Real-Time Volume Rendering in Dynamic Lighting Environments Using Precomputed Photon Mapping

Yubo Zhang, *Student Member*, *IEEE*, Zhao Dong, *Member*, *IEEE*, and Kwan-Liu Ma, *Fellow*, *IEEE*

Abstract—We present a framework for precomputed volume radiance transfer that achieves real-time rendering of global illumination effects for volume data sets such as multiple scattering, volumetric shadows, and so on. Our approach incorporates the volumetric photon mapping method into the classical precomputed radiance transfer pipeline. We contribute several techniques for light approximation, radiance transfer precomputation, and real-time radiance estimation, which are essential to make the approach practical and to achieve high frame rates. For light approximation, we propose a new discrete spherical function that has better performance for construction and evaluation when compared with existing rotational invariant spherical functions such as spherical harmonics and spherical radial basis functions. In addition, we present a fast splatting-based radiance transfer precomputation method and an early evaluation technique for real-time radiance estimation in the clustered principal component analysis space. Our techniques are validated through comprehensive evaluations and rendering tests. We also apply our rendering approach to volume visualization.

Index Terms—Volume rendering, precomputed radiance transfer, volume ray casting, multiple scattering, volume shadow

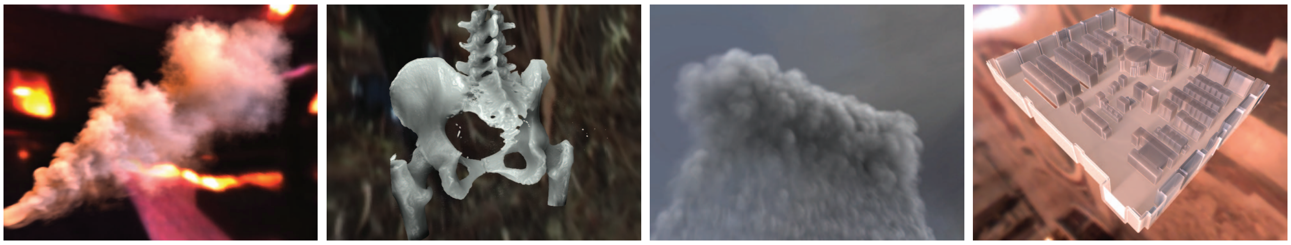
1 Introduction

Light traversing a volume usually undergoes absorption and scattering events. Before reaching the viewpoint, the radiance energy can scatter one time (i.e., single scattering (SS)) or multiple times (i.e., multiple scattering (MS)) inside the volume. Following the definition of surface light transport, we can regard the shading from SS as the *direct illumination* and from MS as the *indirect illumination* or *global illumination* (GI) for the volume data set. The volume GI effects can greatly enhance the visual perception of the volume data set. For example, global occlusion and shadow effects present better depth order information, and MS effects can also highlight the local thickness of the spatial structures given appropriate light settings. However, simulating all light transportation within volume including absorption and scattering is time-consuming. Therefore, most existing methods for volume rendering have to strike a balance between performance and quality. Great efforts have been made to accelerate the light simulation process while preserving shading quality such as two-step radiance estimation [1] and volumetric photon mapping (VPM) [2]. Although they achieve high-quality results, it is difficult to achieve interactive frame rates with these techniques. For comparison, classic volume ray casting with simple Phong shading can achieve real-time performance using modern graphics processing units (GPUs). Still, even with advanced shading techniques such as ambient occlusion and approximated scattering effects, the quality and accuracy of the real-time methods cannot match the previously mentioned offline rendering techniques. To achieve real-time performance and high visual quality at the same time, a practical strategy is to introduce preprocessing.

Recently, the precomputed radiance transfer (PRT) technique was developed for real-time surface rendering with GI effects including soft shadows and subsurface scattering [3]. Its application for rendering surface meshes gives promising results, so many extensions and improvements have been made. However, most PRT-based research focuses on rendering surface meshes, and the design of a general PRT framework for volume rendering has not received much attention. The issues involved are lighting approximation, volume radiance approximation, efficient precomputation of general volumetric effects, volume radiance compression, and sampling strategy during real-time volume rendering. To achieve efficient precomputation while maintaining volume GI effects, VPM [2] is a practical method to use. The major challenges are the seamless incorporation of VPM into the precomputation stage with an appropriate photon casting method and the determination of a fast radiance estimation strategy. We present a novel precomputed volume radiance transfer (PVRT) method to achieve successful integration so that high-quality volume GI effects can be rendered in real time (Fig. 1). With PVRT, we introduce a photon casting method that generates all the photons at the volume boundary. Each photon has varying energy and may also carry negative energy. To estimate the radiance, we adopt an efficient splatting-based radiance estimation method. We

• Y. Zhang and K.-L. Ma are with the Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616. E-mail: {ybzhang, klma}@ucdavis.edu.
• Z. Dong is with the Program of Computer Graphics, Cornell University, 588 Rhodes Hall, Ithaca, NY 14853. E-mail: zd@graphics.cornell.edu.

Manuscript received 8 Nov. 2011; revised 8 Aug. 2012; accepted 19 Jan. 2013; published online 13 Feb. 2013.
Recommended for acceptance by K. Mueller.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-11-0276.
Digital Object Identifier no. 10.1109/TVCG.2013.17.



(a) Smoke Plume, $300 \times 300 \times 150$, 45fps (b) Bone, $512 \times 512 \times 512$, 40fps (c) Smoke Wall, $400 \times 300 \times 100$, 42fps (d) Machine Room, $417 \times 345 \times 60$, 47fps

Fig. 1. Examples of real-time rendering including volume GI effects using the PVRT method.

also contribute a new discrete spherical function (DSF) for lighting approximation. DSF achieves better quality than the classic spherical harmonics (SH) basis function for approximating high-frequency lighting, and unlike wavelet basis functions, it is also rotationally invariant. In addition, the interpolation property of DSF enables much faster construction and evaluation when compared with spherical radial basis functions (SRBFs) [4], while the approximation quality is similar. To summarize, our contributions include

- A novel DSF for light approximation, which is rotationally invariant, fast to construct and evaluate, and has the ability to approximate high-frequency lighting.
- A PVRT method that successfully incorporates the VPM method into the PRT framework and achieves efficient and general volume radiance estimation.
- A comprehensive performance and quality evaluation of our approach. We also demonstrate its application in interactive scientific and medical volume visualization.

The remainder of this paper is organized as follows: Section 2 describes existing works, and Section 3 reviews the background of light transport in participating media. After an overview of the PVRT method in Section 4, Section 5 introduces the new DSF and its properties. Next, Sections 6 and 7 discuss the general procedure for incorporating the VPM technique into the PRT framework as well as the final rendering step. Several implementation details are discussed in Section 8. The experimental results and a comprehensive evaluation of our method are given in Section 9. Finally, Section 10 concludes the paper and introduces plan for future works.

2 Related Works

We refer readers to [5], [6] for a complete survey of volume GI rendering, and to [7], [8] for a comprehensive introduction of PRT methods. In this section, we only cover the most relevant works.

Offline volume GI rendering. The rendering of volume GI effects is essentially the solution of the radiative transfer equation [9]. Kajiya and Von Herzen [1] presented a two-pass ray tracing method. First, to estimate radiance for each voxel, the radiative transfer equation is approximately evaluated using SH. Thereafter, ray marching is performed along view rays to gather the final radiance. Since the MS inside the volume usually exhibits low-frequency behavior, only the first few SH bases are necessary for generating visually convincing results. Afterward, several methods have been proposed to compute numerical solutions of the radiative transfer equation by using either ray tracing [10], [11] or radiosity [12]. These solutions can render various volume GI effects, but the computational cost is enormous, so they can only run in offline mode. To improve performance, Stam [13] suggested to approximate MS effects using the diffusion process, which can be simulated by either a multigrid scheme or a finite-element blob method. Nevertheless, the speed is still far from interactive.

The VPM method [2] was proposed to render volume caustics. In its first stage, photons are traced through the volume and stored into a hierarchical data structure, like a kd-tree. Afterward, ray marching is performed, and at each step, the stored photons are gathered to compute the final radiance. This final radiance gathering step can also be computed more efficiently by photon splatting [14]. The creation and maintenance of the hierarchical data structure prevent VPM from achieving interactive performance. Even with a GPU-based kd-tree [15], rendering high-frequency volume caustics with tens of millions of photons is still nontrivial. Our PVRT method also adopts photon tracing to simulate light transport and efficiently precomputes the radiant exitance for each voxel based on photon splatting.

Basis function in PRT. In the precomputation stage, PRT-based methods simulate complex light transfer and approximate the radiance transfer using certain kinds of basis functions. Then, the radiant exitance can be approximately reconstructed by multiplying the coefficients of incident lighting and the radiance transfer matrix. Several functions have been successfully applied in PRT, such as SH [3], wavelets [16], [17], SRBFs [4], Gaussians [18], and so on. Each of these functions has its own pros and cons. The SH basis smoothly reconstructs low-frequency lighting effects very well and can be freely rotated, but it cannot efficiently represent high-frequency signals. The wavelet basis can represent all-frequency effects with a small number of coefficients, but its rotation is nontrivial and could introduce temporal flickering due to nonlinear reconstruction. The SRBF can reconstruct all-frequency effects and is easy to rotate, but its precomputation cost is much higher than that of other functions. To realize all-frequency rendering, we introduce a novel DSF that achieves similar quality to SRBF with much less precompuation.

Precomputation-based volume GI rendering. Most existing PRT-based methods deal with the light transfer between rigid surfaces, and recently some advanced techniques can additionally handle deformable scenes [19]. However, precomputation-based GI rendering for volume data sets has received far less attention. Wyman et al. [20] introduced a method to precompute the GI using SH for interactive rendering of isosurfaces extracted from volume data sets. Their method supports direct lighting, volumetric shadows, and indirect lighting effects with interactively changing viewpoints, lighting, and isovalues. Still, the complex scattering behavior inside the volume is ignored. Sharing the same motivation, Beason et al. [21] proposed an enhanced isosurface GI rendering method that further supports translucency and caustics, but this method is limited to static lighting. Ritschel [22] incorporated a GPU-based hierarchical visibility approximation into the precomputation to quickly compute the visibility inside the volume, but it only handles low-frequency effects. In comparison, our PVRT method can generally support various volume GI effects with few constraints for the input volume data set. Furthermore, the precomputation-based routine has also been adopted in the offline rendering of the MS effect in hair [23].

Real-time volume GI rendering. Kaplanyan and Dachsbacher [24] developed a cascaded light propagation volume (LPV) method to approximate indirect lighting very efficiently, which can further be extended to render SS effects in volumetric media. However, the LPV method only supports low-frequency effects in coarse resolution volumes, and it is nontrivial to support complex lighting with this method. For optically thick media, like smoke, the MS effect, which is much more costly compared with SS, dominates the overall appearance. Recently, Engelhardt et al. [25] suggested to create a set of virtual point lights (VPLs) within the medium and combine the SS contributions from all VPLs to render MS in real time. Unfortunately, the limited number of VPLs usually leads to low-frequency effects. Several GPU-accelerated approaches [26], [27] have been proposed to render high-quality volumetric smoke under low-frequency environment light. For comparison, our PVRT method can render the MS effect under an all-frequency environment light in real time.

Hernell et al. [28] presented an efficient method to compute ambient and emissive tissue illumination, which exploits ray casting to determine ambient illumination and adopts a multiresolution volume to speed up rendering. Similarly, to render simple scattering and ambient occlusion for isosurfaces, a GPU-based Monte Carlo ray casting method was proposed in [29]. When compared with these methods, our PVRT method is superior in its support for complex scattering, especially MS effects, for general volume data formats under complex lighting conditions. Kniss et al. [30] proposed a shading model to interactively render volumes, which generates volumetric shadows and forward scattering through translucent materials. Ropinski et al. [31] presented a physically more accurate optical model for volume rendering, which incorporates scattering, shadowing, and specular reflections. However, applying these volume shading models to handle complex environmental lighting and general scattering effects is nontrivial. To render high-quality volume shadows, Kronander et al. [32] presented a method that encodes local and global volumetric visibility into an efficient multiresolution grid over the extent of the volume to achieve real-time performance. Compared with it, our PVRT method can generate various volume GI effects, not just shadow, in real time.

3 Radiance Transport in Volumetric Media

Before the introduction of our PVRT method, we briefly review radiance transport in volumetric media. Following the definition in [5], the radiance quantities that we use are listed in the table of Fig. 3. The input volume data are modeled as a sequence of volumetric density fields. Regular mesh grids are used to store the volumetric density field at cell centers. Considering a 3D position p in the volumetric media (shown in Fig. 4), we represent the volume density field of p at each frame as $D(p)$, the absorption coefficient as $\kappa_a(p)$, the scattering coefficient as $\kappa_s(p)$, the extinction coefficient as $\kappa_t(p) = \kappa_a(p) + \kappa_s(p)$, and the scattering albedo as $\Omega(p) = \frac{\kappa_s(p)}{\kappa_t(p)}$. Our method supports both environmental and local directional lighting. We use L_{in} to represent the incident radiance from the light source.

Fig. 4 illustrates the radiance transport in volumetric media under environmental lighting. For viewpoint x, the radiance reaching x along direction ω_o is composed of two parts: the reduced incident radiance L_{ri} and the media radiance L_m:

$$L_{out}(x, \omega_o) = L_{ri}(x, \omega_o) + L_m(x, \omega_o). \tag{1}$$

The reduced incident radiance $L_{ri}(x, \omega_o)$ describes the source illumination $L_{in}(\omega_o)$ that arrives directly at x along direction ω_o with some attenuation by the medium:

$$L_{ri}(x, \omega_o) = \tau_\infty(x, \omega_o) L_{in}(\omega_o), \tag{2}$$

where $\tau_\infty(x, \omega_o)$ is the transmittance from infinite lighting to x along the direction ω_o, which can be computed as $exp(-\int_x^{\infty_{\omega_o}} \kappa_t(u) D(u) du)$. The media radiance $L_m(x, \omega_o)$ represents the radiance energy that has scattered one or multiple times in the medium before arriving at x along the direction ω_o. It is computed by integrating radiance contributions along the view ray:

$$L_m(x, \omega_o) = \int_{p_{in}}^{p_{out}} \tau(p, x) \kappa_t(p) D(p) J(p, \omega_o) dp, \tag{3}$$

where source radiance $J(p, \omega_o)$ represents the light that scatters at point p toward direction ω_o. In usual nonemissive media, such as smoke, fog, and so on, this source radiance consists of an SS term J_{ss} and an MS term J_{ms}:

$$J(p, \omega_o) = J_{ss}(p, \omega_o) + J_{ms}(p, \omega_o). \tag{4}$$

As shown in Fig. 4, the SS term J_{ss} represents the reduced incident radiance L_{ri} whose first scattering interaction in the medium occurs at p:

$$J_{ss}(p, \omega_o) = \frac{\Omega(p)}{4\pi} \int_{\mathbb{S}} L_{ri}(p, \omega_i) f(p, \omega_o, \omega_i) d\omega_i, \tag{5}$$

where $f(p, \omega_o, \omega_i)$ is the phase function that defines the scattering distribution of the current media, and $\mathbb{S}$ represents the set of different incident directions ω_i that distributes on the local sphere centered at p. Same as (2),

$L_{ri}(p, \omega_i)$ can be computed as $\tau_\infty(p, \omega_i)L_{in}(\omega_i)$. Notice that $L_{ri}(x, \omega_o)$ can also be regarded as a special SS case. $L_{ri}(x, \omega_o)$ and $J_{ss}(p, \omega_o)$ together contribute to the *direct illumination* effects for the volume.

The MS term J_{ms} represents the radiance that has been bounced more than one time before arriving at p, which accounts for the GI effects. In Fig. 4, the red arrows represent the incident media radiance L_m contributed from MS:

$$J_{ms}(p, \omega_o) = \frac{\Omega(p)}{4\pi} \int_{\mathbb{S}} L_m(p, \omega_i) f(p, \omega_o, \omega_i) d\omega_i. \qquad (6)$$

Similarly to surface shading, the computation of GI effects is much more time-consuming when compared with direct illumination, so MS always becomes the bottleneck of the volume GI rendering.

4 Algorithm Overview

Like PRT, the PVRT method includes a precomputation stage and a rendering stage, as shown in Algorithm 1.

Algorithm 1. Pseudocode of the PVRT method

```
1  Define final view radiance $L_{out} = 0$
2  **Precomputation**:
3  Approximate the lighting using certain basis
   (SH or DSF).
4  FOR each basis light:
5    Part1: Direct illumination
6      Sample $N_s$ directions uniformly.
7      FOR each voxel:
8        Accumulate $L_{ri}$ (Eq.2) from each direction.
9        Compute radiant exitance vector.
10     ENDFOR
11   Part2: Global illumination
12       Trace photons through volume grids.
13       Splat absorbed photon's energy into
         neighbor voxels.
14       FOR each voxel:
15         Accumulate the photon's energy.
16         Compute radiant exitance vector.
17       ENDFOR
18 ENDFOR
19 Compress the radiant exitance matrices of all
   voxels using CPCA.
20 **Rendering**:
21 Project the current lightings into basis space.
22 FOR each CPCA cluster:
23   FOR each basis transfer matrix:
24     Compute view-dependent radiance.
25   ENDFOR
26 ENDFOR
27 Reconstruct and store radiance volume.
28 GPU-based ray marching to evaluate $L_{out}$.
```

Precomputation. The precomputation stage is involved with both direct and GI. To achieve efficient volume GI effects, one of our contributions is the seamless integration of the VPM method [2] into the precomputation stage, including photon traversal through the volume as well as the evaluation and compression of the radiance transfer matrices for each voxel. To achieve this goal, there exist three major challenges in the precomputation stage.

The first challenge is the selection of the basis functions for representing radiance from high-frequency incident lighting. As aforementioned, SH can only handle low-frequency lighting. To approximate higher frequency lighting, spherical wavelet basis functions (SWBFs) [17] or SRBFs [4] are good candidates. However, both have disadvantages as well. The SWBFs are not rotationally invariant, which prevents fast rotation of the basis functions. The SRBFs can be freely rotated but their construction and evaluation are quite slow, because the SRBFs do not have the interpolation property and local support. The interpolation property means the coefficients are equal to the spherical function values at the sampling directions, so that reconstructing the spherical function can be rapidly achieved by interpolation. The basis functions with local support ensure that only a small number of coefficients are needed for evaluation of the spherical function at arbitrary directions. To fulfill these properties, we introduced a new DSF that is rotationally invariant, interpolation based, and also has locally supported basis functions. The DSF can achieve similar accuracy to the SRBFs for approximating high-frequency environment light.

The second challenge is how to cast photons. Photon casting is simple if all the light sources and geometries are clearly defined. However, in the PRT pipeline, the lighting environment is approximated using a set of basis functions, such as SH and DSF. The values of these basis functions are distributed nonuniformly, and the values can be either positive or negative. Therefore, we must design a general photon casting scheme for different types of basis functions. To generate photons with correct energy distributions, instead of casting photons from distant planes facing random directions with direction-dependent casting rate, we cast photons from the boundary surface of the volume, and each photon has a random direction and also a direction-dependent energy value. This assures that all the generated photons will go into the volume, and the energy distributions will converge to the spherical basis functions faster than a normal implementation. Such a photon casting strategy has been tested and worked well for both SH and DSF.

Another challenge is the efficient radiance transfer estimation using VPM. For classic PRT rendering, we can directly integrate the incident radiance on the surface over a hemisphere using backward ray tracing. Since photon mapping is a forward light-tracing technique, a photon gathering step is required during the radiance estimation within the volume. If using an unmodified VPM that stores the photons inside a kd-tree, the computational cost for approximating the volume radiance transfer is extremely high. This is because the spherical numerical integrations for all pairs of irradiance and radiant exitance basis functions have to be considered. Moreover, sampling photons from the kd-tree is difficult to parallelize, and memory efficiency is low. To address such a situation, we introduce a fast splatting-based radiance transfer estimation technique, wherein each photon's energy is splatted

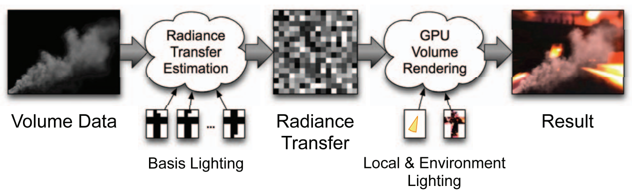


Fig. 2. The pipeline of the PVRT method for rendering MS.

and accumulated into its neighbor voxels for further numerical integration.

Rendering. To achieve real-time volume rendering, there are also some technical challenges. Volume rendering usually demands lots of samples along the rays, which is fast for simple scalar fields or low-dimensional vector fields. However, combined with the PRT framework, evaluating each sample requires the processing of hundreds of variables. For example, if we use sixth order SH for irradiance and fourth order SH for radiant exitance, the size of the radiance transfer matrix is $36 \times 16 = 576$, and the size of the light vector is 36. Then, we need to perform a matrix-vector multiplication of this size for each sample, and it is impossible to evaluate it in real time for all voxels. Therefore, we introduce an early evaluation technique to significantly reduce the rendering cost. It is achieved by using clustered principal component analysis (CPCA) for transfer matrix compression and reducing the compressed basis transfer matrices into scalar values based on view and lighting changes before decompression. The technical details are presented in Section 7.

In general, as shown in Fig. 2, the PVRT method approximates the lighting using a set of basis lights, and each basis light corresponds to a basis function. During the precomputation stage, aided by the VPM technique, the radiance transfer estimation is performed to compute the radiant exitance for each voxel. To save storage and improve performance, we further compress the radiant exitance data for the whole volume. Thereafter, during the rendering stage, the precomputed radiant exitance data are efficiently decompressed and used to compute the final view-dependent radiance for each voxel. The PVRT method supports both environment light and directional light.

p	A 3d point within medium
p_{in},p_{out}	Entry/Exit point in medium
x	A 3d point
ω_i, ω_o	Incident, outgoing radiance direction
S	Sphere of directions
$D(p)$	Volume density field at p
$\kappa_s(p)$	Scattering coefficient
$\kappa_a(p)$	Absorption coefficient
$\kappa_t(p)$	Extinction coefficient
$\Omega(p)$	Albedo of participating medium
$\tau(p, x)$	Transmittance from u to x
$\tau_\infty(p, \omega)$	Transmittance from infinity to x
$f(u, \omega_o, \omega_i)$	scattering phase function at u
$L_{in}(\omega)$	Incoming radiance from light source
$L_{out}(x, \omega)$	Outgoing radiance arriving at x
$L_{ri}(x, \omega)$	Reduced incident radiance
$L_m(x, \omega)$	Media radiance
$J(x, \omega)$	Scattered radiance
$L_{re}(\omega)$	Radiance exitance function
T	Radiant exitance matrix

Fig. 3. Nomenclature.

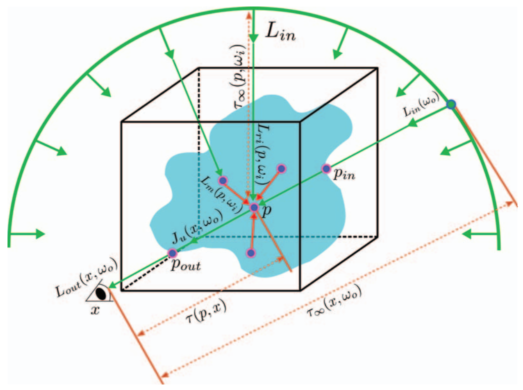


Fig. 4. Illustration of the radiance transport in volumetric media.

5 Discrete Spherical Function

Before introducing the PVRT method, we first present the DSF that has both the interpolation property and local support.

5.1 Definition

We sample a set of uniformly distributed directions $\Omega = \omega_1, \omega_2, \ldots, \omega_n$ over a unit sphere using icosahedral mesh subdivision, as shown in Fig. 5. Using s_k to represent the sampled value at direction ω_k, the DSF for an arbitrary direction ω is defined as

$$S(\omega) = \sum_{k=1}^{n} \frac{W_h(\omega, \omega_k)}{\sum_{i=1}^{n} W_h(\omega, \omega_i)} s_k = \sum_{k=1}^{n} \bar{w}_k(\omega) s_k, \qquad (7)$$

where $\bar{w}_k(\omega)$ is the DSF basis function, which represents a normalized weight. $W_h(\omega_i, \omega_j)$ is a symmetric kernel function, and it is defined as

$$W_h(\omega_i, \omega_j) = \begin{cases} \left(h^2 - |\theta_{ij}|^2\right)^3, |\theta_{ij}| < h \\ 0, |\theta_{ij}| \geq h, \end{cases} \qquad (8)$$

where $\theta_{ij} = \cos^{-1}(\omega_i \cdot \omega_j)$ is the angle between any two directions ω_i and ω_j. h is defined as $h = \min_{i,j} \theta_{k_i k_j}$, which is the minimum angular distance between any pair of sampled directions ω_{k_i} and ω_{k_j} in Ω. As shown in Fig. 5, h can be regarded as the radius of the local impact region of each

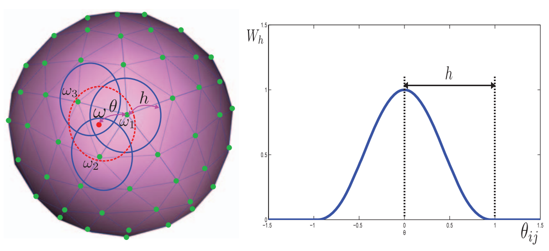


Fig. 5. Illustration of the DSF.

sampled direction. The definition of W_h follows the kernel definition in smoothed particle hydrodynamics [33], which is widely applied for fluid simulation. Compared with the Gaussian kernel, it has been proven to be more efficient for evaluation and has compact support in local region.

With such a definition, the kernel function W_h has the following properties:

- *Local support*

$$W_h(\omega_i, \omega_j)\begin{cases} > 0, \theta_{ij} < h \\ = 0, \theta_{ij} \geq h. \end{cases}$$

  As shown in Fig. 5, W_h provides compact support in local region.
- *Monotonicity*

$$\frac{dW_h(\omega_i, \omega_j)}{d\theta_{ij}} \leq 0.$$

- *Interpolation.* Based on the definition of h, if we pick up any two sampled directions ω_{k_i} and ω_{k_j} in Ω, then

$$W_h(\omega_{k_i}, \omega_{k_j}) = \begin{cases} 1, \ k_i = k_j \\ 0, \ k_i \neq k_j. \end{cases}$$

  Therefore, W_h guarantees that when $\omega = \omega_k$, $S(\omega_k) = s_k$, and the DSF function can recover the sampled original value. When ω is located between several samples, as shown in Fig. 5, its value is the interpolation of its neighbor samples (inside red dot circle) weighted by W_h.

Equation (7) actually represents the reconstruction of the spherical function using DSF, and s_k can be regarded as the corresponding coefficients for the DSF bases. For other bases, such as SH and SRBF, computing the coefficients is a process of projecting the spherical function onto basis space and such a projection step is usually time-consuming. However, for the DSF, s_k is just a sampling of the spherical function, so the projection step is very efficient. Actually, computing s_k is not a simple point sampling, and to balance the frequency between the spherical function and the DSF, we introduce a prefiltering step for the spherical function, which will be discussed in Section 8. Relying on the local support property of DSF, the evaluation of (7) for arbitrary directions can be efficiently computed because only a limited number of bases need to be considered. The definition of the DSF basis $\bar{w}_k(\omega)$ is a normalized weight for local interpolation, so $\bar{w}_k(\omega)$ is invariant during the rotation of the spherical function. Therefore, when rotating the environment light, we only need to rotate the prefiltered environment light and sample it again to get the new coefficient s_k, which is simple and efficient.

5.2 Lighting Approximation Using DSF

To approximate the lighting using this basis function, we project the original lighting function onto basis space and compute the corresponding coefficient for each basis. With these coefficients, afterward we can reconstruct the value of the lighting function at arbitrary directions by evaluating the basis functions. Comparing the reconstructed value with the original value can help us verify the accuracy of the lighting approximation. We perform such a lighting reconstruction experiment [16] to test the DSF.

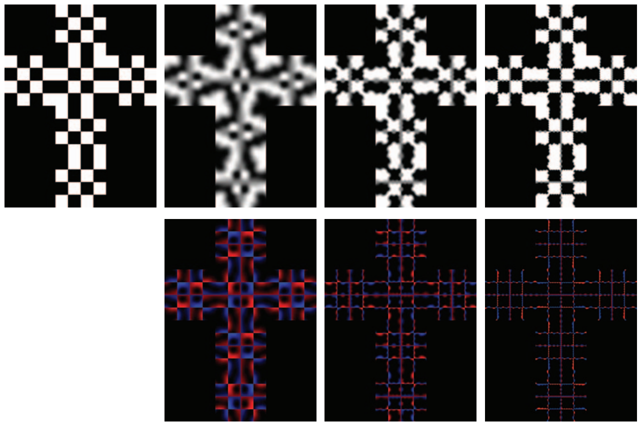

(a) Input EM (b) $162, 13.2\%$ (c) $642, 8.9\%$ (d) $2562, 7.3\%$

Fig. 6. Comparison of the approximation accuracy using different number DSF bases for a synthetic high-frequency EM (a). The first row lists the reconstruction results, and the second row shows the approximation errors ("red" and "blue" represent the positive and negative values, respectively.). The number of DSF bases and L^2 errors are given for (b), (c), and (d).

The initial directional sampling of the DSF is based on icosahedral mesh generation, and the number of samples over the sphere Ω can be $42, 162, 642, 2,562\ldots$ depending on the order of icosahedral subdivision. To select the optimal number of the DSF bases to achieve a good balance between approximation quality and performance, we reconstruct a synthetic high-frequency "checkerboard" environment map (EM) with different numbers of DSF bases and compare the results using standard L^2 errors between the pixels in the original cube map and the pixels in the reconstructed one. As shown in Fig. 6, the reconstruction quality essentially improves by increasing the number of DSF bases. Notice that using 2,562 bases rather than 642 bases only improves quality by a factor of 1.6 percent, but multiplies the total number of basis lights by 4. Therefore, we choose the optimal number of DSF bases to be 642.

To compare the approximation quality between the DSF and other existing bases, in Fig. 7, we test the lighting reconstruction for both synthetic and real EM lights using different bases with a similar number of basis functions. Clearly, DSF can achieve the best approximation quality with lowest L^2 error. To verify the high-frequency rendering effects using different bases, in Fig. 8, we test the shadow effects under directional light. It is obvious that using a low-frequency SH basis misses the high-frequency shadow details and the ringing artifacts of SH darken the whole scene. Both DSF and SRBF achieve good high-frequency shadow effects.

The statistical results in our experiments show that dealing with an arbitrary EM light with size $6 \times 256 \times 256$, for each SH, SRBF, and DSF basis, the average timing of the projection step is 539, 91, and 2 ms, respectively, and the average timing of evaluation for an arbitrary direction is 0.369, 0.059, and 0.013 ms, respectively. It is clear that SH is

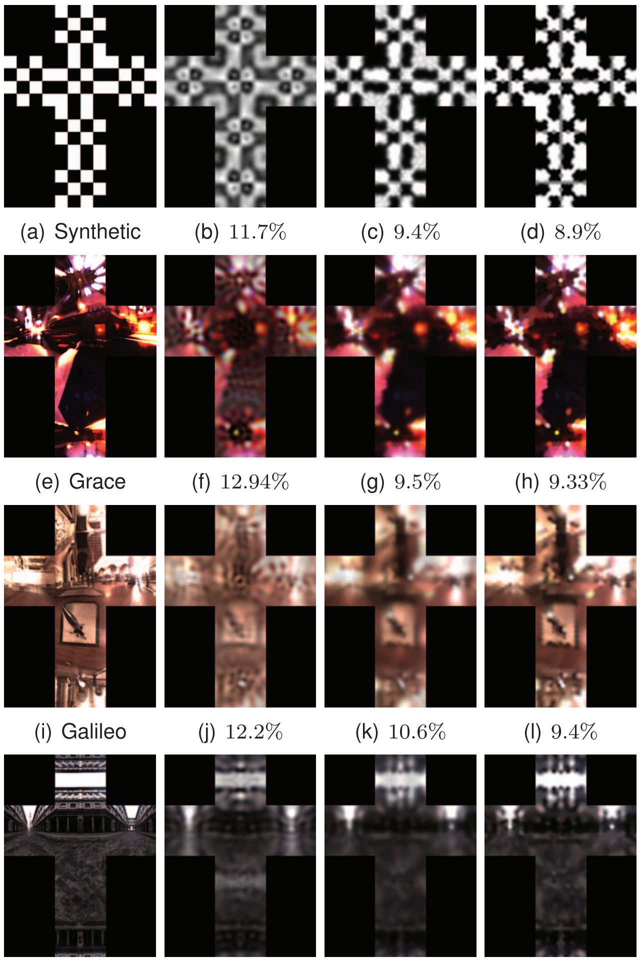


Fig. 7. Comparison of the approximation results for the synthetic (a) and real EMs (e), (i), (m). The second column uses the 26th order SH with 676 coefficients, the third column uses SRBF with 642 coefficients, and the fourth column uses DSF with 642 coefficients. The L^2 errors are given for all the reconstructed results.

the most time-consuming basis for both projection and evaluation, and hence, normally, we should not use many SH bases to approximate high-frequency functions. SRBF computation involves numerical integration [4], and therefore, its time cost is much higher than the DSF, especially for the projection step. The DSF performs very efficiently for both steps because its interpolation property and local support ensure fast light projection and evaluation.

6 Precomputing Radiant Exitance

In this section, we describe the precomputation stage of the PVRT method in detail. Since the environment light is more challenging, we use it for explaining the algorithmic steps.

6.1 Radiance and Lighting Approximation

As described in [3], a low-frequency spherical function can be approximated using sth order SH

$$L(\omega_i) = \sum_{l=0}^{s-1} \sum_{t=-l}^{l} c_l^t Y_l^t(\omega_i), \tag{9}$$

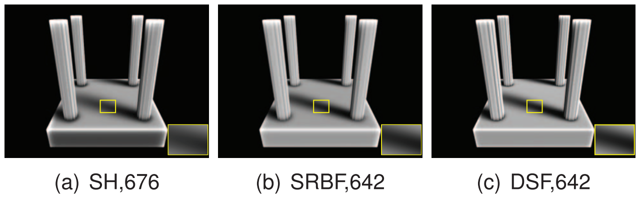


Fig. 8. Comparison of the shadow reconstruction accuracy using different bases.

where Y_l^t are the orthogonal real SH basis functions and c_l^t are their corresponding coefficients. The radiant exitance of each voxel is actually a 2D spherical function that has a corresponding value for each direction over the unit sphere based on the phase function within the current voxel. Arbitrary phase functions are supported for computing radiant exitance in our method. However, storing the radiant exitance values can be challenging. Based on the fact that MS inside the volume exhibits low-frequency behavior [1], we choose the SH basis to approximate the 2D spherical function of the radiant exitance for each voxel. Depending on the current phase function, the selected order of SH varies. Following [26], we adopt the first order SH for the isotropic phase function and fourth to sixth order SH for the anisotropic phase function.

The low-frequency environment light can also be well approximated using the low-order SH (i.e., $l \leq 6$) basis [3]. For the high-frequency environment light, we use the aforementioned DSF for lighting approximation, and the radiance from arbitrary direction ω_i can be represented as

$$L(\omega_i) = \sum_{k=1}^{n} \bar{w}_k(\omega_i) s_k, \tag{10}$$

where $\bar{w}_k(\omega_i)$ is the DSF basis function and s_k is the radiance value at the kth presampled direction. After the approximation, the environment light is converted into a set of basis lights, and each basis light can be treated as distant spherical lighting.

6.2 Direct Illumination

For direct illumination, the incident energy for each voxel is just the reduced radiance L_{ri} (2), so photon tracing is unnecessary for its precomputation. For a voxel at position p, when using the SH or DSF basis to approximate incident lighting, the incident reduced radiance in ω_i from each basis light can be computed as $L_{ri}(p,\omega_i) = \tau_\infty(p,\omega_i) Y_l^m(\omega_i)$ or $L_{ri}(p,\omega_i) = \tau_\infty(p,\omega_i)\bar{w}_k(\omega_i)$. In the precomputation stage, for each basis light, we uniformly sample N_s directions either over a unit sphere (for SH) or inside a small solid angle around the current basis light (for DSF). Considering all the L_{ri}s from N_s directions, each L_{ri} is multiplied with the phase function value at the current voxel position to get the radiant exitance L_{re}. As mentioned above, L_{re}, which is a 2D spherical function, is nontrivial to store directly. Therefore, we use SH to approximate L_{re} with good accuracy. For each basis light, the radiant exitance of each voxel can thus be approximated and stored as a vector. After processing all basis lights, the radiant exitance of each voxel finally forms a *radiant exitance matrix*. Methods for determining the value of N_s will be discussed in Section 8.

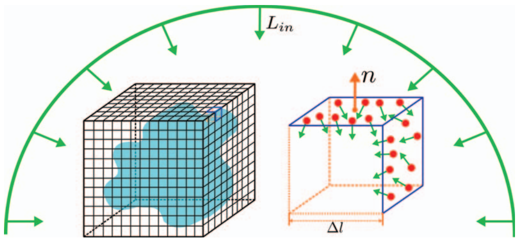


Fig. 9. Illustration of the volume photon sampling.

6.3 GI Using VPM

In this section, we focus on how to address the second and third challenges that are mentioned in Section 4 for the seamless incorporation of the VPM method into the PRT framework and the achievement of efficient precomputation for the rendering of volume GI effects.

6.3.1 Volume Photon Sampling under Spherical Lighting

As is the case for the standard VPM method, the first step is to generate the photons for each basis light. Our basic idea is to cast photons at the boundary cells (voxels) of the volume with varying flux energy for each photon. The photon generation fulfills the following requirements: 1) the initial directions of all the generated photons in each boundary voxel are uniformly distributed on the unit sphere, and 2) each photon's energy is proportional to the radiance from the basis light in the photon's direction. As shown in Fig. 9, for each boundary voxel in the volume, we generate k photons from the boundary surfaces of the voxel. Each photon starts from a random position over the boundary surface and has a random direction ω that is uniformly distributed over the sphere. For each photon under SH basis light Y_l^t, the flux energy is

$$\Phi(\omega) = Y_l^t(\omega)\Delta l^2(\omega \cdot \mathbf{n})d\omega, \tag{11}$$

where Δl and $\mathbf{n}$ are the side length and the outward normal of the boundary voxel, respectively. For each DSF basis light $\bar{w}_k$, the flux energy of each photon is

$$\Phi(\omega) = \bar{w}_k(\omega)\Delta l^2(\omega \cdot \mathbf{n})d\omega. \tag{12}$$

Some of the generated photons may immediately exit the volume region. Such photons are discarded before further computation. It is obvious that the number of generated photons increases as the resolution of volume data increases. To reduce the number of photons when dealing with high-frequency lighting, rather than distributing photon's direction ω over the whole unit sphere, we may restrict ω inside a small differential solid angle for each DSF basis light. Therefore, although the number of DSF bases is usually much higher than the number of SH bases, much fewer photons are actually needed for each DSF basis light.

6.3.2 Volumetric Photon Tracing

Starting from the boundary voxel, the photons travel through the volume with a predefined step size. At each step, a photon can either pass unaffected through the medium inside current voxel, or be scattered or absorbed. We choose the side length of the voxel Δl as step size for photon propagation so that the maximum number of potential steps matches the volume resolution. The probability of an absorption or scattering event occurring at point p is integrated along the photon path from the entry point p_{in} to p:

$$P(p) = 1 - \exp\left\{-\int_{p_{in}}^{p}(\sigma_a + \sigma_s)\rho(x)dx\right\}, \tag{13}$$

where ρ is the density field function for volume media, which is often obtained from the original volume data through a user-defined mapping. σ_a and σ_s are absorption and scattering coefficients, respectively. When a random number $0 \le X \le 1$ satisfies $X \le P$, Russian Roulette is used to determine whether it is an absorption or scattering event based on the probability values $\sigma_a/(\sigma_a + \sigma_s)$ and $\sigma_s/(\sigma_a + \sigma_s)$. The new scattering direction of a photon is chosen using importance sampling based on the phase function within the current voxel [2], and arbitrary phase functions can be supported in our method.

6.3.3 Photon Splatting for Radiance Estimation

To avoid creating a hierarchical data structure to store photons [2], rather than gathering neighbor photons' energy for each voxel, the PVRT method chooses to splat each photon's energy into its neighbor voxels [14]. For an absorbed photon p with direction ω, its energy $\Phi_p(\omega)$ (11) or (12) is splatted into the photon's neighbor voxels weighted by the distance between the photon and voxel. Such a splatting process has an equivalent effect to the standard gathering process [2]. To ensure both efficient evaluation and compact support in local regions, again we adopt the polynomial kernel [33] as the weighting function:

$$W(d) = \begin{cases} \frac{315}{64\pi h^9}(h^2 - |d|^2)^3, & |d| < h \\ 0, & |d| >= h, \end{cases} \tag{14}$$

where d is the distance between the voxel and the photon p. $h = c\Delta l$ is the kernel radius that is set based on the grid resolution, and c is a user-defined constant (e.g., $c = 4$).

After the splatting, we compute the radiant exitance for each neighbor voxel based on its phase function $f(x, \omega_o, \omega_i)$. With the weighted photon energy $W(d)\Phi_p(\omega)$ and photon's direction ω, the radiant exitance of each neighbor voxel can be represented as $L_{re}^p(\omega_o) = W(d)\Phi_p(\omega)f(x, \omega_o, \omega)$. After processing all the photons, the final radiant exitance of each voxel is $L_{re}(\omega_o) = \sum_{p=0}^{Q} W(d)\Phi_p(\omega)f(x, \omega_o, \omega)$, where Q is the number of the neighboring photons. Again, we use SH to approximate $L_{re}(\omega_o)$ and the radiant exitance of each voxel is approximated and stored as a vector. With the radiant exitance for each voxel, we can evaluate the incident radiance for arbitrary view rays in the subsequent volume ray casting process.

6.4 The Radiant Exitance Matrix and Its Compression

For each voxel, the radiant exitance under each basis light is represented as a vector with length n. After processing m basis lights, the radiant exitance from all the basis lights forms a $m \times n$ matrix T, called the *radiant exitance matrix*, which can be regarded as a linear transformation that transforms the input lighting vector L_{in} into final radiant exitance vector I_e:

$$I_e = TL_{in}. \tag{15}$$

Here, the length of L_{in} and I_e is m and n, respectively, which represent the number of used bases. Computing T for each voxel is the major task during the precomputation stage. If using a sixth order SH for the environment light and a fourth order SH for the anisotropic phase function (i.e., $m = 36$ and $n = 16$), the dimension of T will be 36×16. Storing a 36×16 matrix for each voxel is impractical, so we need to further compress the precomputated data.

We adopt the CPCA method [34] to achieve further compression. All the matrices of the voxels are first clustered in signal space using vector quantization. Note that empty voxels with zero density gradients are excluded from the clustering, and zero weights are assigned directly. The total number of clusters is adjustable by compromising between quality and speed, and a relatively optimal cluster number is 128 based on our experiments (Section 9). Thereafter, within each cluster, the standard principal component analysis (PCA) is performed to compute the mean matrix, the set of the basis matrices, and the set of corresponding coefficients for each voxel's radiant exitance matrix. Using CPCA, the transfer matrix T at a certain voxel t can be approximated through

$$T \approx T^k_{mean} + \sum_{i=1}^{N} w^t_i T^k_i, \tag{16}$$

where k is the index of the cluster that t belongs to, T_{mean} is the mean matrix of the kth cluster, T^k_i is the ith PCA basis matrix, and w^t_i is the corresponding coefficient. N is the number of PCA basis transfer matrices, and our experiments demonstrate that $N = 4$ works well in most cases (Section 9). Putting (16) into (15), the linear radiance transformation at each voxel is approximated as

$$I_e \approx \left(T^k_{mean} + \sum_{i=1}^{N} w^k_i T^k_i \right) L_{in}. \tag{17}$$

After the CPCA compression, the storage of the precomputation results of each voxel includes only one integer cluster index k and $N = 4$ PCA floating-point coefficients. For each matrix cluster, we also store the mean matrix T_{mean} and $N = 4$ basis matrices. Overall, such a strategy saves lots of storage and also ensures real-time performance for further radiance estimation in the ray casting step. The accuracy of our compression strategy is discussed in Section 9.

7 Real-Time Volume Rendering

The major steps in the real-time rendering stage consist of 1) view-dependent volume radiance estimation, and 2) volume ray casting. All the operations in the rendering stage can be efficiently performed on the GPU.

7.1 Lighting Approximation

Incident lighting is projected into basis function space, either SH or DSF, and can be represented as a vector of projection coefficients L_{in}. This is a standard step in PRT-based methods and is performed in each frame.

7.2 View-Dependent Volume Radiance Estimation

After updating the lighting environment, in each cluster k, we multiply the new light coefficient vector L_{in} to the mean matrix T^k_{mean} as well as all the basis transfer matrices T^k_i:

$$I^k_{mean} = T^k_{mean} L_{in} \quad I^k_i = \sum_{i=1}^{N} T^k_i L_{in}, \tag{18}$$

where I^k_{mean} and I^k_i represent the transferred mean and basis radiant exitance vector of the kth cluster, respectively. The final radiance vector I_e at voxel t can be recovered by

$$I_e = I^k_{mean} + \sum_{i=1}^{N} w^t_i I^k_i, \tag{19}$$

where w^t_i is the PCA coefficients for voxel t. The straightforward strategy is to store the I_e for each voxel for further radiance estimation. However, I_e is a vector of length n and directly saving it for all voxels requires $n \times Size_{volume}$ GPU memory, which is costly. Furthermore, evaluating (19) is also inefficient for all the voxels in each frame. Notice that in each frame, the current view info is known before volume ray casting. As shown in Lines 22-26 in Algorithm 1, for the kth cluster, we can first compute the view-dependent radiance value L^k_V for I^k_{mean} and each I^k_i, respectively, by evaluating the basis functions with the view direction. When using the SH basis, I^k_V is computed as (using (9)):

$$L^k_V = \sum_{l=0}^{s-1} \sum_{t=-l}^{l} I_{l(l+1)+t+1} Y^t_l(\omega_{x \to x_V}), \tag{20}$$

when using DSF basis, it is computed as (using (10)):

$$L^k_V = \sum_{k=1}^{n} \bar{w}_k(\omega_{x \to x_V}) s_k, \tag{21}$$

where $x \to x_V$ is a unit vector from voxel position x to current viewpoint x_V. Thereafter, for each voxel, its view-dependent radiance value L_V can be efficiently computed as the linear combination of the L^k_Vs weighted by the PCA coefficients of current voxel. With such an early evaluation strategy, we only need to save one floating point value for each voxel, and the storage for all the voxels is reduced to be just one additional *radiance volume* texture.

7.3 Volume Ray Casting

The aforementioned radiance estimation for each voxel leads to a very simple ray casting stage. The traditional GPU-based volume ray casting method [35] can be applied to render the volume data. In addition, we just need to sample the radiance L_V from the *radiance volume* texture at each sampling point along the ray.

8 Implementation

In this section, we will discuss several issues and the corresponding solution strategies we use for implementation.

8.1 EM Prefiltering for DSF

We adopt 642 DSF bases to approximate the high-frequency lighting. However, the size of input EM usually contains more than 642 pixels. For example, a standard EM might have $6 \times 256 \times 256$ pixels. Sampling 642 values discretely from the EM will probably miss important lighting energy, especially for high-frequency lighting. Therefore, to match the frequency between the EM and DSF, we introduce a prefiltering step for the EM. The directions of DSF bases are uniformally distributed over the sphere (Fig. 5), and since each EM pixel corresponds to a direction, we can map the DSF basis direction to the pixels in EM. To ensure the prefiltering results cover the whole spherical domain, we choose a spherical Gaussian with bandwidth equal to minimum angular distance h (defined in (8)). To perform the prefiltering, we use an image space Gaussian centered at the pixel corresponding to the DSF basis direction with the bandwidth computed by mapping h into image space. The prefiltering step can be computed in real time. After the prefiltering, each pixel value in the prefiltered EM is a coefficient for a DSF basis. The prefiltering step only needs to be performed once unless the content of input EM changes (e.g., video EM). When rotating the EM lighting, we just need to rotate the prefiltered EM and then sample it directly to get the new coefficients after rotation.

8.2 Directional Light

It is well known that the directional light can be easily represented by the SH bases evaluated at the light direction. Approximating the local directional light using DSF, we only need to find the three nearest DSF directions for the current light direction based on angular distance and compute three corresponding weights using barycentric interpolation. Then, the directional light can be approximated by the linear combination of the three DSF basis lights.

8.3 Number of Photons for Each Voxel

In the PVRT method, the photons are emitted from boundary voxels, and the number of photons per voxel N_P is controlled by the user. N_P affects the rendering quality, and it scales well as the resolution of the input volume data set changes. For example, if we set N_P to be 100, a 100^3 volume would cast $6 \times 100^3 = 6$ million photons in total because there are only 6×100^2 boundary voxels. As discussed in Section 6.3.1, when using DSF N_P is much smaller compared with using SH. Based on our experimental results, we found that $N_P = 10$ and $N_P = 100$ achieve a good balance between quality and speed for DSF and SH, respectively. More results and discussions will be shown in Section 9.

8.4 Performance Optimizations

There are several performance optimization strategies: 1) When computing direct illumination during precomputation, each voxel marches $N_s = 642$ rays through the volume to gather incident lighting. If the volume resolution is 512^3, the total number of ray samplings will be 86 billion. To accelerate this process, we skip all the empty voxels ($D(p) \leq 0$). Also, a ray is terminated early if a high cumulative density is reached during its marching procedure. 2) We use precalculated lookup tables to store SH basis function values instead of evaluating them at runtime, which greatly accelerates the photon splatting operation and radiance integration. 3) In the precomputation stage, the computation for different basis lights and different photons is highly independent. Therefore, it is easy to parallelize the computation of each individual case (i.e., certain basis light) using the evolving many-core hardware. Our CUDA-based parallel implementation for precomputation achieves more than two orders of magnitude speed improvement over the single-threaded CPU implementation.

TABLE 1
Performance Data

VolData	DataRes	NumP (mil.)	Prep (sec.)	RV (ms)	Render (ms)
Smoke Plume	$300 \times 300 \times 150$	3.6	110	0.8	20
Smoke Wall	$400 \times 300 \times 100$	3.8	165	0.8	23
Aneurism	$256 \times 256 \times 256$	3.9	81	0.9	19
Bonsai	$256 \times 256 \times 256$	3.9	113	0.9	20
Bone	$512 \times 512 \times 512$	7.9	241	1	23
Part	$504 \times 504 \times 228$	9.7	395	1	25
Machine Room	$417 \times 345 \times 60$	3	73	0.7	21

8.5 System Integration

It is easy to integrate our method into existing rendering frameworks. Our renderer consists of two modules including a preprocessor and a real-time renderer. The preprocessor is called once, which takes the density volume together with several parameters as input and outputs the volume radiance transfer data. Then, the real-time renderer takes the volume data, the volume radiance transfer data as well as other rendering parameters as input and outputs the final image. In our implementation, we fit our renderer into an OpenGL-based application framework and created several graphics user interfaces for adjusting the parameters.

9 Results and Discussions

In this section, we report on the accuracy and performance of the PVRT method. Further, we compare the results of our method with the references generated by the VPM [2]. Our method is implemented in OpenGL and CUDA [36]. All results were rendered on a laptop equipped with an Intel i7 740QM 1.73-GHz CPU, a NVIDIA Geforce GTX 460M GPU and 6-GB of physical memory. All the results, if not specifically mentioned, utilize DSF with 642 bases to approximate incident lighting. The number of CPCA clusters is 128, and the number of PCA bases is 4.

The performance timings for different data sets are listed in Table 1. As shown, the number of sampled photons (col. *NumP*) increases as the resolution of volume data set increases. The precomputation time (col. *Prep*) is related to both the number of photons and the density field occupation inside the volume that determines the effective voxels during ray/photon marching. Usually, more photons demand more precomputation time. Note, the precomputation time listed in Table 1 includes the timings of both SS and MS. The PVRT method aims at handling high-resolution volumes, and even with a $504 \times 504 \times 228$ volume, the precomputation time is just around 6.5 minutes. Also note that the current test GPU is designed for laptops and, thus, has much weaker performance when compared with the PC

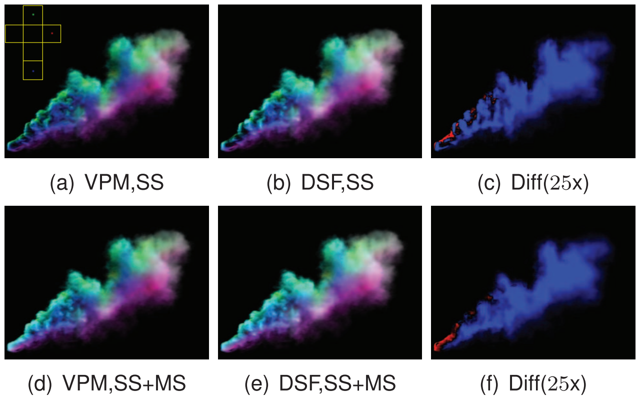


Fig. 10. Comparison between the ground truth generated by VPM and our results using DSF.

version. With state-of-the-art GPUs, our precomputation performance is expected to be further improved by at least a factor of 3. For the rendering stage, the step of generating the radiance volume texture is very efficient, and for all data sets, we tested it only takes around 1 ms (col. *RV*). Relying on the radiance volume texture, the following ray marching can be performed in real time (col. *Render*).

To validate the accuracy of our method, Fig. 10 gives a comparison between our results using DSF and the references generated by VPM, which is treated as ground truth. The first row shows the results with only direct illumination from SS, and the second row shows the full GI with both SS and MS. Visually, we can clearly distinguish the difference between with and without volumetric GI effects. The difference images show that our results are very close to the references. To compare the accuracy of different basis functions, Fig. 11 shows the comparison between the references and the results generated by the PVRT method using different bases. In the first row, we utilized a synthetic environment light with three spherical lights in different colors, and in the second row, we adopted a real EM. It is easy to notice that DSF reaches the

TABLE 2
Statistics of Different Number of CPCA Clusters

No. Cluster	L^2	CPCA Time (s)
32	2.2%	12
64	1.3%	19
128	0.9%	31
256	0.8%	52

best visual quality compared with other bases, especially SH. The L^2 errors also numerically demonstrate this conclusion. To verify the quality loss of using CPCA compression, we compared the rendering of a smoke data set, wherein density distribution is random, using different numbers of CPCA clusters. The statistical data are shown in Table 2, which shows that using 128 clusters only introduces 0.1 percent more L^2 error compared with using 256 clusters, while the CPCA compression using 256 clusters spends almost double the time compared with using 128 clusters. Hence, using 128 clusters for CPCA compression is optimal to achieve a good balance between performance and quality. To test the impact of different input volume resolutions on the rendering quality, we spatially downsampled the machine room data set to half the resolution and rendered it using directional light. Since the resolution decreases, the total number of photons and the precomputation time is reduced to be 0.75 mil. and 21 sec., respectively, compared with the original data set (Table 1). Yet, as shown in Fig. 12, the rendering quality also declines and the L^2 error from the result using high-resolution volume is 4.3 percent. As mentioned in Section 8.3, when using DSF, each boundary voxel shoots $N_P = 10$ photons for each basis light. To verify the quality, in Fig. 13, we compared the rendering results using $N_P = 1$ and $N_P = 10$ with the reference. This resolution of this Pillars volume is 100^3, and hence, the total number of boundary voxels is 6×100^2. Since the VPM strategy only contributes to the indirect illumination of the final rendering, visually it is not

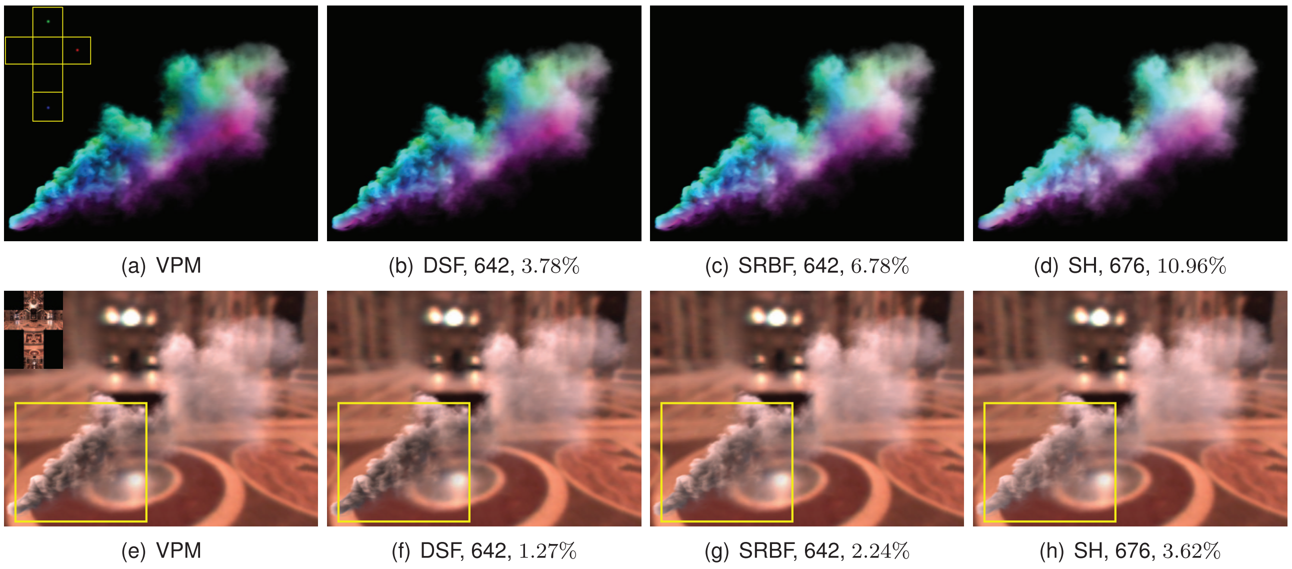
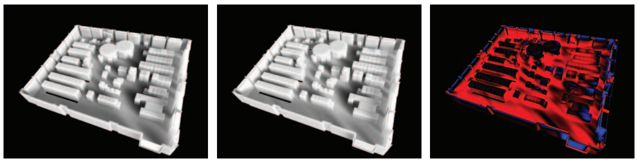
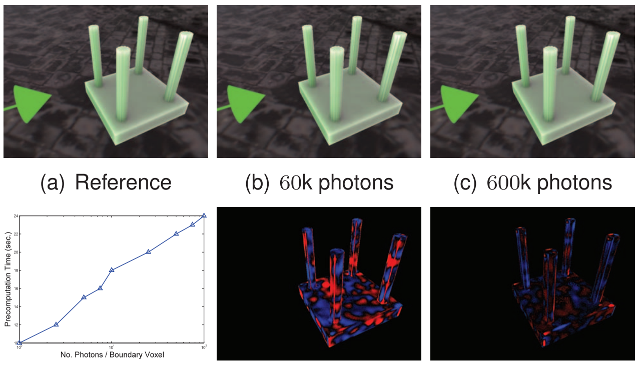


Fig. 11. Comparison between the references generated by VPM and the results using different bases.



(a) $209 \times 173 \times 30$ (b) $417 \times 345 \times 60$ (c) Diff(10x)

Fig. 12. Comparison of volume GI rendering with different volume resolution.



(a) Reference (b) 60k photons (c) 600k photons

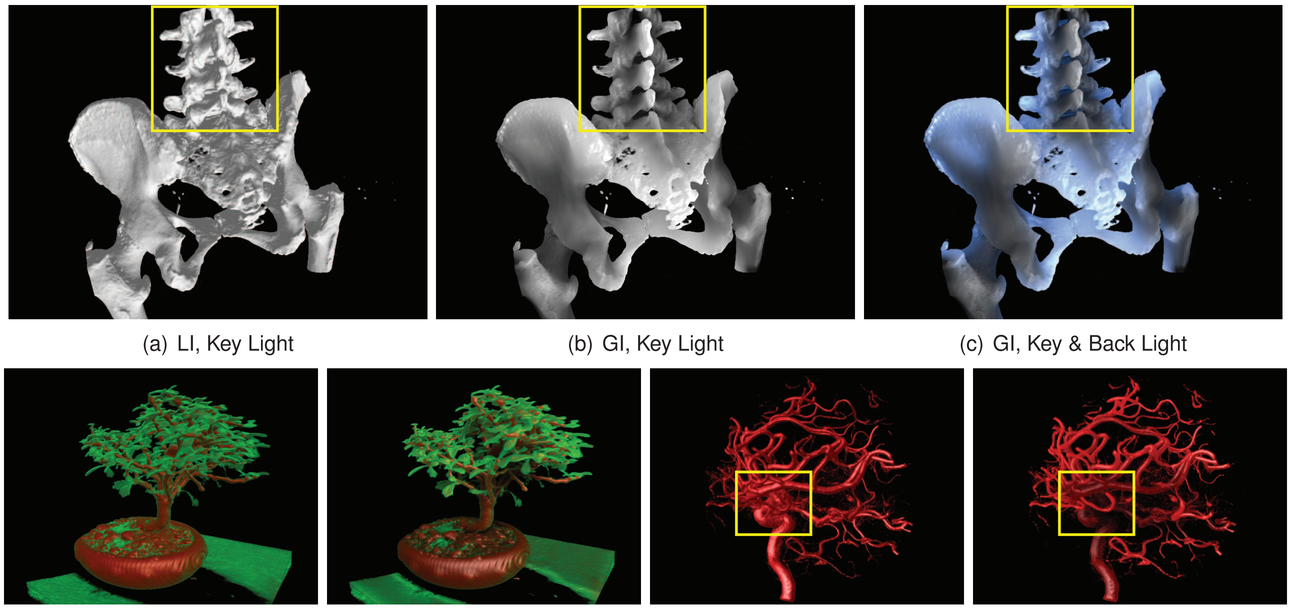
(d) Timing (e) Diff(25x),1.4% (f) Diff(25x),0.6%

Fig. 13. Comparison of using different number of photons.

easy to distinguish the differences between using different numbers of photons. However, the difference images ($10\times$) and the L^2 errors clearly show the accuracy difference. Since when using $N_P = 10$ photons, the L^2 error is below 1 percent, and hence, it provides sufficient visual quality at good performance. Further, Fig. 13d illustrates the trend of the precomputation time as a function of number of photons for each boundary voxel, which yields almost a linear curve.

We applied the PVRT method for various visualization scenarios. Fig. 14 shows three study cases where our renderer is used for volume visualization. The first row shows the rendering of a bone volume data set using conventional volume ray casting with local shading and our method (using DSF) with volume GI effects. In the first image, it is obvious that certain features, such as the spatial structure of the spine, cannot be clearly perceived under direct illumination. However, these features are correctly presented using our method with volume GI. In addition, by using both key light and back light, MS effects from the back light offer better perception of the structure within the spine. The second and third data sets are a CT scan of a bonsai tree and an aneurysm, respectively. It is noticeable that our result looks more 3D and provides a much better visual perception of the depth order of complex leaves or vessel structures. To demonstrate whether the high-frequency lighting is meaningful for visualization, in Fig. 15, we tested the part data set under environment light that is approximated using sixth SH and DSF with 642 bases, respectively. From this test, we can easily see that the rendering using low-frequency lighting smoothes out a great deal of the spatial features over the volume (shown in the yellow square), while the volume rendering results with high-frequency lighting preserves the important volume features and, hence, provides a much better perception. Therefore, along with the real-time performance, the PVRT method offers high potential for improving the scientific and medical visualization quality.

Limitations. In the precomputation stage of the PVRT method, the physical properties of each voxel, such as density field value, phase function, and so on, can impact the transport of each photon and determine the final radiant exitance. Therefore, although we can render the volumetric effects in real time, the physical properties of each voxel cannot be changed during rendering because the radiance transportation is precomputed. Also, the indirect light



(a) LI, Key Light (b) GI, Key Light (c) GI, Key & Back Light

(d) Bonsai, LI (e) Bonsai, GI (f) Aneurism, LI (g) Aneurism, GI

Fig. 14. Comparison of volume rendering results between local illumination and GI using our method.

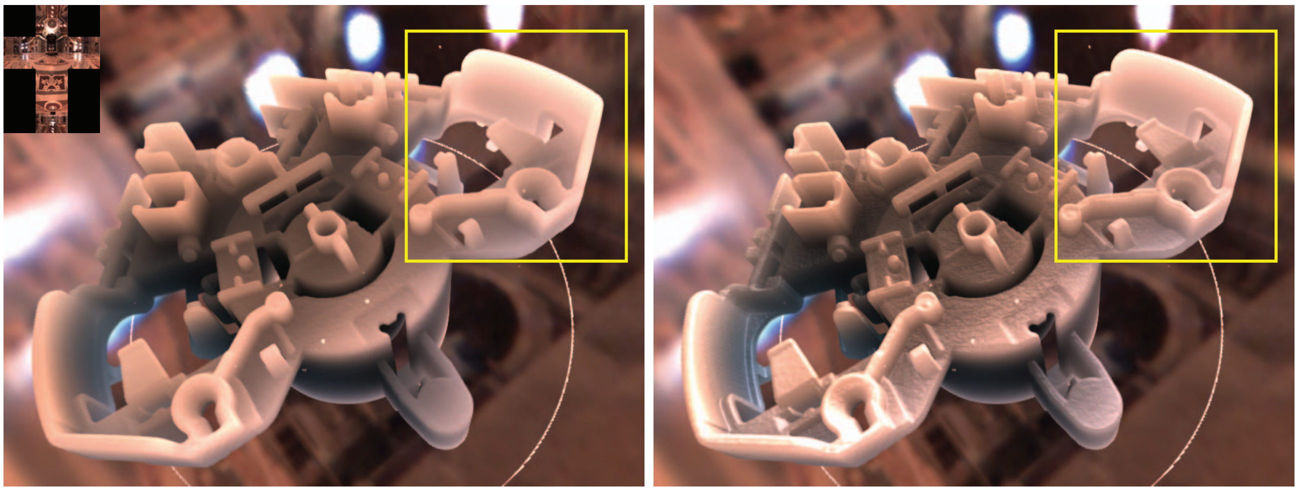


Fig. 15. Comparison of volume GI rendering for visualization between using low-frequency SH (left column) and using high-frequency DSF (right column).

transportation does not change according to the change of opacity mapping. However, since photon tracing is pursued for each basis light, the basis coefficients for current lighting can be freely changed. Hence, we can support dynamic lighting, view and color transfer function change, in real time (as shown in the accompanying video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2013.17).

10 Conclusion and Future Works

In this paper, we presented a PVRT method that is feasible to incorporate GI at real-time rates for rendering volume data. The added realism and clarity can greatly benefit practical visualization tasks. To simulate time-consuming MS, the PVRT method incorporates the VPM technique into the precomputation stage and introduces a novel photon sampling strategy and photon splatting routine to compute the radiant exitance for each voxel. Our PVRT framework currently supports both distant spherical lighting and local directional lighting. To approximate all-frequency lighting, we further present the DSFs, which have a local interpolation property and are accurate and efficient for reconstruction. The PVRT method is practical for rendering high-quality volume GI effects for various high-resolution volume data sets. Our experimental results demonstrate its merits and show that it can benefit both interactive graphics and scientific visualization applications.

For future work, we aim at removing the current constraint of fixing the volume physical properties in precomputation so that the user can interactively edit the volume data set. Further, since the PVRT method inherits the routine of VPM, it inherently could support the rendering of volume caustics. As a next step, we would thus like to investigate high-quality volume caustics rendering using the PVRT method.

Acknowledgments

This research was sponsored in part by the US National Science Foundation through grants OCI 0905008, OCI 0850566, and OCI 0749227, and also by the US Department of Energy through grants DE-FC02-06ER25777 and DE-FC02-12ER26072, program manager Lucy Nowell. The authors would like to thank volvis.org for the volume data and Paul Debevec for the environment maps. Finally, they sincerely thank the reviewers and Bob Miller for helping in improving the manuscript.

References

[1] J.T. Kajiya and B.P. Von Herzen, "Ray Tracing Volume Densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 165-174, 1984.

[2] H.W. Jensen and P.H. Christensen, "Efficient Simulation of Light Transport in Scences with Participating Media Using Photon Maps," *Proc. ACM SIGGRAPH*, pp. 311-320, 1998.

[3] P.-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, pp. 527-536, 2002.

[4] Y.-T. Tsai and Z.-C. Shih, "All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation," *ACM Trans. Graphics*, vol. 25, pp. 967-976, 2006.

[5] E. Cerezo, F. Perez-Cazorla, X. Pueyo, F. Seron, and F. Sillion, "A Survey on Participating Media Rendering Techniques," *Visual Computer*, vol. 21, pp. 303-328, 2005.

[6] M. Hadwiger, P. Ljung, C.R. Salama, and T. Ropinski, "Advanced Illumination Techniques for GPU Volume Raycasting," *Proc. ACM SIGGRAPH ASIA*, pp. 1:1-1:166, 2008.

[7] J. Kautz, P.-P. Sloan, and J. Lehtinen, "Precomputed Radiance Transfer: Theory and Practice," *Proc. ACM SIGGRAPH*, 2005.

[8] R. Ramamoorthi, "Precomputation-Based Rendering," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 4, pp. 281-369, 2009.

[9] S. Chandrasekhar, *Radiative Transfer*. Dover Publications, 1960.

[10] M. Levoy, "Efficient Ray Tracing of Volume Data," *ACM Trans. Graphics*, vol. 9, pp. 245-261, 1990.

[11] E.P. Lafortune and Y.D. Willems, "Rendering Participating Media with Bidirectional Path Tracing," *Proc. Eurographics Rendering Workshop (EGWR)*, pp. 91-100, 1996.

[12] H.E. Rushmeier and K.E. Torrance, "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium," *SIGGRAPH Computer Graphics*, vol. 21, pp. 293-302, 1987.

[13] J. Stam, "Multiple Scattering as a Diffusion Process," *Proc. Eurographics Rendering Workshop (EGWR)*, pp. 41-50, 1995.

[14] A. Boudet, P. Pitot, D. Pratmarty, and M. Paulin, "Photon Splatting for Participating Media," *Proc. Third Int'l Conf. Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pp. 197-204, 2005.

[15] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-Time Kd-Tree Construction on Graphics Hardware," *ACM Trans. Graphics,* vol. 27, pp. 126:1-126:11, 2008.
[16] R. Ng, R. Ramamoorthi, and P. Hanrahan, "All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation," *ACM Trans. Graphics,* vol. 22, pp. 376-381, 2003.
[17] W.-C. Ma, C.-T. Hsiao, K.-Y. Lee, Y.-Y. Chuang, and B.-Y. Chen, "Real-Time Triple Product Relighting Using Spherical Local-Frame Parameterization," *Visual Computer,* vol. 22, no. 9, pp. 682-692, Sept. 2006.
[18] 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 (I3D),* pp. 7-14, 2006.
[19] 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, July 2005.
[20] C. Wyman, S. Parker, P. Shirley, and C. Hansen, "Interactive Display of Isosurfaces with Global Illumination," *IEEE Trans. Visualization Computer Graphics,* vol. 12, no. 2, pp. 186-196, Mar. 2006.
[21] K.M. Beason, J. Grant, D.C. Banks, B. Futch, and M.Y. Hussaini, "Pre-Computed Illumination for Isosurfaces," *Proc. Conf. Visualization and Data Analysis,* pp. 98-108, 2006.
[22] T. Ritschel, "Fast GPU-Based Visibility Computation for Natural Illumination of Volume Data Sets," *Proc. Eurographics Conf.,* pp. 17-20, 2007.
[23] J.T. Moon, B. Walter, and S. Marschner, "Efficient Multiple Scattering in Hair Using Spherical Harmonics," *ACM Trans. Graphics,* vol. 27, pp. 31:1-31:7, 2008.
[24] A. Kaplanyan and C. Dachsbacher, "Cascaded Light Propagation Volumes for Real-Time Indirect Illumination," *Proc. Symp. Interactive 3D Graphics and Games (I3D),* pp. 99-107, 2010.
[25] T. Engelhardt, J. Novak, and C. Dachsbacher, "Instant Multiple Scattering for Interactive Rendering of Heterogeneous Participating Media," technical report, Karlsruhe Inst. of Technology, Dec. 2010.
[26] K. Zhou, Z. Ren, S. Lin, H. Bao, B. Guo, and H.-Y. Shum, "Real-Time Smoke Rendering Using Compensated Ray Marching," *ACM Trans. Graphics,* vol. 27, pp. 36:1-36:12, 2008.
[27] Z. Ren, K. Zhou, S. Lin, and B. Guo, "Gradient-Based Interpolation and Sampling for Real-Time Rendering of Inhomogeneous, Single-Scattering Media," *Computer Graphics Forum,* vol. 27, pp. 1945-1953, 2008.
[28] F. Hernell, P. Ljung, and A. Ynnerman, "Efficient Ambient and Emissive Tissue Illumination Using Local Occlusion in Multi-resolution Volume Rendering," *Proc. IEEE/EG Conf. Vol. Graphics,* 2007.
[29] C.R. Salama, "GPU-Based Monte-Carlo Volume Raycasting," *Proc. Pacific Conf. Computer Graphics and Applications,* pp. 411-414, 2007.
[30] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson, "A Model for Volume Lighting and Modeling," *IEEE Trans. Visualization Computer Graphics,* vol. 9, no. 2, pp. 150-162, Apr. 2003.
[31] T. Ropinski, C. Döring, and C. Rezk-Salama, "Interactive Volumetric Lighting Simulating Scattering and Shadowing," *Proc. IEEE Pacific Visualization Symp. (PacificVis),* pp. 169-176, 2010.
[32] J. Kronander, D. Jönsson, J. Low, P. Ljung, A. Ynnerman, and J. Unger, "Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering," *IEEE Trans. Visualization Computer Graphics,* vol. 18, no. 3, pp. 447-462, Mar. 2012.
[33] M. Müller, D. Charypar, and M. Gross, "Particle-Based Fluid Simulation for Interactive Applications," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '03),* pp. 154-159, 2003.
[34] 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.
[35] J. Kruger and R. Westermann, "Acceleration Techniques for GPU-Based Volume Rendering," *Proc. IEEE Visualization (VIS '03),* pp. 287-292, 2003.
[36] NVIDIA "Cuda Programming Guide,"http://developer.nvidia.com/cuda-downloads, 2011.

Yubo Zhang received the BS degree in computational mathematics from Zhejiang University in 2007 and the MPhil degree in applied mathematics from Hong Kong Baptist University in 2009. He is currently working toward the PhD degree in computer science at the University of California, Davis. His current research interests include scientific visualization, volume rendering, and physically based animation. He is a student member of the IEEE.

Zhao Dong received the BS and MS degrees from Zhejiang University, China, in 2001 and 2005, respectively, and the PhD degree in computer graphics from Max-Planck-Institut Informatik, Germany, in 2011. He is a postdoc at Program of Computer Graphics of Cornell University. His current research interests include real-time global illumination rendering, physically-based material modeling and rendering, and volume graphics. He is a member of the IEEE.

Kwan-Liu Ma received the PhD degree in computer science from the University of Utah in 1993. He is a professor of computer science and the chair of the Graduate Group in Computer Science at the University of California, Davis. He leads the VIDi Research Group and directs the UC Davis Center for Visualization. He received the PECASE Award in 2000. His research interests include visualization, high-performance computing, computer graphics, and user-interface design. He was a paper chair of the IEEE Visualization Conference in 2008 and 2009, and an associate editor of IEEE TVCG (2007-2011). He is a founding member of the IEEE Pacific Visualization Symposium and the IEEE Symposium on Large Data Analysis and Visualization. He currently serves on the editorial boards of the *IEEE CG&A*, the *Journal of Computational Science and Discoveries*, and the *Journal of Visualization*. He is a fellow of the IEEE.