Spherical Piecewise Constant Basis Functions for All-Frequency Precomputed Radiance Transfer

Kun Xu, Yun-Tao Jia, Hongbo Fu,
Shi-Min Hu, *Member*, *IEEE Computer Society*, and Chiew-Lan Tai

Abstract—This paper presents a novel basis function, called *spherical piecewise constant basis function* (SPCBF), for precomputed radiance transfer. SPCBFs have several desirable properties: rotatability, ability to represent all-frequency signals, and support for efficient multiple product. By smartly partitioning the illumination sphere into a set of subregions and associating each subregion with an SPCBF valued 1 inside the region and 0 elsewhere, we precompute the light coefficients using the resulting SPCBFs. Efficient rotation of the light representation in SPCBFs is achieved by rotating the domain of SPCBFs. During runtime rendering, we approximate the BRDF and visibility coefficients using the set of SPCBFs for light, possibly rotated, through fast lookup of *summed-area table* (SAT) and *visibility distance table* (VDT), respectively. SPCBFs enable new effects such as object rotation in all-frequency rendering of dynamic scenes and on-the-fly BRDF editing under rotating environment lighting. With graphics hardware acceleration, our method achieves real-time frame rates.

Index Terms—Real-time rendering, precomputed radiance transfer, spherical piecewise constant basis functions.

1 Introduction

Real-time realistic global illumination for static or dynamic scenes under dynamic environment lighting is a challenging problem. The difficulty lies in fast computation of per-vertex integration of lighting functions (represented by irradiance environment maps), BRDF, and (self and/or occluder) visibility functions over the hemisphere of lighting directions. The precomputed radiance transfer (PRT) technique [1] and its variants [2], [3], [4], [5] have demonstrated their great ability in real-time rendering of complex scenes under dynamic environment lighting. They employ different kinds of basis functions to approximately represent light, BRDF, and visibility functions, thus making precomputation memory affordable and simplifying the expensive rendering integrals to simple and fast dot/multiple products.

There are three desirable properties that basis functions for PRT should possess. First, the basis functions should be able to effectively approximate all-frequency signals, providing all-frequency shadowing effects using only a low-order basis. Second, they should support efficient rotation, which has different significance for light, BRDF, and occluder-visibility functions. Efficient rotation of the light representation under a basis enables the efficient rotation of environment lighting. Efficient rotation of the BRDF representation enables separate storage of BRDF from a model (instead of storing one BRDF per vertex) and efficient rotation of BRDF from the local frame at each vertex to the global frame, thus supporting on-the-fly BRDF editing. Efficient rotation of the occluder-visibility representation enables the occluder-visibility function to be efficiently rotated from the local frame at an object to the global frame, thereby allowing rotatable objects in dynamic scenes. Third, they should support efficient multiple product. This property is crucial for dynamic scene rendering, which involves multiple products between light, BRDF, self-visibility, and occluder-visibility functions.

Several kinds of basis functions have been proposed for PRT in past years such as spherical harmonics (SH) [1], wavelet [2], [3], spherical radial basis functions (SRBFs) [5]. However, each one of them lacks certain desirable properties, as summarized in Table 1 (see more details in Section 2).

In this paper, we present a novel spherical basis representation for PRT, called *spherical piecewise constant basis functions* (SPCBFs), which possess all the abovementioned three desirable properties (Section 3). The key idea is stated as follows: We first partition a unit sphere on which light, BRDF, and visibility functions are all defined into a common set of subregions and associate each subregion with an SPCBF valued 1 within the subregion and 0 outside it. The resulting SPCBFs naturally form an orthogonal basis. Then, we approximate the integrands in the rendering integral with piecewise constant functions, which are all represented in the same SPCBF basis. The apparent advantage of representing the integrands with the SPCBFs

- *K. Xu and S.-M. Hu are with the Department of Computer Science and Technology, Tsinghua University, Beijing, P.R. China, P.C. 100084. E-mail: xu-k@mails.tsinghua.ed.cn, shimin@tsinghua.edu.cn.*
- *Y.-T. Jia is with the University of Illinois at Urbana-Champaign, Room 3305, Siebel Center, 201 N. Goodwin, Urbana, IL 61801. E-mail: yjia3@uiuc.edu.*
- *H. Fu is with the Department of Computer Science, Hong Kong University of Science and Technology, UBC, 201 2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada. E-mail: fuplus@gmail.com.*
- *C.-L. Tai is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: taicl@cs.ust.hk.*

Manuscript received 23 Dec. 2006; revised 2 June 2007; accepted 23 Sept. 2007; published online 4 Oct. 2007.
Recommended for acceptance by H.-P. Seidel.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0230-1206.
Digital Object Identifier no. 10.1109/TVCG.2007.70442.

TABLE 1
Properties of SPCBF Compared to Previous Basis Functions

	SH	Wavelet	SRBF	SPCBF
All-frequency rendering	×	√	√	√
Rotation	√	×	√	√
Multiple-product	√	√	×	√

is that, the rendering integral can then be approximated by efficient multiple products between the individual coefficients of the integrands under the basis.

SPCBFs can represent all-frequency signals. However, not all arbitrary SPCBFs are suitable for all-frequency PRT rendering. As the environment light is considered distant and thus is the same for every object point in the scene, we choose to use the environment map to define the partition of the unit sphere (that is, the domain of the SPCBFs with which the BRDF and visibility functions are also represented). Inspired by importance sampling [6], we partition the sphere according to the light energy; specifically, regions with high light intensity are partitioned into small subregions. Such a deliberate partitioning guarantees that the resulting SPCBFs are able to represent all-frequency signals of a given environment map, and yet, the representation is compact.

Rotating functions in SPCBFs representation can be easily done by rotating the domain of SPCBFs. However, as the BRDF and visibility functions are represented with the same set of SPCBFs as those defined over the environment map, when the light is rotated, the coefficients of BRDF and visibility functions in SPCBFs representation need to be recomputed over the rotated SPCBFs. Recomputation of these coefficients is very time consuming. We propose to precompute summed-area table (SAT) and visibility distance tables (VDTs) for BRDF and visibility integration, respectively. At runtime rendering, we approximate the coefficients of BRDF and visibility by fast lookup of SAT and VDTs, respectively. With this strategy of precomputing the light coefficients and runtime approximating the BRDF and visibility coefficients, our method supports efficient rotation of light, BRDF, and visibility.

Our rendering algorithm involves two approximations. Roughly speaking, we first approximate the environment map using disjoint area light sources of constant intensity. We then use SAT/VDT to approximate the integral of BRDF/Visibility with respect to each area light source. Although the individual errors introduced by the BRDF and visibility representations are possibly large, they will be significantly weakened by the small light coefficient, thanks to our importance-sampling-like partitioning of the environment map. Thus, our method always produces compelling rendering results.

In summary, our main contributions consist of the following:

- A novel basis for PRT that possesses rotatability, the ability to represent all-frequency signals, and support for efficient multiple product.
- A real-time PRT rendering framework that supports new effects, in particular
  - all-frequency rendering of dynamic scenes involving both object rotation and translation, and
  - on-the-fly BRDF editing under rotating environment lighting,

  and is capable of incorporating existing challenging effects such as
  - local light illumination, and
  - local deformable shading.

2 Related Work

In this section, we first discuss the existing PRT techniques for static and dynamic scenes and, then, give a brief review of SAT.

2.1 Precomputed Radiance Transfer for Static Scenes

PRT framework for environment light rendering was first proposed by Sloan et al. [1]. The rationale of PRT is to represent the environment light and the light transport function with a certain linear basis, thus making precomputation storage affordable and, meanwhile, approximating the computationally expensive rendering integral at each vertex of a scene with a simple dot product of the coefficients of the basis.

SH is the first type of basis used in PRT rendering. It has several attractive properties such as orthonormality, rotational-invariance projection, and support of efficient multiple products. The resulting PRT frameworks [1], [7], [8] are effective in real-time rendering of static scenes under dynamic low-frequency environment maps. However, as SH cannot effectively approximate high-frequency signals, these frameworks can only handle low-frequency shadowing effects and low-frequency materials (BRDF).

Wavelet bases can encode functions at all frequencies in a compact way. Ng et al. [2] propose a nonlinear wavelet lighting approximation technique to perform all-frequency PRT rendering of glossy objects with fixed views or diffuse objects. Using an efficient triple product wavelet algorithm [3] or BRDF factorization [9], [10], wavelet-based PRT frameworks allow high-resolution lighting effects with changing views. However, unlike SH, wavelet representations cannot be easily rotated, making efficient rotation of light, BRDF, and visibility difficult. For example, the rotation of an environment map requires reprojection of the environment map to the wavelet basis, which may cause flickering artifact. To address the problem, Wang et al. [11] give a computational wavelet rotation method by precomputing rotation matrices. However, due to large data storage, their method can only sample rotation matrices at 2D normal directions, which is insufficient for the 3D rotation space and thus does not support arbitrary wavelet rotation.

Tsai and Shih [5] propose to use SRBFs for PRT. They find a compact set of SRBFs to represent for high-frequency signals of environment light through an optimization process. Similar to the rotation of functions in our SPCBFs representation, rotating the functions represented with SRBFs is achieved by rotating the SRBFs themselves. Unlike ours, their method employs different sets of SRBFs for the

light, BRDF, and visibility functions and thus does not need to change the SRBF representations of BRDF and visibility functions when the light rotates. However, a combined set of SRBFs is generally not orthogonal. To our best knowledge, the product of triple or multiple functions represented with nonorthogonal SRBFs cannot be efficiently calculated, limiting their method to only static scenes.

Green et al. [12] present a real-time method with complex view-dependent effects under all-frequency environment lighting. They approximate the light transport function as a summation of Gaussian functions, which leads to a fast convolution with the lighting function at render time. Their method models high-frequency specular effects well, but it is unclear how their method can produce high-frequency shadows.

Recently, Xu et al. [13] propose a real-time homogenous translucent material editing method. They utilize a 1D piecewise polynomial basis to approximate the multiple scattering diffusion reflectance function and the single scattering exponential attenuation function. However, their focus is on translucent materials and the method is limited to approximating 1D curves.

2.2 PRT for Dynamic Scenes

PRT rendering for dynamic scenes is a hard problem because the movement of objects invalidates the precomputed light transport function. Mei et al. [14] present a real-time rendering method for dynamic glossy objects under all-frequency environment lighting. Their method is based on the precomputation of shadow maps under the assumption that the illuminants are distant.

Zhou et al. [4] introduce a shadow field framework for rendering dynamic scenes with both distant illuminants and local light sources. Each occluder's shadowing effects are precomputed and stored in the SH or wavelet basis at the sample points of the occluder's surrounding space. The irradiance map of each local light is precomputed in a similar way. Their approach enables real-time low-frequency shadowing effects (if using the SH basis) and interactive all-frequency shadowing effects of dynamics scenes (if using the wavelet basis). Because the occluder visibility vector in SH can be efficiently rotated to global frame due to the rotatability of SH, their approach also supports rotation of objects in low-frequency lighting environments. Sun and Mukherjee [15] extend the dot or triple product to a generalized multifunction product in the wavelet domain. They also propose a just-in-time radiance transfer (JRT) technique to accelerate shadow computation. Their method renders all-frequency shadows of dynamic scenes in real time. As the wavelet representation cannot be easily rotated, none of the above techniques can handle real-time rendering of rotating objects in all-frequency lighting environments.

The amount of PRT data sets can be extremely large and thus infeasible to store, especially for dynamic scene rendering. Exploiting intervertex data coherence, several compression techniques have been proposed [16] such as clustered principle component analysis (CPCA) [7] and clustered tensor approximation (CTA) [5]. We adopt CPCA to compress the PRT data sets.

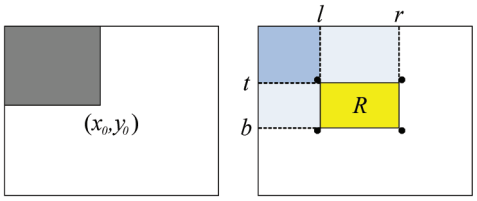


Fig. 1. An illustration of SAT definition and usage.

2.3 Summed-Area Table

Crow [17] introduces the *SAT* for rapid box filtering (averaging) in texture mapping. Its usage is later extended to volume rendering, image and video processing (see [18] and references therein). As illustrated in Fig. 1, by preintegrating the top-left rectangular area corresponding to each sample point, the integral in a rectangle area R defined by (l, r, t, b) can be rapidly computed with four lookups:

$$\int_R f(x,y)dxdy = SAT(r,b) - SAT(l,b) - SAT(r,t) + SAT(l,t). \quad (1)$$

When SAT is implemented in graphics hardware, a precision problem occurs, which can be alleviated with techniques proposed in [18].

The remainder of this paper is organized as follows: We introduce the definition of SPCBFs and their properties in Section 3. We give the overview of the proposed PRT rendering framework with SPCBFs in Section 4 and describe the algorithm details in Section 5. After presenting the implementation details in Section 6, we analyze the rendering errors of our method in Section 7. Results are given in Section 8 and conclusion and future work in Section 9.

3 Spherical Piecewise Constant Basis Functions

In this section, we state the definition of *SPCBF*s and discuss their general properties. The PRT framework using SPCBFs is introduced in the next section.

3.1 Definition

Given a *partition* $\{S_1, S_2, \ldots, S_n\}$ of a unit sphere S (that is, $S = S_1 \cup S_2 \cup \cdots \cup S_n$, and $S_i \cap S_j = \emptyset$ for any $i \neq j$), for each subregion S_i, we define a spherical function valued 1 inside S_i and 0 elsewhere as an SPCBF $B_i(\omega)$. The set $\{B_i(\omega)\}$ forms an orthogonal basis for a function space defined over the sphere. Any function $F(\omega)$ in this function space is called a *spherical piecewise constant function* (SPCF) and thus can be represented as a linear combination of the basis functions $\{B_i(\omega)\}$:

$$F(\omega) = \sum_i c_i B_i(\omega),$$

where c_i is the coefficient corresponding to the ith basis function $B_i(\omega)$.

3.2 Projection and Reconstruction

Due to the orthogonality of $\{B_i(\omega)\}$, a scalar function $G(\omega)$ defined over the sphere S can be projected into its coefficients via an integral of $G(\omega)$ over each subregion S_i:

$$c_i = \frac{1}{|S_i|}\int_S G(\omega)B_i(\omega)d\omega = \frac{1}{|S_i|}\int_{S_i} G(\omega)d\omega, \qquad (2)$$

where $|S_i|$ denotes the solid angle of S_i.

An approximation of $G(\omega)$ reconstructed from the coefficients can be formulated as

$$G(\omega) \approx \sum_i c_i B_i(\omega),$$

which approximates the spherical function well by designing a partition of the sphere specific to $G(\omega)$ (that is, defining a set of SPCBFs $\{B_i(\omega)\}$ specific to $G(\omega)$).

3.3 Properties

All-frequency. Given a spherical function to be encoded, a partition of the sphere S (that is, the locations and shapes of the subregions) can be optimized to fit the function well. The spatial localization property of SPCBFs allows both high-frequency and low-frequency signals to be represented effectively using only a small number of coefficients.

Rotation. SPCBFs support efficient rotation. Rotating functions represented with SPCBFs can be equivalently done by rotating the partitioned subregions. Formally, given a spherical function in SPCBF representation $G(\omega) \approx \sum_i c_i B_i(\omega)$, the reconstruction function $G'(\omega)$ rotated by R can be represented by

$$R(G(\omega)) \approx R\left(\sum_i c_i B_i(\omega)\right) = \sum_i c_i R(B_i(\omega)) = \sum_i c_i B'_i(\omega),$$

where $\{B'_i(\omega)\}$ are the SPCBFs defined over the partition rotated by R. Once the SPCBFs are rotated, we propose to use the precomputed SAT/VDT, making the reprojection of BRDF/visibility to the rotated SPCBFs efficient.

Multiple product. Unlike the SH or wavelet basis, which needs different methods to compute dot product and triple/multiple product, SPCBFs support a uniform scheme to compute them. This is because SPCBFs not only are orthogonal, implying $\int B_i(\omega)B_j(\omega)d\omega = 0$, for any $i \neq j$, but also satisfy $\int B_{i_1}(\omega)B_{i_2}(\omega)\cdots B_{i_l}(\omega)d\omega = 0$, $l \geq 3$, when there exists $i_j \neq i_k$.

Formally, given a set of spherical functions $G_1(\omega), G_2(\omega), \ldots, G_m(\omega)$, $m \geq 2$ and their representations in the same set of SPCBFs $\{B_i(\omega)\}$:

$$G_j(\omega) \approx \sum_i c_{j,i}B_i(\omega), \quad 1 \leq j \leq m,$$

the integral of the function products can be approximated by

$$\int G_1(\omega)G_2(\omega)\ldots G_m(\omega)d\omega \approx \sum_i |S_i|c_{1,i}c_{2,i}\ldots c_{m,i}. \qquad (3)$$

In other words, the integration of the product of multiple functions reduces to a multiple product of their coefficients.

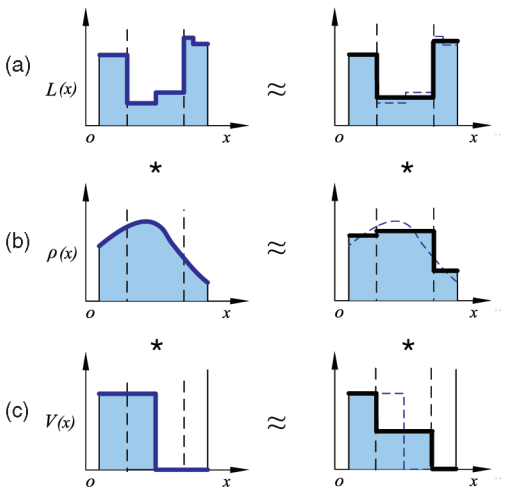


Fig. 2. SPCBF approximation of (a) light, (b) BRDF, and (c) visibility functions. The left column shows the original functions and the right column shows the corresponding SPCBF representations.

4 Overview

Without considering interreflection, the rendering equation for environment map illumination of static scenes [19] is formulated as

$$E(x,\omega_0) = \int_S L(\omega)\rho(x,\omega,\omega_0)V(x,\omega)d\omega,$$

where $E(x,\omega_0)$ is the outgoing radiance at a point x of the scene to be illuminated in direction ω_0, $L(\omega)$ is the incident radiance in direction ω, and ρ and V denote the 4D BRDF function and the self-visibility function at x, respectively. By incorporating occluder visibility, Zhou et al. [4] present a shadow field framework for rendering of dynamic scenes. The new rendering equation becomes

$$E(x,\omega_0) = \int_S L(\omega)\rho(x,\omega,\omega_0)V(x,\omega)\prod_j V_{O_j}(x,\omega)d\omega, \qquad (4)$$

where V_{O_j} is the occluder-visibility function for an opaque object j at location x.

4.1 PRT Representation in SPCBFs

We approximate the light, BRDF, self-visibility and occluder-visibility functions using the same set of SPCBFs. Because the environment light is distant and, thus, the same for all vertices of a scene, we define the partition $\{S_i\}$ of the sphere S and, thus, SPCBFs $\{B_i(\omega)\}$, according to the environment map. With this basis, the integrands in (4) can be approximated as follows (see a 1D illustration in Fig. 2):

$$L(\omega) \approx \sum_i l_i B_i(\omega),$$
$$\rho(x,\omega,\omega_o) \approx \sum_i \rho_{x,\omega_o,i}B_i(\omega),$$
$$V(x,\omega) \approx \sum_i v_{x,i}B_i(\omega),$$
$$V_{O_j}(x,\omega) \approx \sum_i v_{O_j,x,i}B_i(\omega),$$

where l_i, $\rho_{x,\omega_o,i}$, $v_{x,i}$, and $v_{O_j,x,i}$ are the coefficients of the light, BRDF, self-visibility and occluder-visibility functions

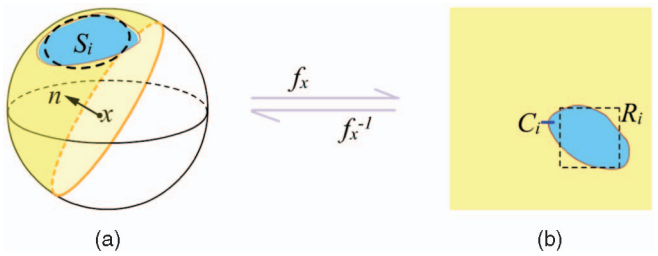


Fig. 3. (a) We define a mapping f_x from the hemisphere determined by the local frame at a vertex x to the 2D parameter plane. (b) Then, we approximate a subregion S_i by the *preimage* of an axis-aligned rectangular region R_i in the parameter domain under f_x.

in SPCBFs, respectively. Using the multiple product integration equation of SPCBFs (3), the rendering integration for dynamic scenes in (4) reduces to a multiple product between light, BRDF, self-visibility and occluder visibility coefficients:

$$E(x,\omega_0) \approx \sum_i |S_i| \left(l_i \rho_{x,\omega_o,i} v_{x,i} \prod_j v_{O_j,x,i} \right). \quad (5)$$

4.2 Precomputation

When the environment map needs rotates, we correspondingly rotate the partition, implying that the light coefficients l_i remain unchanged (see the rotation property of SPCBFs in Section 3). However, as the coefficients l_i, $\rho_{x,\omega_o,i}$, $v_{x,i}$ and $v_{O_j,x,i}$ are defined over the same partition of the environment map, the BRDF and visibility coefficients have to be recomputed when the SPCBFs are rotated caused by the rotation of the partition. The recomputation of these coefficients is very time consuming, requiring us to seek a precomputation method. Naive precomputation of these coefficients for all the possible rotations of the partition is infeasible due to the unwieldy size of the data sets.

Computing the coefficient with respect to $B_i(\omega)$ for a specific function is equivalent to integrating that function over S_i (2). Therefore, to make the precomputation storage affordable, we employ SAT for BRDF preintegration and employ *VDT* for visibility preintegration. The efficient rotatability of SPCBFs allows us to precompute VDT in the local frame at each vertex and a global BRDF SAT, which leads to more compact storage.

Rather than directly store the coefficients, SAT and VDT employ lookups to compute them. To fast evaluate the coefficients by simple lookups (1), we parameterize S_i and fit the *preimage* of S_i in the parameter domain, C_i, with an axis-aligned rectangle R_i (Fig. 3). Let $f_x : \omega \rightarrow p$ be a mapping from the hemisphere determined by the *local* frame at a given object vertex x to the 2D parameter plane (Fig. 3), where ω is a 3D direction in the *global* frame, and p is a 2D point in the parameter domain. Since the partition allows rotations of any angle to reduce the fitting error between R_i and $C_i = f_x(S_i)$, intuitively, we prefer S_i and R_i of aspect ratio equal to 1. Therefore, we use an axis-aligned *square* R_i to compute the coefficients corresponding to S_i. Let c_i be the centroid of S_i. For a given object vertex x, R_i is then identified as an axis-aligned square $R_x(c_i, r_i)$, whose respective center and size are $f_x(c_i)$ and $\sqrt{2}r_i$. Note that we use a unique pair of parameters $\langle c_i, r_i \rangle$ to represent each subregion S_i.

In summary, the tasks done in the precomputation stage are listed as follows:

1. Given an environment map, we simultaneously compute the partition of the illumination sphere and the parameters $\langle c_i, r_i \rangle$ for each subregion S_i through a bottom-up optimization algorithm to make two sources of errors tractable (Section 5.1). The light coefficients l_i are also computed using (2) during the optimization.
2. We precompute VDT for visibility (Section 5.3) in the local frame at each vertex and a global SAT for BRDF (Section 5.2).

4.3 Runtime Rendering

During runtime rendering, the following steps are performed:

1. When the environment light rotates, we keep r_i and l_i unchanged and rotate c_i according to the rotation transformation of the environment light. Let c_i' be the resulting direction by rotating c_i.
2. At each vertex x, we compute $\rho_{x,\omega_o,i}$ for each subregion S_i by fast lookup of SAT (Section 5.2) and compute $v_{x,i}$ and $v_{O_j,x,i}$ by fast lookup of VDT (Section 5.3) using the axis-aligned square $R_x(c_i', r_i)$.[1]
3. We sum the multiple products of l_i, $\rho_{x,\omega_o,i}$, $v_{x,i}$, and $v_{O_j,x,i}$ over all the subregions to approximate the outgoing radiance $B(x,\omega_0)$ at point x in direction ω_0 (5).

5 Algorithm

We give the details of the SPCBF-based PRT rendering algorithm in this section. We first describe an approach to partition an environment map (Section 5.1). We then explain how to precompute and runtime look up SAT for BRDF (Section 5.2) and VDT for visibility (Section 5.3).

5.1 Environment Map Partitioning for SPCBF Construction

Our PRT rendering algorithm involves two approximation steps: The approximation of the rendering integral by a multiple product of the integrands in SPCBFs and the approximate representation of each subregion S_i with $\langle c_i, r_i \rangle$. The approximation errors of both steps are highly dependent on the partition of the environment map. In this section, we first define two metrics (that is, ξ_i and η_i) to measure these two sources of approximation errors for a given subregion S_i. We then present a bottom-up algorithm to partition the environment map under the guidance of these error metrics.

According to the definition of SPCBFs, we associate each subregion S_i with an SPCBF $B_i(\omega)$. The light coefficient corresponding to $B_i(\omega)$ is then computed as $l_i = \frac{1}{|S_i|} \int_{S_i} L(\omega) d\omega$. We use the variance σ_i^2 of the light intensity within S_i to measure the approximation error of

1. The coefficients computed by looking up SAT or VDT are essentially only the approximate coefficients under the original set of SPCBFs.

representing the light function with an SPCF $l_i B_i(\omega)$ within S_i.

Since the BRDF and visibility functions are dependent on rotation of the partition, and thus are not fixed with respect to S_i, we cannot give an exact representation to measure the approximation errors of BRDF and visibility caused by SPCBF representations. Fortunately, what we really care for is largely the expected approximation error of representing the rendering integral with a multiple product of the integrands in SPCBFs, rather than the representation error of each integrand. Clearly, to reduce the expected approximation error, we need to restrict subregions with high-intensity light to small areas while allowing subregions with low-intensity light to have large areas, that is, we prefer small values of $|S_i|l_i$. Therefore, we use the following metric to measure the expected approximation error of representing the rendering integral with a multiple product of the coefficients in SPCBFs:

$$\xi_i = |S_i|\ l_i\ \sigma_i^2.$$

In a sense, our strategy is similar to the strategy of sampling environment map based on importance proposed by Agarwal et al. [6].

Now, we explain how to define a metric η_i to measure the fitting error of representing the shape of the subregion S_i with $\langle c_i, r_i\rangle$. We set c_i as the centroid of S_i and aim to find an optimal value of r_i that results in the minimum fitting error. Let $U_i = \cup_{x\in\Omega} f_x^{-1}(R_x(c_i, r_i))$ denote the union of the *preimages* of the axis-aligned squares $R_x(c_i, r_i)$ at every position x of a model surface Ω under the mapping f_x. Intuitively, U_i is the circular region covered when rotating a rectangle-like shape around the center c_i. For a direction ω within U_i, the probability that ω is covered by $f_x^{-1}(R_x(c_i, r_i))$ at a certain position x, denoted by $\alpha_i(\omega)$, is different. Specifically, the probability $\alpha_i(\omega)$ is only related to the distance between ω and c_i and can be approximated as

$$\alpha_i(\omega) \approx \begin{cases} 1, & ratio(\omega) < ratio_0, \\ \frac{ratio(\omega)-ratio_0}{ratio_1-ratio_0}, & ratio_0 < ratio(\omega) < ratio_1, \\ 0, & ratio(\omega) > ratio_1, \end{cases}$$

where $ratio(\omega) = |\omega - c_i|/r_i$, and $ratio_0$ and $ratio_1$ are the minimum and maximum values of $ratio(\omega)$ among all $x \in U_i$, respectively.[2] Due to the different covering probability $\alpha_i(\omega)$ within U_i, using only the boundaries of S_i and U_i are not sufficient to define the approximation error η_i of representing S_i with $\langle c_i, r_i\rangle$. Therefore, we propose to define η_i by minimizing the area difference between U_i and S_i weighted by the covering probability $\alpha_i(\omega)$ in a least-squares sense:

$$\eta_i = \min_{r_i} \int_{\overline{S}_i} \alpha_i(\omega)\ (T_i(\omega) - 1)^2 + (1 - \alpha_i(\omega))\ T_i^2(\omega)\ d\omega,$$

where $\overline{S}_i = S_i \cup U_i$, and $T_i(\omega) = 1$ if $\omega \in S_i$, and 0 otherwise. We iterate over a finite number of possible values to find the optimal value of r_i.

We aim to find a partition of the environment map that minimizes $\sum_i \xi_i \eta_i^t$, where t is a scalar to control the relative importance of ξ_i and η_i for guiding the partitioning (experiments show that $t = 0.5$ gives best results). An *ideal* partition would have the subregions all having equal energy and each having a corresponding axis-aligned square in the parameter domain. This minimization problem is solved by a bottom-up algorithm. Considering each pixel in the initial environment map as a subregion, we iteratively merge the subregions under the guidance of ξ_i and η_i. For each iteration, we select two connected subregions and merge them into one subregion. The selection criterion is that, among all the candidate subregion pairs, the newly merged subregion introduces the least change to the current overall error $\sum_i \xi_i \eta_i^t$. Each iteration decreases the number of the current subregions by one. Therefore, after thousands of iteration steps, we obtain a prescribed number of subregions. This procedure takes about 30 seconds for a $6 \times 32 \times 32$ environment map. The pseudocode of environment map partitioning is listed in Algorithm 1.

Input: An environment map with m pixels.
Output: A set of subregions $S = \{s_i | 0 \le i < k\}$ where k is the prescribed number of subregions.

Set $S = \{s_i | 0 \le i < m\}$ where subregion s_i only consists of the i-th pixel of the environment map;
while $m > k$ **do**
  Declare error array $error[m]$;
  Declare index array $index[m]$;
  for $i \leftarrow 0$ **to** m **do**
    if s_i *has no neighbor* **then**
      $error[i] \leftarrow \infty$;
      $index[i] \leftarrow -1$;
    else
      /* find min error during merging s_i and s_k */
      $k \leftarrow \arg\min_k \{\xi_{i,k}\eta_{i,k}^t | k \in neighbor(i)\}$;
      $error[i] \leftarrow \xi_{i,k}\eta_{i,k}^t$;
      $index[i] \leftarrow k$;
    end
  end
  $s_j \leftarrow$ subregion with $\min_i error[i]$;
  if $index[j] = -1$ **then** break;
  $s_j \leftarrow s_j \cup s_{index[j]}$;
  $s_{index[j]} \leftarrow s_{m-1}$;
  $m \leftarrow m - 1$;
end

Fig. 4 shows two examples of partitioning environment light maps. Note that the regions with high-intensity light have dense distribution of subregions. This effect is very similar to the importance sampling strategy [6]. In our experiments, we found that $20 \sim 40$ subregions are enough to give compelling rendering results for most kinds of environment maps.

Besides distant environment maps, our method can also handle local lights by employing the source radiance field (SRF) of shadow field framework [4]. At each sample point around a local light, we partition the recorded radiance map from that local light to a set of subregions and associate each subregion with $\langle c_i, r_i\rangle$. The remaining tasks are similar to what we do for rendering under environment maps.

2. $ratio_0 = \sqrt{2}/2$ and $ratio_1 = 2$ in our adopted hemisphere parameterization method (Section 6.1).

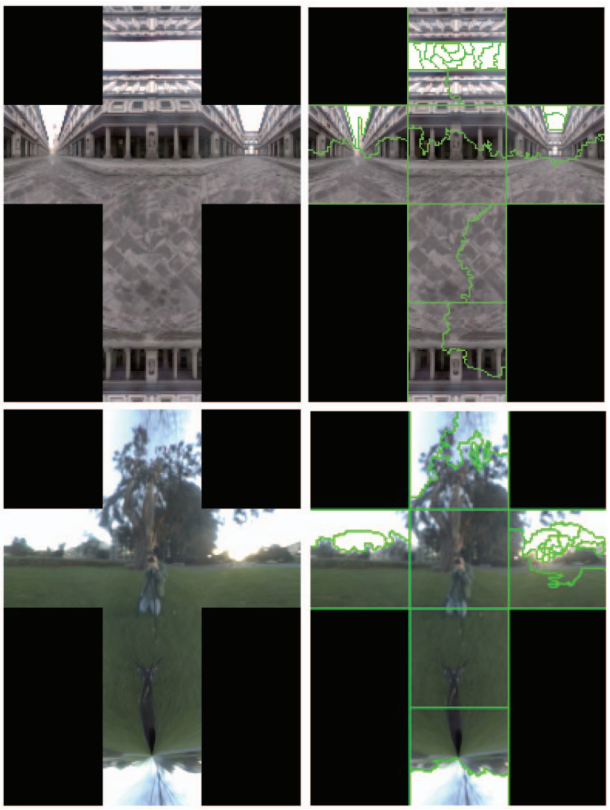


Fig. 4. Examples of environment map partition. Left column: the original environment maps. Right column: the partitioned subregions bounded by green lines.

5.2 BRDF SAT Precomputation and Runtime Lookup

According to (2), computing the BRDF coefficient $\rho_{x,\omega_0,i}$ at a point x is equivalent to integrating the BRDF function $\rho(x,\omega,\omega_0)$ over subregion S_i (Fig. 3a). Equivalently, the integral can be evaluated over the corresponding region C_i in the 2D parameter domain (Fig. 3b). In addition, as BRDF is a function of both incoming light direction ω and outgoing view direction ω_0 *relative to a local orientation at* x, BRDF at the local frame of every vertex is the same. Therefore, the evaluation of $\rho_{x,\omega_0,i}$ in the local frame at x can be formulated as

$$\rho_{x,\omega_0,i} = \frac{1}{|S_i|}\int_{S_i} \rho(x,\omega,\omega_0)d\omega = \frac{1}{|C_i|}\int_{C_i} \rho(p,\omega_0')dp,$$

where ω_0' denotes the view direction ω_0 rotated to the local frame at x. We approximate the integral $\frac{1}{|C_i|}\int_{C_i} \rho(p,\omega_0')dp$ using the SAT technique as explained below.

In the precomputation step, for each view direction ω_0', we preintegrate the SAT of the BRDF function as

$$SAT(u,v,\omega_0') = \int_{R(u,v)} \rho(p,\omega_0')dp,$$

where (u,v) is a point in the parameter domain corresponding to a light direction in the global frame, and $R(u,v)$ is a rectangle determined by points $(0,0)$ and (u,v) (Fig. 1).

Unlike previous related methods [3], [4], [15], which define the BRDF in the global frame and require 6D BRDF data storage, our method only needs a global 3D or 4D BRDF SAT,

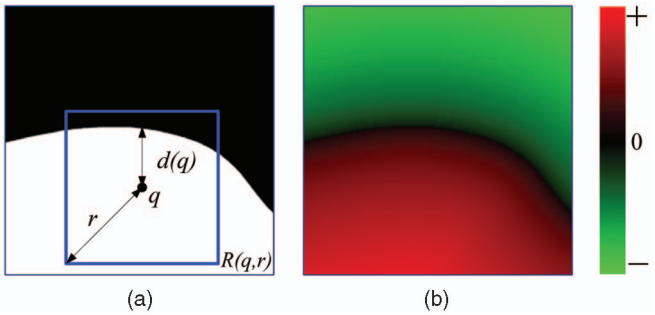


Fig. 5. VDT definition. (a) A visibility map. (b) The corresponding VDT. The distance is normalized.

since $SAT(u,v,\omega_0')$ is the same in the local frame at any vertex x of a scene. For an anisotropic BRDF, we have to tabulate the view direction ω_0' over the whole hemisphere, which is 2D, so the global BRDF SAT is 4D. For an isotropic BRDF, we only need to tabulate the polar angle of the view direction ω_0'; therefore, the whole SAT is 3D.

The coefficient $\rho_{x,\omega_0,i}$ can be approximately evaluated by the following formula:

$$\rho_{x,\omega_0,i} = \frac{1}{|C_i|}\int_{C_i} \rho(p,\omega_0')dp \approx \frac{1}{|R_i|}\int_{R_i} \rho(p,\omega_0')dp,$$

where $R_i = R_x(c_i,r_i)$ is the axis-aligned square associated with S_i in the local parameter domain at x. Therefore, during runtime rendering, $\rho_{x,\omega_0,i}$ (the integral $\int_{R_i} \rho(p,\omega_0')dp$ over R_i) can be efficiently computed by using only four lookups of the SAT (1).

Because we compute and store the BRDF SAT independently of a scene to be rendered, we achieve the following benefits. First, we can on-the-fly swap or edit the BRDF of a model. For analytic BRDFs, given the new parameters, the BRDF SAT data (a 3D/4D table) can be regenerated on the fly, achieving interactive editing of BRDF. Second, under the assumption that visibility integrals are ignored for local shading, our method can easily handle local deformable shading effects, as the BRDF integral is computed in local frame.

5.3 Visibility VDT Precomputation and Runtime Lookup

Similar to the BRDF integral, the SAT technique is directly applicable to the fast approximation of the visibility integrals. However, noticing that visibility data are binary, we propose a more efficient method to approximate the visibility integrals, called *VDT*. In this section, we first present the idea of approximating the integral of a general visibility function over a square in the parameter domain using VDT and then explain how to use the VDT technique to approximate the self-visibility and occluder-visibility coefficients efficiently.

Given a 2D visibility map $v(p)$ (Fig. 5a), which is a function of point p in the hemisphere parameterization domain, the VDT (Fig. 5b) is defined as follows:

$$VDT(p) = d(p)\ sign(p),$$

where $d(p)$ is the nearest distance from point p to the binary-change boundary of $v(p)$, and $sign(p) = 1$ if $v(p) = 1$, and -1 otherwise. We use the method proposed by Danielsson [20] to compute $d(p)$. We propose to approximate the

integral of $v(p)$ over the square $R(q,r)$, which is centered at q and of size $\sqrt{2}r$, as

$$\frac{1}{|R(q,r)|}\int_{R(q,r)} v(p)dp \approx min(1, max(0, F(q,r)), \qquad (6)$$

where $F(q,r) = \frac{\sqrt{2}VDT(q)+r}{2r}$ approximates the percentage of points with value 1 in the integral square $R(q,r)$.

VDT has several advantages. First, only one lookup is needed for visibility integral approximation, which is four times faster than the lookup of SAT. Second, unlike SAT, VDT has no precision problem when used in graphics hardware. Third, VDT is a continuous signal, which can be compressed more efficiently while giving fewer artifacts. However, using VDT to look up an integral is only accurate when the visibility boundary is a straight line (see a comparison example in Fig. 10). Nevertheless, our experiments show that rendering results are acceptable in most cases.

5.3.1 Self-Visibility

Similar to the evaluation of the BRDF coefficients, the self-visibility coefficient $v_{x,i}$ can be approximated by an integral over the square $R_i = R_x(c_i, s_i)$ in the 2D parameter plane of the hemisphere in the local frame at x:

$$v_{x,i} = \frac{1}{|S_i|}\int_{S_i} V(x,\omega)d\omega \approx \frac{1}{|R_i|}\int_{R_i} V(x,p)dp.$$

In the precomputation step, we ray trace to compute the visibility map $V(x,p)$ at each vertex x in its local frame and generate the corresponding VDT $VDT(x,p)$ (that is, per-vertex 2D VDT). We call all these tables collectively as the *self-visibility distance field* (SVDF). During runtime rendering, we compute the approximation of $v_{x,i}$ (that is, $\frac{1}{|R_i|}\int_{R_i} V(x,p)dp$) by (6) through one lookup of $VDT(x,p)$.

5.3.2 Occluder Visibility

We employ the shadow field framework [4] to handle dynamic-scene rendering. Similar to the approximation of the self-visibility coefficients, we approximately compute the occluder-visibility coefficient $v_{O_j,x,i}$ using the following formula:

$$v_{O_j,x,i} = \frac{1}{|S_i|}\int_{S_i} V_{O_j}(x,\omega)d\omega \approx \frac{1}{|R_i|}\int_{R_i} V_{O_j}(x,p)dp.$$

In the precomputation step, we compute and store 2D visibility data for each sampled point in the 3D surrounding space of an object using the sampling method of object occlusion field (OOF) [4]. As our rendering framework utilizes SPCBFs instead of the SH or wavelet basis used by Zhou et al. [4], our method differs from theirs in the following aspects. First, at each sampled point x around an object O_j, we capture the visibility map only on the hemisphere defined by the direction from x to the center of O_j, rather than the visibility map in the global frame. Second, we store a VDT instead of a visibility map at each sampled point. The VDTs at all sampled points are collectively referred to as the *occluder visibility distance field* (OVDF). In the rendering step, we approximate each occluder integral on the VDTs of the sampled points using (6). Similar to that in [4], the VDT at an intermediate point is approximated by a trililnear interpolation of the eight nearest samples.

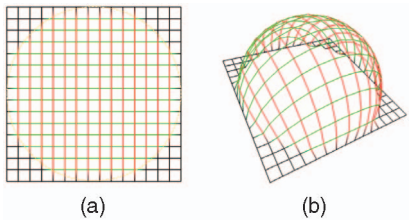


Fig. 6. Modified Lambert equal-area parameterization.

6 Implementation

We have implemented our rendering algorithm on graphics hardware. This section presents the implementation details, including hemisphere parameterization, data compression and the shader program in GPU.

6.1 Hemisphere Parameterization

We use a variant of the Lambert equal-area parameterization method mentioned in [21] to parameterize a unit hemisphere, which is originally used for building an area-preserving mapping between a unit sphere and a unit disk. Specifically, we define a one-to-one mapping from a unit hemisphere to a unit disk through the following mapping function:

$$(u,v) = f(x,y,z) = (x/\sqrt{1-z}, y/\sqrt{1-z}).$$

For the convenience of sampling and storage, we extend the unit disk to a 2×2 square (Fig. 6a). For BRDF and self-visibility functions, the extended region is valued by 0. For occluder visibility function, the extended region is valued by 1.

We have also tested cube map parameterization and hemisphere parameterization [22], but we find that the Lambert equal-area parameterization gives the least distortion. The cube map parameterization is not area preserving, and the size of a region changes when mapping from a sphere to a cube map, leading to noises. The hemisphere parameterization in [22] produces artifacts when a region crosses the diagonal of the unit disk.

6.2 Data Compression

Let ω_0 and ω denote the view and light directions, respectively, and let θ_0 denote the polar angle of the view direction ω_0. For an isotropic BRDF $\rho(\omega,\theta_0)$, we sample ω at 32×32 directions of the hemisphere parameterization and θ_0 at 32 angles. For an anisotropic BRDF $\rho(\omega,\omega_0)$, we sample both ω and ω_0 at 32×32 directions. As there is only one globally stored BRDF SAT, no compression is needed.

The 4D SVDF is precomputed with a sampling rate of $32 \times 32 \times N$, where 32×32 is the size of the VDT at each vertex, and N is the number of vertices in the scene. The 5D OVDF is precomputed at $32 \times 32 \times (6 \times 32 \times 32) \times 16$, where 32×32 is the VDT size, $(6 \times 32 \times 32) \times 16$ is because, like OOF in [4], we use a cube sampling of $6 \times 32 \times 32$ to sample the space around each object on 16 different concentric spheres with radii ranging from $0.4r$ to $6r$, where r is the radius of the bounding sphere. As a result, the data of the SVDF for a 40K vertex mesh is about 160M (32FP), and the data of the OVDF is about 384M. Both of them need to be compressed before putting into the GPU.

For SVDF compression, we use the Clustered PCA (CPCA) method [7]. Taking a 40K vertex model for example, with 256 clusters and eight eigenvectors for each cluster, a compression ratio of about 1:17(9M) gives a good result.

We compress OVDF as follows: For each concentric sampling sphere (using cube map sampling), we split all the six faces of the cube map into 2×2 segments and obtain 24 segments on each sphere, like in [9]. We use 16 concentric sampling spheres, resulting in a total of $24 \times 16 = 384$ segments. Each segment is compressed using PCA. With eight eigenvectors per segment, a compression ratio of about 1:26(15M) gives a good result.

We pack BRDF, SVDF, and OVDF data into textures to load into the GPU. For higher accuracy, we use 16FP textures instead of 8BP. For BRDFs, we pack both isotropic and anisotropic BRDFs into 3D textures.[3] For the compressed SVDF and OVDF, we pack the eigenvalues and the cluster indices of the vertices into 2D textures. Eigenvectors of SVDF and OVDF are packed into 2D and 3D textures, respectively.

We use one more texture to store $\langle c_i, r_i \rangle$ associated with each partitioned subregion, which is updated after the rotation of the environment light in each frame.

6.3 Shader Program in GPU

Although our algorithm is a per-vertex rendering method, we utilize the render-to-vertex-array technique and perform the rendering process in two steps. In the first step, we pack the object vertices into a 2D rectangle, with each vertex corresponding to one pixel in the rectangle, and use a pixel shader program to calculate the color for each pixel and render the rectangle to a *frame buffer object (FBO)*. The color is copied from the *FBO* to the vertex array using the OpenGL extension *pixel buffer object (PBO)*. In this step, the vertex attributes are needed for color calculation at each vertex; so, we pack the positions, normals, and tangents of the vertices into a 2D texture. In the second step, we use the OpenGL extension *vertex buffer object (VBO)* to render the scene using the color calculated in the first step. The pseudocode of the pixel shader is shown in Algorithm 2.

```
1  foreach object vertex x do
2      Initialize color B_x ← 0;
3      foreach subregion S_i associated with ⟨c_i, r_i⟩ and light
       coefficient l_i do
4          c'_i ← the corresponding direction of c_i in x's local
           frame;
5          ω' ← the corresponding direction of ω in x's local
           frame;
6          Look up SVDF with (c'_i, r_i) to get self-visibility
           integral V_s;
7          Look up BRDF SAT with (c'_i, r_i) and ω' to get
           BRDF integral B;
8          T ← |S_i| l_i (V_s / r_i^2) (B / r_i^2);
9          foreach occluder O_j do
10             Let p be the corresponding position of x in the
               local space of O_j;
11             c''_i ← the corresponding direction of c_i in p's
               local frame;
12             Look up OVDF with (c''_i, r_i) and p to get
               occluder visibility integral V_{O_j};
13             T ← T (V_{O_j} / r_i^2);
14         end
15         B_x ← B_x + T;
16     end
17     Assign color B_x to vertex x;
18 end
```

3. Specifically, we pack 4D anisotropic BRDF of resolution $32 \times 32 \times 32 \times 32$ to a 3D texture of size $(4 \times 32) \times (8 \times 32) \times 32$.

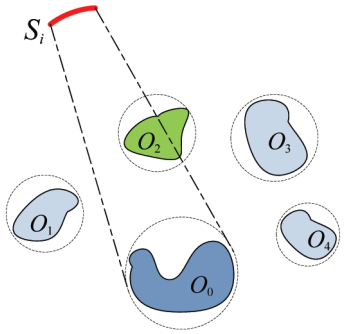


Fig. 7. Occluder culling. For subregion S_i, to process object O_0, only occluder O_2 needs to be considered, whereas O_1, O_3, O_4 can be ignored.

In the shader program, several textures are looked up for each vertex, namely, attribute texture, BRDF texture, eigenvalue, cluster index, and eigenvector textures of SVDF and OVDF. As a result, texture fetching is the bottleneck. We use occluder culling to accelerate it (as shown in Fig. 7). To do this, we perform multipass rendering in the first step, with one pass for each subregion. The resulting images of each pass are blended together to generate a final image. Then, before each pass, we determine the occluders that can be neglected in that pass using CPU, and only send the remaining occluders to the GPU.

7 Error Analysis

7.1 Errors in SPCBF-Based PRT Representation

Previous all-frequency PRT frameworks choose a basis separately for each integrand (that is, the light, BRDF, and visibility functions), guaranteeing that the representation of each function in its own basis has a low error rate. In contrast, our method determines a basis according to the environment map and represents both the BRDF and visibility functions in the same basis as the light function. Since the light-driven basis is not optimized based on the BRDF and visibility signals, the representation error of BRDF or visibility might be large.

We show that the potential large errors in the BRDF and visibility representations are effectively suppressed in the multiple product computation (5). If the partition of the environment map is dense enough, all the representation errors of light, BRDF, and visibility in SPCBFs should be small. For a specific SPCBF $B_i(\omega)$, the representation error of BRDF or visibility corresponding to $B_i(\omega)$ is likely to be large only when the corresponding subregion S_i is large. However, since we design an error metric that is similar to an importance sampling strategy [6] to guide the environment map partitioning (Section 5.1), the subregions with large areas must have low-light intensity. Therefore, after multiplying the light, BRDF, and visibility representations together, the low-light intensity value (approaching zero) significantly weakens the errors introduced by the BRDF and visibility representations.

We have tested a variety of rendering scenarios to investigate how the representation error of BRDF or visibility is related to the sizes of the subregions. All the experiments demonstrate that although the representation error of each term is possibly large, the errors in the final multiple product are always very small (see statistical data

TABLE 2
Representation Errors of Light, Self-Visibility, and BRDF Functions in SPCBFs and Errors in the Final Multiple Products

	Light	Visibility	BRDF	Total
Uffizi Gallery	1.43%	20.97%	14.31%	**0.04%**
	1.43%	26.80%	17.87%	**0.12%**
	1.43%	21.42%	23.89%	**1.04%**
	1.43%	34.67%	21.02%	**1.05%**
	1.43%	45.37%	15.88%	**0.69%**
Campus	10.89%	22.57%	19.57%	**0.36%**
	10.89%	16.86%	15.73%	**0.15%**
	10.89%	24.03%	20.96%	**1.14%**
	10.89%	18.43%	28.03%	**0.05%**

in Table 2 and an example of light, self-visibility, and BRDF representations in SPCBFs shown in Fig. 8, with errors corresponding to the underlined row in Table 2). We use the Sum of Squared Difference (SSD) error to measure the error between an approximation representation and its corresponding ground truth.

7.2 Errors in SAT and VDT Lookup

During runtime rendering, we use the parameter $\langle c_i, r_i \rangle$ associated with each subregion S_i to fast look up SAT (VDT) for the computation of the corresponding coefficient of BRDF (visibility). The approximation error introduced in this step is guaranteed to be negligible when S_i is small enough, because we exploit the mismatch error metric of $R_x(c_i, r_i)$ and $C_i = f_x(S_i)$ to guide the partition of the environment map. By experiments, we found that the SSD error in the BRDF or visibility coefficient computed through lookups, which is due to both the mismatch between the square $R_x(c_i, r_i)$ and C_i and the variance of BRDF or visibility in S_i, is almost linearly proportional to the region size of S_i (Fig. 9). Therefore, the errors in the SAT and VDT lookups are also suppressed when computing the multiple product of light, BRDF, and visibility coefficients, because for each subregion, the light intensity is roughly inversely proportional to its region size.

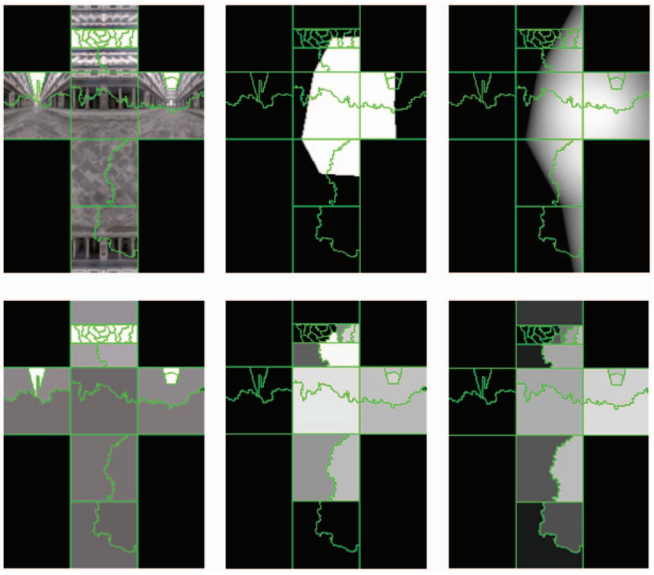


Fig. 8. Top row: the light (Uffizi Gallery environment map), self-visibility, and BRDF functions (from left to right). Bottom row: the corresponding SPCBF representations.

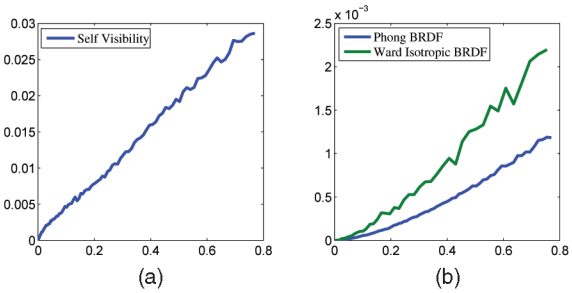


Fig. 9. Relationship between the SSD errors (y-axis) of the (a) self-visibility and (b) BRDF coefficients of a subregion S_i computed through SAT/VDT lookups, and the region size of S_i (x-axis) is almost linear.

Fig. 10 illustrates a comparison example between using SAT and VDT for computing the visibility coefficients. Compared with the ground truth (obtained by ray tracing), the rendering results with either SAT or VDT have only small SSD errors. VDT approximation leads to slightly larger errors, since the assumption that visibility boundaries are straight is not always fully satisfied, causing more black shadows. As expected, VDT approximation gives less artifacts than SAT approximation when the compression ratio becomes higher, since VDT is more suitable for compression due to its continuous representation.

7.3 Errors in Rendered Images

In Figs. 11 and 12, we compare the proposed PRT rendering results with the ground truth and the results of the area-weighted wavelet method [2] for both Kitchen and Grace Cathedral environment maps. 20, 30, 60, and 100 SPCBFs are used in our method, and 20, 30, 60, and 100 wavelet terms (for each of RGB channels) are used in the wavelet method [2]. L^2 errors are measured for both methods. From the figures, we can see that the rendering error of our method under all-frequency environment lighting is small. Results with 100 SPCBFs are almost as accurate as the ground truth; with 30 SPCBFs, the accuracy is as that of wavelets with 100 terms per channel. Note that, under the same number of basis functions, our method is faster than the methods based on Wavelets for the computation of multiple products but needs more storage (to store SAT and VDT).

8 Results and Discussions

The performance of different scenes is shown in Table 3. We have applied occluder culling to dynamic scenes. The performance is reported on a Pentium IV 3.2-GHz PC with an Nvidia GeForce 7800GT 256-Mbytes graphics card. Interactive frame rates are achieved for large dynamic scenes under all-frequency environment lighting, and real-time frame rates are achieved for static scenes under all-frequency environment lighting, which is much faster than previous methods.

Fig. 13 compares the results of SVDF under different levels of compression, whereas Fig. 14 compares the results of OVDF under different levels of compression. Figs. 15 and 16 show some rendering results of the robot scene and the kitchen scene. Fig. 17 shows rendering results of different BRDFs; we use the BRDFs in [23]. Fig. 18 shows results of local light illumination and local deformable shading.

Fig. 10. Comparison between using SAT and VDT for computing the visibility coefficients. All the scenes are rendered under rectangle-shaped area lights of solid $\text{angle} = \pi/36$. Left four subfigures: top row (left: $\text{error} = 0.02$ percent, right: $\text{error} = 0.01$ percent) and bottom row (left: $\text{error} = 0.30$ percent, right: $\text{error} = 0.13$ percent) are results using SAT and VDT, respectively. Note the difference in the cast shadows (of the stems of the plant or of the teapot handle and spout) on the ground plane. The artifacts arose because the assumption that visibility boundary is a straight line does not hold here. Right four subfigures: Top row are rendered using SAT with compression rate 1:14 (left) and 1:29 (right); bottom row are rendered using VDT with compression rate 1:14 (left) and 1:29 (right).

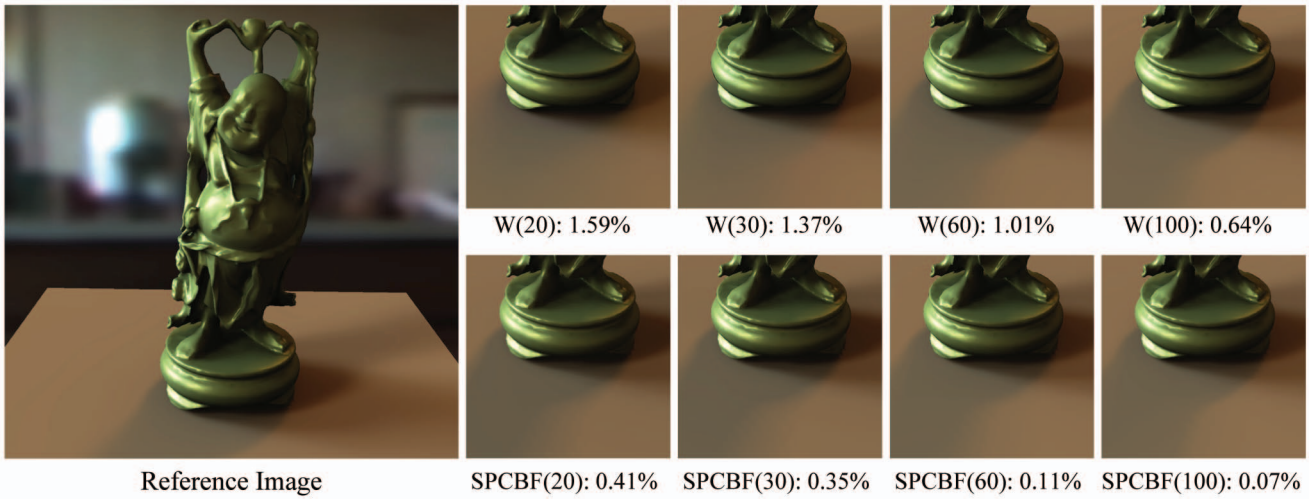


Fig. 11. Rendering of a Buddha scene in Kitchen. Note the difference in the shadow boundaries.

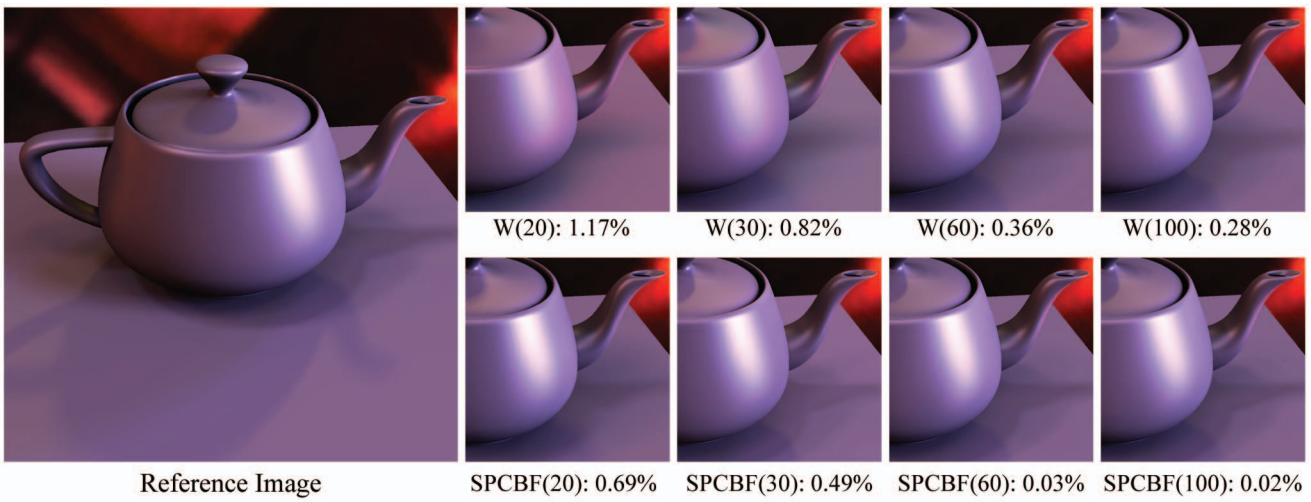


Fig. 12. Rendering of a teapot scene in Grace Cathedral. Note the difference in the glossy area and the shadow boundaries.

Limitations. Since an optimization process is needed to find a set of SPCBFs for a given environment light, which typically takes about 30 seconds with our unoptimized implementation, real-time replacement of environment maps is not allowed. The method based on SRBFs [5] shares the same limitation. Furthermore, our method cannot handle some situations where the BRDF has more details in a large partitioned region of the environment map, since BRDF is represented with the SPCBFs optimized for the light. Last, our current framework does not allow indirect

TABLE 3
Performance Results

Scene	Vertices	FPS	Num. of SPCBFs
Robot	60k	10.8	30
Kitchen	80k	9.9	30
Buddha (static scene)	45k	27	30

lighting, because supporting varying viewing directions and dynamic scenes requires a 4D transport matrix at each vertex, making precomputation not memory affordable. Unlike the original PRT method proposed by Sloan et al. [1], many of the following techniques in [3], [4], and [5] also sacrifice indirect lighting for newer effects.

9 Conclusion and Future Work

In this paper, we present a new basis function SPCBF for PRT, which is able to represent all-frequency signals and support efficient rotation and efficient multiple product. By precomputing the light coefficients and runtime computing the BRDF and visibility coefficients, the proposed PRT framework in SPCBFs supports a variety of rendering effects.

In our current implementation, we use SAT and VDT to fast approximate the coefficients of BRDF and visibility, respectively. This step inevitably introduces approximation errors and involves large size precomputation storage (per-vertex 2D VDT). We are seeking new fast integration techniques, which are more precise and need less storage.

There exists a number of interesting directions for further investigation. First, the proposed method is limited to direct lighting. As future work, we would like to incorporate indirect lighting and interreflection into our framework. Second, we would also like to extend our current material representation, 4D BRDF, to higher dimensional materials, like spatial variant BRDF and BTF. Third, we will explore other compression techniques (for example, tensor approximation techniques [5]) rather than CPCA to compress the visibility data.

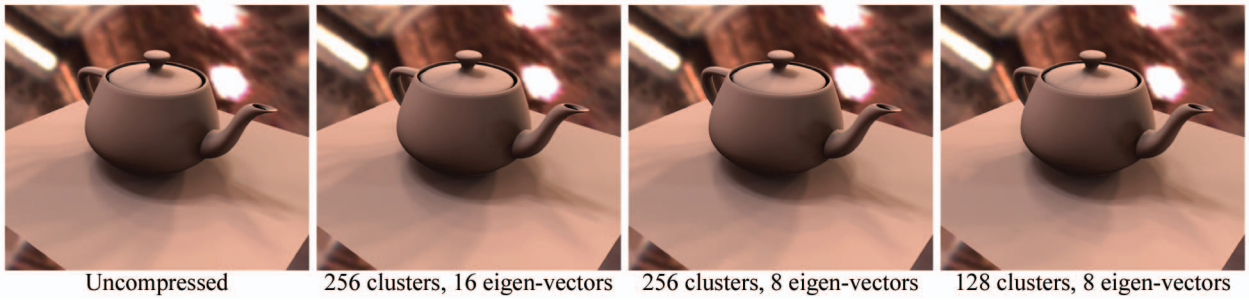


Fig. 13. SVDF compression result. From left to right: uncompressed, using 256 clusters and 16 eigenvectors, using 256 clusters and eight eigenvectors, and using 128 clusters and eight eigenvectors.

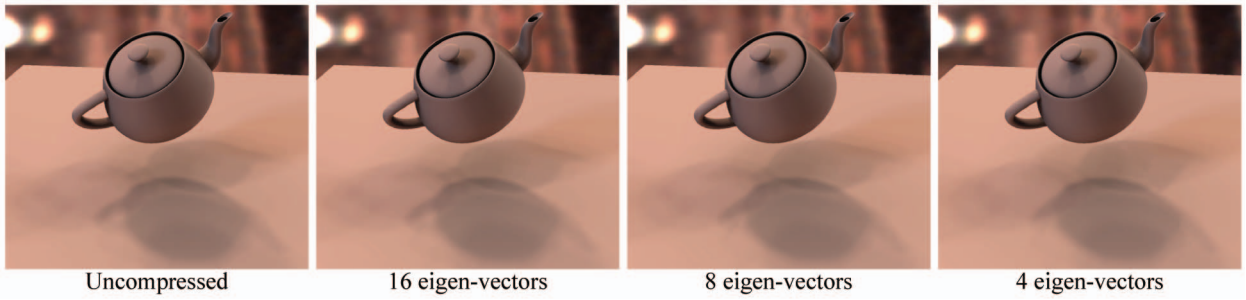


Fig. 14. OVDF compression result. From left to right: uncompressed, using 16 eigenvectors, using eight eigenvectors, and using four eigenvectors.

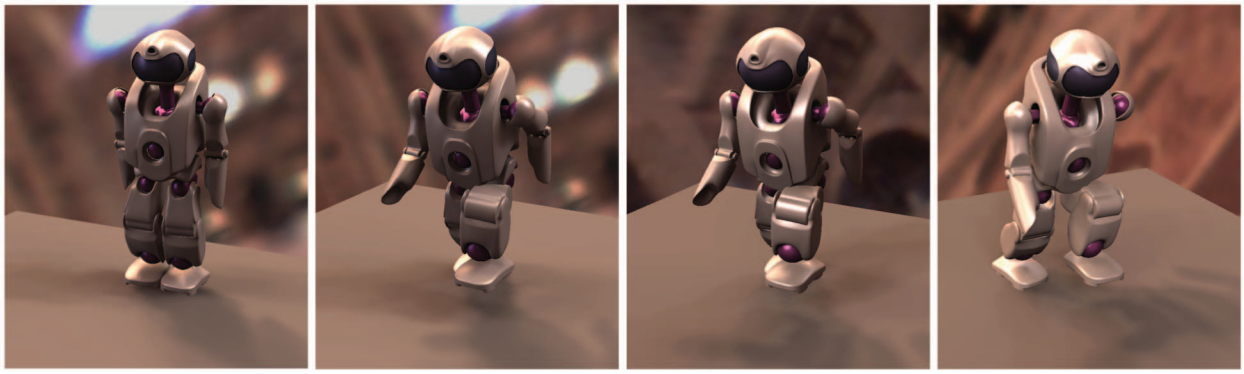


Fig. 15. Rendering results of the robot scene under dynamic environment map and changing view. The robot is composed of 12 components.

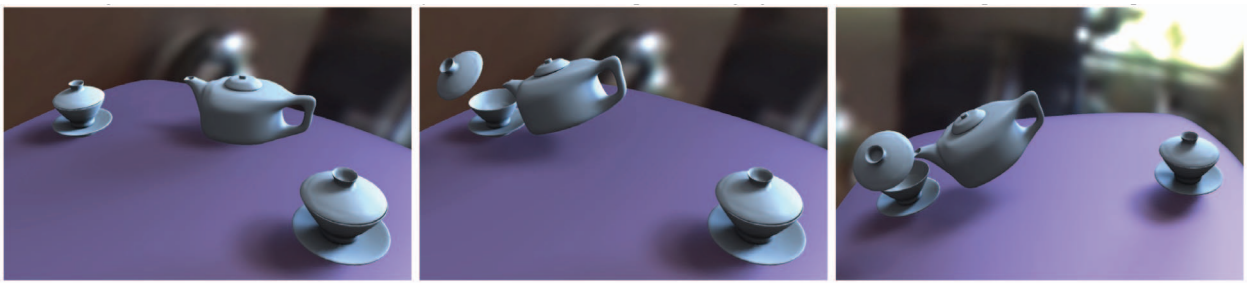


Fig. 16. Rendering results of the kitchen scene under dynamic environment map and changing view. The scene is composed of five components.

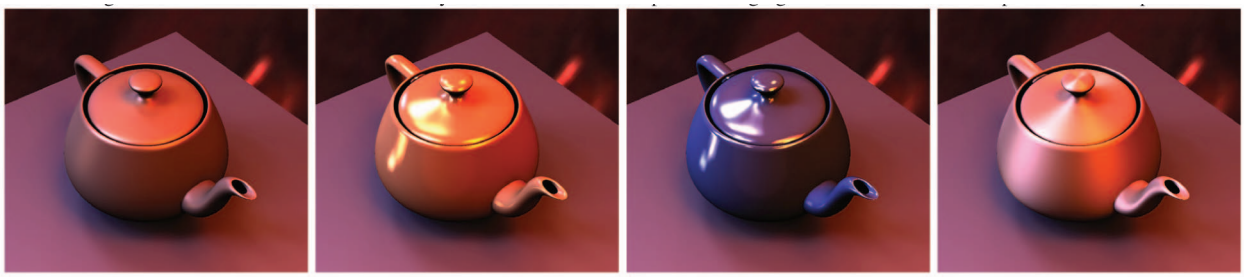


Fig. 17. Rendering results of different BRDFs. From left to right, the BRDF models we used are steel Phong, bronze Cook-Torrance, metal Ward Isotropic, and steel Ward Anisotropic.

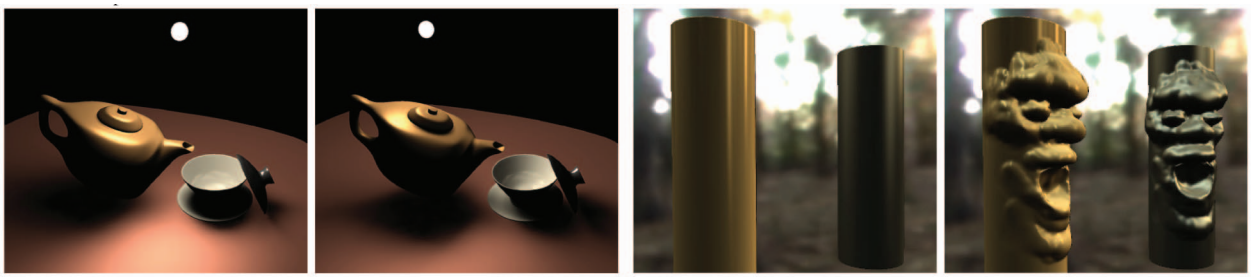


Fig. 18. Local light illumination and local deformable shading. The left two figures illustrate the kitchen scene under a local light source. The right two figures show local deformable shading under environment map.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. They would also like to thank Xi Wang for a lot of useful discussions and suggestions. This work was partly supported by the National Basic Research Project of China (Project 2006CB303106), Specialized Research Fund for the Doctoral Program of Higher Education (Project 20060003057), and the National High Technology Research and Development Program of China (Project 2007AA01Z336). This work was also supported by the Research Grant Council of the Hong Kong Special Administrative Region, China (Project 619905).

REFERENCES

[1] P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 527-536, 2002.

[2] R. Ng, R. Ramamoorthi, and P. Hanrahan, "All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 376-381, 2003.

[3] R. Ng, R. Ramamoorthi, and P. Hanrahan, "Triple Product Wavelet Integrals for All-Frequency Relighting," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 477-487, 2004.

[4] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum, "Precomputed Shadow Fields for Dynamic Scenes," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1196-1201, 2005.

[5] Y. Tsai and Z. Shih, "All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 967-976, 2006.

[6] S. Agarwal, R. Ramamoorthi, S. Belongie, and H.W. Jensen, "Structured Importance Sampling of Environment Maps," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 605-612, 2003.

[7] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder, "Clustered Principal Components for Precomputed Radiance Transfer," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 382-391, 2003.

[8] P.-P.J. Sloan, X. Liu, H.-Y. Shum, and J. Snyder, "Bi-Scale Radiance Transfer," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 370-375, 2003.

[9] X. Liu, P.-P.J. Sloan, H.-Y. Shum, and J. Snyder, "All-Frequency Precomputed Radiance Transfer for Glossy Objects," *Rendering Techniques,* pp. 337-344, 2004.

[10] R. Wang, J. Tran, and D.P. Luebke, "All-Frequency Relighting of Non-Diffuse Objects Using Separable BRDF Approximation," *Rendering Techniques,* pp. 345-354, 2004.

[11] R. Wang, R. Ng, D. Luebke, and G. Humphreys, "Efficient Wavelet Rotation for Environment Map Rendering," *Proc. Eurographics Symp. Rendering (EGSR '06),* 2006.

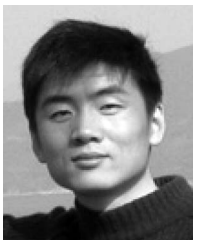
[12] P. Green, J. Kautz, W. Matusik, and F. Durand, "View-Dependent Precomputed Light Transport Using Nonlinear Gaussian Function Approximations," *Proc. Symp. Interactive 3D Graphics and Games,* pp. 7-14, 2006.

[13] K. Xu, Y. Gao, Y. Li, T. Ju, and S.-M. Hu, "Real-Time Homogenous Translucent Material Editing," *Eurographics,* 2007.

[14] C. Mei, J. Shi, and F. Wu, "Rendering with Spherical Radiance Transport Maps," *Computer Graphics Forum,* vol. 23, no. 3, pp. 281-290, 2004.
[15] W. Sun and A. Mukherjee, "Generalized Wavelet Product Integral for Rendering Dynamic Glossy Objects," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 955-966, 2006.
[16] J. Kautz, J. Lehtinen, and P.-P. Sloan, "Precomputed Radiance Transfer: Theory and Practice," *ACM SIGGRAPH Course Notes,* 2005.
[17] F.C. Crow, "Summed-Area Tables for Texture Mapping," *Proc. ACM SIGGRAPH '84,* pp. 207-212, 1984.
[18] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra, "Fast Summed-Area Table Generation and Its Applications," *Computer Graphics Forum,* vol. 24, no. 3, pp. 547-555, 2005.
[19] J.T. Kajiya, "The Rendering Equation," *Proc. ACM SIGGRAPH '86,* pp. 143-150, 1986.
[20] P.-E. Danielsson, "Euclidean Distance Mapping," *Computer Graphics and Image Processing,* pp. 227-248, 1980.
[21] J. Snyder and D. Mitchell, "Sampling-Efficient Mapping of Spherical Images," technical report, Microsoft, 2001.
[22] X. Liu, Y. Hu, J. Zhang, X. Tong, B. Guo, and H.-Y. Shum, "Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces," *IEEE Trans. Visual Computer Graphics,* vol. 10, no. 3, pp. 278-289, May/June 2004.
[23] A. Ngan, F. Durand, and W. Matusik, "Experimental Analysis of BRDF Models," *Proc. Symp. Rendering Techniques,* pp. 117-126, 2005.

Kun Xu received the bachelor's degree in computer science from Tsinghua University in 2005. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Tsinghua University. His research interests include real-time rendering and appearance modeling.

Yun-Tao Jia received the bachelor's and master's degrees from Tsinghua University in 2004 and 2006, respectively. He is currently a PhD candidate in computer science at the University of Illinois, Urbana-Champaign. His current research interests include global illumination, GPU rendering, and graph visualization.



Hongbo Fu received the BS degree in information science from Peking University, China, in 2002 and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology in 2007. He is currently a postdoctorate research Fellow in the University of British Columbia. His primary research interests include computer graphics with an emphasis on digital geometry processing, character animation, and hairstyle synthesis and analysis.

Shi-Min Hu received the PhD degree from Zhejiang University in 1996. He is currently a professor of computer science at Tsinghua University. His research interests include digital geometry processing, video-based rendering, rendering, computer animation, and computer-aided geometric design. He is on the editorial boards of *Computer Aided Design*. He is a member of the IEEE Computer Society.

Chiew-Lan Tai received the BSc degree in mathematics from the University of Malaya, the MSc degree in computer and information sciences from the National University of Singapore, and the DSc degree in information science from the University of Tokyo. She is an associate professor of computer science at the Hong Kong University of Science and Technology. Her research interests include geometric modeling and processing, computer graphics, and reconstruction from architecture drawings.