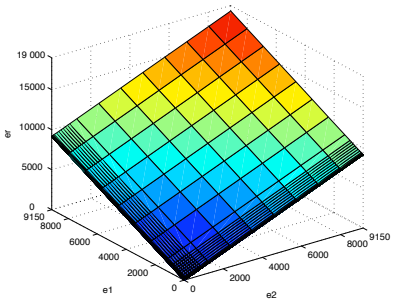


Figure 6: *Representation of e_r according to e_1 and e_2 for the approximation of the product $c_1.c_2$ for a fixed α. Note that e_r is greater or equal than e_1 and e_2 (the corresponding surface is not a plane).*

where

$$c_d(\overrightarrow{v}) = k_d \max(\overrightarrow{n} \cdot \overrightarrow{v}, 0)^1 \tag{26}$$

and

$$c_s(\overrightarrow{v}) = k_s \max(\overrightarrow{r} \cdot \overrightarrow{v}, 0)^{e_s} \tag{27}$$

k_d defines the diffuse cosine lobe and $\overrightarrow{n}$ is the surface normal. e_s is the specular cosine lobe exponent, and k_s the scale factor. $\overrightarrow{r}$ is the reflection of $\overrightarrow{v_o}$ by $\overrightarrow{n}$.

Note that the original Phong model is not physically correct, but we can as well express the *modified Phong model* [Lew94] with our representation.

Lafortune Model

$$f_r(\overrightarrow{v}) = \sum_i \rho_i \max(\overrightarrow{v}^{\mathrm{T}} M_i \overrightarrow{v_o})^{e_i} \tag{28}$$

$$= \sum_i c_i(\overrightarrow{v}) \tag{29}$$

where, according to [LFTG97], and using Helmoltz reciprocity :

$$c_i(\overrightarrow{v}) = k_i \|M_i \overrightarrow{v_o}\|^{e_i} \max\left(\frac{M_i \overrightarrow{v_o}}{\|M_i \overrightarrow{v_o}\|} \cdot \overrightarrow{v}, 0\right)^{e_i} \tag{30}$$

k_i is the scale factor of c_i, M_i represents the lobe axis in a local frame depending on material anisotropy and e_i the exponent.

5.2. Incoming Radiance

Given a spherical light source (r, p, v) defined by a radius r, a position p and a radiance value v, the incident radiance on the hemisphere centered around a given point x is delimited by a spherical cap (a cone); the cosine lobe corresponding to the incident radiance coming from this light source is:

$$c_{light}(\overrightarrow{v}) = v \max\left(\frac{p-x}{\|p-x\|} \cdot \overrightarrow{v}, 0\right)^{e_{cap}} \tag{31}$$

e_{cap} parameter corresponds the the spherical cap angle α_{cap} (see Figure 7). e_{cap} is estimated solving the following opti-

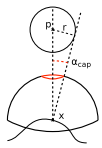


Figure 7: *α_{cap} represents the spherical cap angle corresponding to sphere (p, r).*

mization problem

$$\{e_{cap}\} = \underset{\{e_{cap}\}}{\arg\min} \int_\Omega \left(f_{cap}(\overrightarrow{v}) - c_{cap}(\overrightarrow{v})\right)^2 d\overrightarrow{v} \tag{32}$$

where f_{cap} is a function representing a spherical cap:

$$f_{cap}(\overrightarrow{v}) = \begin{cases} 1 & \text{if } \overrightarrow{v} \cdot \overrightarrow{a} \geq cos(\alpha_{cap}) \\ 0 & \text{else} \end{cases} \tag{33}$$

and

$$c_{cap}(\overrightarrow{v}) = 1\max(\overrightarrow{a}\cdot\overrightarrow{v},0)^{e_{cap}} \quad (34)$$

In practice, e_{cap} is precomputed and stored in a table containing 1000 values, and linearly interpolated at run-time.

When using several light sources, the corresponding cosine lobes may overlap. However, our visibility processing (described in the next Section) naturally handles this case.

5.3. Visibility

In the off-line process, objects are simplified by sphere sets using the methods proposed by Wang *et al.* [WZS*06] or by Bradshaw and O'Sullivan [BO02]; they have been already successfully used for visibility computations by many authors [RWS*06, SGNS07, GJW08]. These methods closely approximate the object surface with only a few spheres (for instance only 64 spheres for the Stanford bunny provide satisfactory shadow results in our application). The computed sphere sets strongly simplify objects geometry and are still deformable with articulated skeletons systems.

As for light sources, we aim at defining cosine lobes using spherical caps (cones):

$$V(\overrightarrow{v}) = 1 - f_{occ}(\overrightarrow{v}) \quad (35)$$

where $f_{occ}(\overrightarrow{v})$ is (see Equation 33 and Figure 7):

$$f_{occ}(\overrightarrow{v}) = \begin{cases} 1 & \text{if } \overrightarrow{v}\cdot\overrightarrow{a_{occ}} \geq cos(\alpha_{occ}) \\ 0 & \text{else} \end{cases} \quad (36)$$

With N spheres, visibility becomes:

$$V(\overrightarrow{v}) = \prod_{j=1}^{N}\left(1 - f_j(\overrightarrow{v})\right) \quad (37)$$

$$= 1 - \sum_{k=1}^{2^N-1} g_k(\overrightarrow{v}) \quad (38)$$

where g_k is a term corresponding to a product of spherical caps.

Unfortunately, this expression leads to the computation of 2^N terms. Replacing each spherical by only one lobe still requires 2^N lobes and introduces imprecisions that cumulate during the computation of the above product. The main reason is that one cosine lobe does not accurately approximate a spherical cap, and finally caps overlapping (that is correctly handled for visibility in Equation 38) produces errors with this representation.

To tackle these problems, we propose to hierarchically build a visibility mask, to control lobes generation (including overlapping problems) and to limit their number. One visibility mask is built for each (spherical) light source so as to precisely identify the incoming light directions and the effective blockers. This process is repeated for each point at each frame.

Given a point x, for each light source represented by a sphere (r,p), a unit square S_{unit} surrounding the corresponding cone (see Figure 8(a)) is placed at a distance $dist$, deduced from the cone angle (α_{cap}). Blockers are clipped and the remaining spheres are *splatted* onto S_{unit}. Note that S_{unit} may be placed behind the light source.

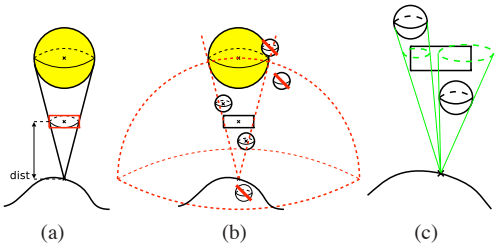


Figure 8: *Occluding spheres splatting: (a) a unit square is placed according to the spherical light source; (b) useless blockers are clipped; (c) remaining spheres are splatted onto the unit square.*

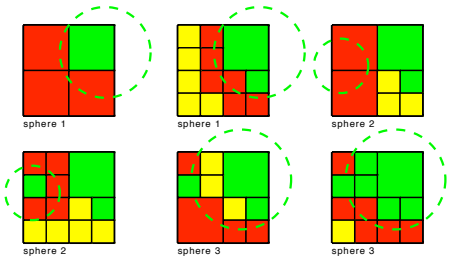


Figure 9: *Quad-tree construction example. 3 spheres are splatted and a maximal depth of 2 is used. Red leafs are subdivided, yellow leafs are not currently considered, and green leafs corresponding to shadows are frozen.*

Splatted spheres are used to build a quad-tree (see Figure 9). The quad-tree depth is practically limited to 4 or 5. We have chosen an *aggressive* approach where partially covered leaves are ignored so as to reduce the number of cosine lobes. However, the *conservative* alternative can be easily implemented, keeping the partially covered leaves (see Figure 10). A cosine lobe is built for each shadow leaf using a bounding sphere (see Figure 11).

Nearby cosine lobes mutually overlap and the error must be corrected. Therefore, we propose to associate a weight to each lobe:

$$V(\overrightarrow{v}) = c_{unit}(\overrightarrow{v}) - \sum_{i}^{N-1} w_i c_i(\overrightarrow{v}) \quad (39)$$

where

$$c_{unit}(\overrightarrow{v}) = 1.\max(\overrightarrow{n}\cdot\overrightarrow{v},0)^0 = 1 \quad (40)$$

$$c_{unit}(\overrightarrow{v})c_a(\overrightarrow{v}) = c_a(\overrightarrow{v}) \quad (41)$$

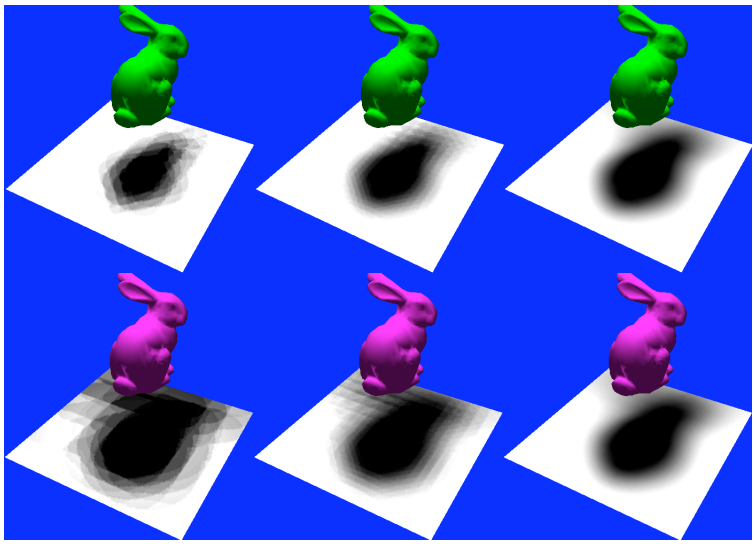


Figure 10: *Shadows artifacts when depth of visibility mask is not high enough. Top row: with aggressive visibility; bottom row: with conservative visibility. From left to right: visibility depth is equal to 2, 3 and 5.*

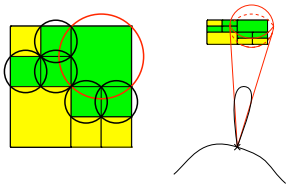


Figure 11: *Once visibility mask is built (left), one bounding sphere is constructed per shadow leaf and the associated cosine lobes are produced (right).*

and w_i depends on the distance *dist* and on the leaves depth. When *dist* varies, the cosine lobes overlap varies as well (see Figure 12). w_i values are fixed experimentally for various values of $dist \in \{0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and for each depth.

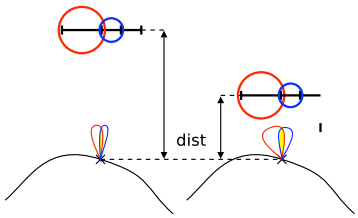


Figure 12: *The cosine lobes overlapping vary according to the distance dist, even with the same visibility mask.*

Weights can be adjusted manually once with a single scene (one plane, one object and one light source is enough) so as to obtain smooth shadows. First, weight values are initialized to 0; the first level weight is then chosen to obtain a perfect dark in the umbra area. The weight associated with each successive level is then fixed so as to avoid shadows sharpness with the previous ones. This process is performed only once and the resulting weights can be used with any scene. At run-time, weights are linearly interpolated according to *dist*.

5.4. Ordering Computations

This section describes our strategy for ordering lobe products in Equation 22, so as to reduce computing time. Let us recall that the cosine lobe c_r resulting from the product of two lobes c_1 and c_2 depends on e_1, e_2 and α (see Section 4.1). Firstly, when α increases, c_r integration value decreases. Secondly, with visibility masks, all the products between visibility lobes and incoming radiance lobes are actually useful (cannot be neglected). Thirdly, in our application (direct lighting), incoming radiance requires less cosine lobes than visibility. Thus, firstly multiplying reflectance and lighting (before visibility) produces fast results. This strategy produces results remaining close to Monte-Carlo integration with interactive rendering (Figure 13), though it does not ensure the best precision.

6. Results and Discussion

This section presents some results obtained with our application, implemented in C++. All the results have been produced with an Intel Core 2 Duo T7500 (2.2GHz) CPU and an NVidia Geforce 8400 GS.



(a) Reference (b) Our application

Figure 13: *Shadows with direct lighting for a scene composed of 5 and a plane: (a) reference path tracing image and (b) image computed using our application. Maximal depth of the visibility mask is set to 5.*

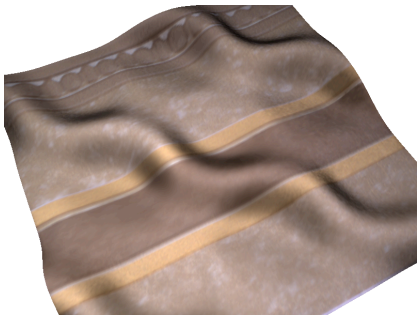


Figure 14: *Spatially varying BRDF are possible using Lafortune model. This scene is composed of 65536 shaded points, 1 spherical light source and no occluder,* 13.73 *frame per second.*

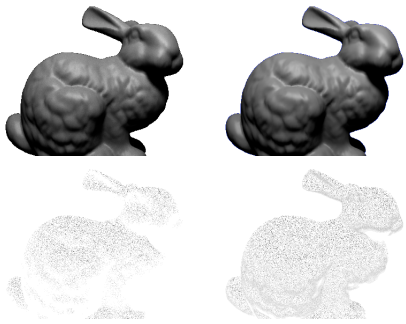


Figure 15: *Comparison between Monte-Carlo integration (top left) and our approach (top right) for the cosine lobe sums integration. Lobe sums are generated using the methods described in Section 5. The bottom images show the difference. The bottom left image represents positive values and the bottom right the negatives.*

Constructing our 3 pre-computed tables (product, integration and spherical caps) requires approximately 4 hours. Let us recall that these tables do not depend on the scene characteristics: they are computed once and for all. The memory required for their storage is less than 10 MB. For the sake of clarity, the following results only corresponds to cosine lobes representation, without any optimization: for each frame, the whole image is completely computed (no caching technique is employed, nor spatial or temporal coherency). Our application consists in shading a selected set of 3D points (either triangle vertices or estimated per pixel, employing a G-buffer). Shading is performed on the CPU and rasterization uses OpenGL.

Due to the flexibility and adaptivity of cosine lobes representation (number of lobes, bandwith, free distribution, *etc.*), computing efforts can be concentrated on regions where lighting needs to be detailed. For instance, penumbra areas requires a higher number of lobes for representing visibility than umbra areas or fully-lit regions, where only a few computations can produce precise results. Consequently, our application performances vary according to chosen materials (Figure 14), number of shaded points, maximum depth of visibility masks and chosen threshold of cosine lobe products (Figures 16 and 5) or relative position between occluders and lights. Frame rates are given in the caption of each figure.

Presently, many approaches dedicated to dynamic scenes are based on SH. Generally speaking such orthogonal basis functions are adapted to integrations. However, the projection on these basis functions is costly. For instance, on Figure 16, umbra and lit regions would have required the same amount of computation and are restricted to low-frequency effects, contrary to our cosine-lobe based approach.

Once cosine lobe sums are defined, the approach given in section 4 approximates lobe products and uses tables for reducing computing time. Figure 15 shows that these approximations lead to a small difference, compared to Monte-Carlo integration. For the Monte-Carlo reference image, the maximum radiance value is 0.701961, the mean radiance value is 0.127906; with our approach, the maximum difference is 0.054902 while the mean difference is 0.003884. Note that our method introduces a bias: regions located in shadows, where more cosine lobes products are computed, exhibit more negative difference values.

We also compare the images produced by our application with path tracing for direct lighting and soft shadows. As shown in Figure 13, the results are hardly distinguishable.

Table 6 presents the computing time ratio corresponding to each rendering part, for different types of scenes, with hard and soft shadows, complex or simple material and various numbers of occluders. The last row corresponds to a scene that does not contain any occluder, implying no visibility computations.The main part of computing time is dedicated to visibility mask construction, as for any other blocker accumulation method.

Shadows are considered by many authors as a key is-

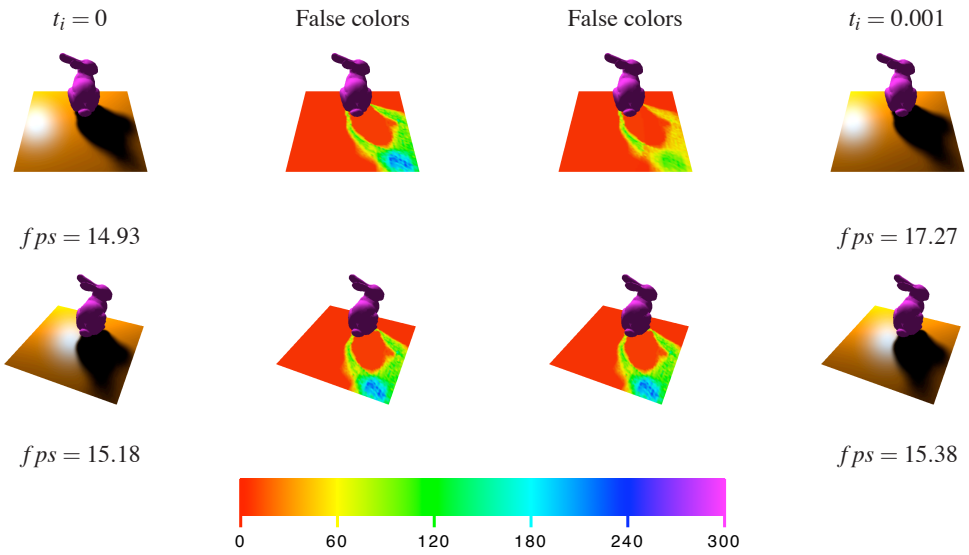


Figure 16: *Performance gains obtained according to the threshold t_i (Section 4.3). The scene is composed of 4096 shaded point, one spherical light source and 64 spherical occluders (the bunny). The threshold is chosen so as to keep indistinguishable the difference between resulting images (left and right columns). The false-color images illustrate the number of cosine lobe products computed (scale is given at the bottom). The specular highlight properly interacts with the shadows, our method naturally allows to avoid many computations due to the threshold. Note that frame rates increase when the highlight moves away from the shadow. We can see a reduction of about 50% of the number of product computations (corresponding to 2 frames per second in practice, because of visibility mask construction cost).*

Figure	a	b	c	d	e
14	29.6	15.5	0	0	54.9
18	0.3	0.2	82.7	6.1	10.7
17(a)	2.6	2.7	76.2	8.1	10.4
17(b)	1.5	1.5	63	15.8	18.2
17(c)	0.7	0.7	54	20.3	24.3

Table 1: *CPU time (percent) for the shading loop: (a) generation of cosine lobe sums for reflectance, (b) lighting, (c) visibility mask construction, (d) visibility mask lobes, (e) rendering equation integration computation.*

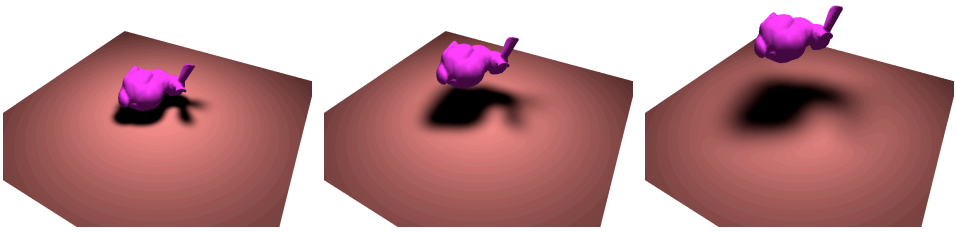
sue for realistic rendering; their processing is often handled specifically. Figures 13 and 17 show that our method naturally produces realistic direct lighting (including hard and soft shadows), without introducing specific processing. Our application can be used with any type of material, using a spatially varying Lafortune BRDF model (see Figure 14). We do not introduce any occluder in this scene for highlighting the low cost for representing reflectance and lighting with cosine lobe sums. Figure 18 illustrates a difficult case for the construction of the visibility mask. Self-shadowing is robustly supported with adapted sphere sets.

7. Conclusion

In this paper, we present a new method for solving the rendering equation using approximated cosine lobe products. We propose an interactive application for direct lighting including all-frequency lighting / materials, soft shadows and dynamic scenes. As shown in the results, cosine lobes as basis functions are useful for properly targeting relevant hemispherical directions (flexibility) in terms of BRDF, incoming radiance or visibility. In addition, their representation only requires a few parameters (compactness).

Shading is only performed on the CPU and performances can be further improved using a full GPU implementation, adapting for instance the streaming reduction algorithm proposed in [RAH07] for avoiding useless computations concerning negligible lobe products. Furthermore, our application demonstrates the feasibility of rapidly computing direct lighting, without using any spatial and temporal coherency. Visibility masks are often identical for close neighborhoods of shaded points; this coherency can be used for reducing the computing time.

Global illumination in dynamic scenes has been addressed using spherical proxies and spherical harmonics [SGNS07, GJW08]. Similarly to these authors, our direct lighting application makes use of blocker accumulation. In the future,



(a) 15.9 fps (b) 10.1 fps (c) 4.75 fps

Figure 17: *Shadow smoothing according to the distance between plane and the Bunny. Frame rates decrease with respect to the (increasing) shadowed area. Penumbra implies complex visibility masks and more cosine lobes. 16384 points are shaded considering a strictly diffuse material, 1 spherical light source, 64 spherical occluders, fixing the maximal depth of the visibility mask to 5 and the threshold to* 0.001 *(Section 4.3).*

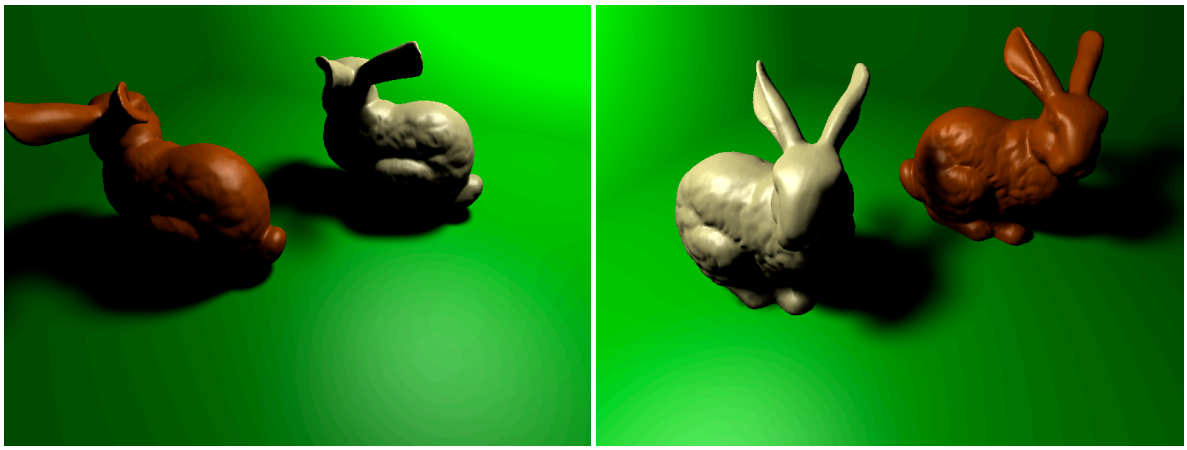


Figure 18: *Two chocolate-flavoured bunnies in a green box. These pictures take 6.25s for rendering. 480000 points are shaded, using 1024 spherical occluders, 2 cosine lobes for the reflectance models and 1 spherical light source. We choose 5 for the maximum depth of the visibility mask.*

we aim at adapting their interesting simplifications to cosine lobes for overcoming the usual SH drawbacks.

Finally, cosine lobes can also be used in various SRBF applications, such as the representation of high frequency environment maps [TS06], anisotropic materials [LFTG97], or real-time environment map rendering applications, such as Green *et al*. [GKD07] method.

References

[BO02] BRADSHAW G., O'SULLIVAN C.: Sphere-tree construction using dynamic medial axis approximation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 33–40.

[Gau06] GAUTRON P.: *Cache de luminance et cartes graphiques : une approche pour la simulation d'éclairage temps réel dans des scènes animées*. PhD thesis, Université de Rennes 1, 2006.

[GJW08] GUERRERO P., JESCHKE S., WIMMER M.: Real-time indirect illumination and soft shadows in dynamic scenes using spherical lights. *Computer Graphics Forum 27*, 8 (2008), 2154–2168.

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

[GKMD06] GREEN P., KAUTZ J., MATUSIK W., DURAND F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006), pp. 7–14.

[KK07] KOZLOWSKI O., KAUTZ J.: Is accurate occlusion of glossy reflections necessary? In *APGV '07: Pro-*

ceedings of the 4th symposium on Applied perception in graphics and visualization (2007), pp. 91–98.

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

[Lew94] LEWIS R. R.: Making shaders more physically plausible. *Computer Graphics Forum (Eurographics '94 Conference Issue) 13*, 3 (1994), 1–13.

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 117–126.

[MLH02] MCALLISTER D. K., LASTRA A., HEIDRICH W.: Efficient rendering of spatial bi-directional reflectance distribution functions. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2002), pp. 79–88.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (2003), pp. 376–381.

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., 2004.

[Pho75] PHONG B. T.: Illumination for computer generated pictures. vol. 18, pp. 311–317.

[RAH07] ROGER D., ASSARSSON U., HOLZSCHUCH N.: Efficient stream reduction on the gpu. In *Workshop on General Purpose Processing on Graphics Processing Units* (2007).

[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.

[SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), pp. 97–105.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (2003), pp. 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 527–536.

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), pp. 967–976.

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), pp. 265–272.

[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.

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), vol. 24, pp. 1196–1201.