Gradient-based Interpolation and Sampling for Real-Time Rendering of Inhomogeneous, Single-Scattering Media

Zhong Ren Kun Zhou Stephen Lin Baining Guo

Microsoft Research Asia

Abstract

We present a real-time rendering algorithm for inhomogeneous, single scattering media, where all-frequency shading effects such as glows, light shafts, and volumetric shadows can all be captured. The algorithm first computes source radiance at a small number of sample points in the medium, then interpolates these values at other points in the volume using a gradient-based scheme that is efficiently applied by sample splatting. The sample points are dynamically determined based on a recursive sample splitting procedure that adapts the number and locations of sample points for accurate and efficient reproduction of shading variations in the medium. The entire pipeline can be easily implemented on the GPU to achieve real-time performance for dynamic lighting and scenes. Rendering results of our method are shown to be comparable to those from ray tracing.

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

Keywords: participating media, single scattering, gradient-based interpolation

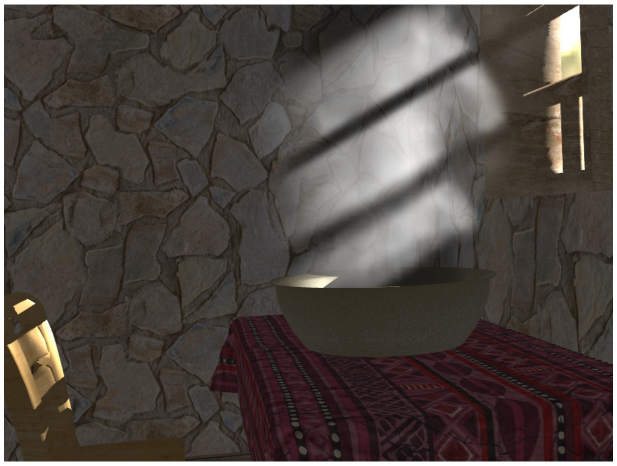


Figure 1: *Real-time rendering of single scattering media that captures fast shading variations inside the volume and generates complex volumetric shadows. The scene is rendered at* 23.4 *fps, with dynamic lighting, medium and scene object.*

1 Introduction

The transport of light within an inhomogeneous participating medium produces a number of volumetric shading effects essential to realistic rendering. Effects such as glows around a light source and shafts of directional light reveal the density variations of the medium and the structure of the illumination. Mutually cast shadows between scene objects and the medium provide further cues for perceiving the organization and properties of the scene.

These shading effects can be accurately reconstructed by a full Monte Carlo simulation, but at an enormous expense in computation. Kajiya and Herzen [1984] reduce computation by separating the rendering procedure into two steps. The first step computes the source radiance at each voxel center in the volume, and the second step then marches along view rays to gather the source radiance among these sample points. Because of the dense sampling of source radiance, the first step requires substantial computation and offline processing.

To reduce sampling of source radiance, subsequent techniques have assumed shading to be smooth within the medium, such that the source radiance throughout the volume can be well approximated by interpolation from a small number of samples. Specifically, they first sample radiance at only a small number of points according to the density distribution of the medium. Then these sampled radiances are smoothly interpolated by radial basis functions (RBFs) to determine the source radiance at other points in the volume [Dobashi et al. 2000; Zhou et al. 2007b]. This optimization works well with distant, low-frequency lighting for which the assumption of shading smoothness generally holds. However, sharp shading variations often exist with local, directional, or high-frequency illumination. This variability in shading arises not only from profiles of light paths (e.g., light shafts from spot lights), but also from volumetric shadows cast by scene objects. In such cases, these RBF-based interpolation methods cannot accurately compute shading in the medium, and may produce severe rendering artifacts.

In this paper, we propose a real-time rendering algorithm for inhomogeneous, single scattering media that accounts for sharp variations of shading in the volume. In contrast to previous works which determine sample points based on density distributions, our method dynamically distributes sample points in the medium in a manner that allows for more accurate reconstruction of source radiance by interpolation and reduces shading errors in the rendered result. Areas in the medium whose reconstructed source radiance results in significant shading errors are assigned more samples to improve rendering accuracy, while other areas are lightly sampled to save computation. At each of the sample points, we numerically compute the source radiance and its gradient, which is used to heighten the accuracy of source radiance interpolation at other points in the volume. The computation of source radiances is followed by a ray march to composite the final radiance along view rays.

This approach yields the first real-time rendering algorithm that captures all-frequency shading effects in scattering media, including glows in inhomogeneous media, volumetric shadows, and shafts of light. Furthermore, no precomputation of light transport is needed, and dynamic changes in lighting, media and scene configurations are supported. As in many real-time volumetric rendering algorithms, we assume the medium to be single scattering and to have a volume representation. With this technique, results comparable to ray tracing can be achieved for challenging illumination and scene conditions, as illustrated in Fig. 1.

2 Related Work

Numerous methods have been proposed for rendering of scattering media [Cerezo et al. 2005]. Here, we review representative works that are most closely related to ours.

Offline Algorithms Starting from [Kajiya and Herzen 1984], researchers have been seeking numerical solutions to the radiative transfer equation [Chandrasekhar 1960] by ray tracing [Kajiya and Herzen 1984; Levoy 1990; Lafortune and Willems 1996] or by finite element methods [Rushmeier and Torrance 1987]. Photorealistic images can be produced with such methods, but at the cost of hours of simulation. Although several acceleration techniques have been proposed [Sakas 1990; Stam 1995; Max 1994; Jensen and Christensen 1998], their performance nevertheless remains far from real-time.

Real-time Algorithms Most real-time rendering algorithms for scattering media exploit the power of graphics hardware. Ebert and Parent [1990] combine volume rendering and the A-buffer technique to render animations of both scene objects and gaseous phenomena. A reduced resolution shadow table is constructed on the fly for volume rendering, but the construction of this table remains too expensive for real-time rendering. Real-time performance with volumetric shading can be achieved with precomputation of radiance data, as done in [Harris and Lastra 2001; Riley et al. 2004] for sky and clouds, in [Dobashi et al. 2002; Hegeman et al. 2005] for shafts of light in foggy scenes, and in [Sloan et al. 2002] for scattering media under distant, low-frequency lighting. However, with precomputed radiance quantities, interactive changes cannot be made to media properties and scene configuration, as well as lighting in some cases.

Kniss et al. [2003] propose a hardware-accelerated algorithm based on half-angle slicing that achieves interactive frame rates for general inhomogeneous media. But the need for explicit shading of each voxel makes it unsuitable for real-time applications, especially when dealing with multiple light sources. Schpok et al. [2003] model clouds by combining implicit ellipsoids and octave noise. Shading is computed at vertices that are uniformly distributed on slice planes generated at runtime. The method focuses on rendering of clouds under directional lighting; sharper shading variations would require more vertices than can be handled at real-time rates. Zhou et al. [2007b] render inhomogeneous smoke animations with both single and multiple scattering by applying a low-frequency shading model and assuming low-frequency environment lighting.

Analytic models that are efficient to compute have been derived for single scattering, homogeneous media. An early model was proposed by Blinn [1982] for homogeneous media illuminated by an infinitely distant light source. Other analytic models have been presented for a point light source immersed in a homogeneous medium [Max 1986; Narasimhan and Nayar 2003; Biri et al. 2006; Sun et al. 2005]. An approximate, analytic expression of radiance was recently derived for smooth, inhomogeneous media modeled as a sum of Gaussians [Zhou et al. 2007a]. This method, however, cannot handle media with fine-scale density variations and cannot generate sharp lighting effects such as shafts of light.

Gradient-based Interpolation and Sampling Gradient-based interpolation and sampling has been used in several offline rendering applications. Ward and Heckbert [1992] compute gradients for interpolation of global illumination on object surfaces. Ramamoorthi et al. [2007] give a first order analysis of lighting, shading and shadows, and show how visibility gradients can be efficiently evaluated by sampling along discontinuities. Gradient-based interpolation and sampling methods for soft shadows on surfaces are also discussed. Jarosz et al. [2007] compute gradients of the radiative transport equation, under the assumption of constant visibility, to estimate the local variation of scattered radiance and to improve the accuracy of interpolation.

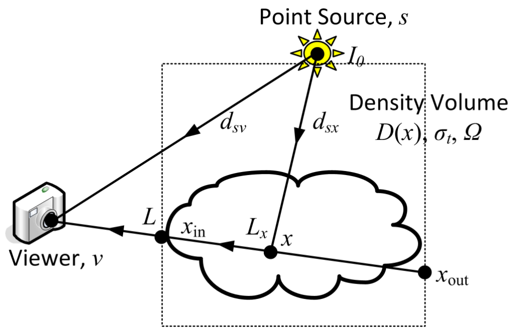


Figure 2: *Light transport in a single scattering medium from a point light source to the viewer.*

In a real-time application, Gautron et al. [2005] perform radiance splatting on the GPU for gradient-based interpolation of indirect illumination on surfaces. In our work, we also employ splatting for gradient-based interpolation, but formulate it instead for shading in participating media.

3 Overview

In this section, we describe basic concepts of our algorithm, namely the lighting model and density field representation, and provide a brief overview of the rendering algorithm.

3.1 Lighting Model

Our work addresses light transport within an inhomogeneous medium represented by a density field D defined in a volume $\mathbb{V}$. The volume is considered to contain a single medium, whose parameters include the *extinction cross section* σ_t, the *scattering cross section* σ_s and the *scattering albedo* $\Omega = \sigma_s/\sigma_t$. We assume the medium to be single scattering, such that radiance reaching the viewer has undergone at most one scattering interaction in the medium, and that the scattering is isotropic, i.e., uniform in all directions.

For simplicity, let us consider here the scenario of a point light source as shown in Fig. 2. Lighting for other source types can be straightforwardly derived from the point source case.

Source radiance refers to light that is directed towards the viewer from a point x in the medium. For a point source s and an isotropic, single scattering medium, source radiance is computed as

$$L_x = \frac{I_0}{4\pi d_{sx}^2}\tau_{sx}, \qquad (1)$$

where I_0 is the point source intensity, d_{ab} denotes the distance from a to b, and the *transmittance* τ_{ab} models the reduction of radiance due to extinction from point a to b, computed as $\exp(-\sigma_t \int_a^b D(x)dx)$.

In terms of source radiance, the radiance L seen at the viewer can be computed as

$$L = \frac{I_0}{d_{sv}^2}\tau_{sv} + \frac{\Omega}{4\pi}\int_{x_{out}}^{x_{in}} D(x)L_x\tau_{xv}dx, \qquad (2)$$

where the first term describes direct transmission of radiance from the source to the viewer, and the second term accounts for single scattered radiance in the medium, which is the cause of glows around a light source. We note that for a point light source, the first term in Eq. (2) contributes to at most a single point on the screen. In the second term, source radiances are modulated by media density and transmittance before being integrated along view rays.

An extension of Eq. (2) to volumes containing scene objects will later be presented in Section 6.

3.2 Density Field Representation

To compactly represent the density field D, we employ the Gaussian model described in [Zhou et al. 2007b]. Density is represented by a weighted sum of Gaussians and a hashed residual field F:

$$D(x) = \tilde{D}(x) + F(x) = \sum_{j=1}^{n} w_j \exp(-\|x - c_j\|^2/r_j^2) + F(x), \tag{3}$$

where each Gaussian is defined by its center c_j, radius r_j and weight w_j. A media animation is then modeled as a sequence of Gaussians and residual fields, which are computed in a preprocess as done in [Zhou et al. 2007b].

We note that preprocessing is used here only for representation of the density field, and it does not prevent run-time changes to media properties, lighting, or scene configuration. This representation was chosen for its efficient modeling of fine density field details, but our rendering algorithm can accommodate any representation that can be rapidly reconstructed at runtime, e.g., the Gaussian+noise representation in [Zhou et al. 2007a] or the advected RBF representation in [Pighin et al. 2004]. With these alternative representations, no preprocessing would be needed.

3.3 Algorithm Overview

For each frame in an animated sequence, our algorithm first generates a set of sample points $\{x_j\}$ at which to evaluate source radiance. This set is selected using a dynamic sampling strategy that aims to minimize shading error in the rendered image by accurately reconstructing the distribution of source radiance. Details of this sampling procedure will be described in Section 5.

Then at each x_j, a volume ray tracer numerically evaluates the source radiance L_{x_j} and its gradient ∇L_{x_j}. The source radiance L_x at other points x in the volume are reconstructed using a gradient-based interpolation scheme, which we present in Section 4. This interpolation is shown to yield significant improvements in quality even without the use of dynamic sampling.

Finally, a ray march is performed for discrete computation of the integral in Eq. (2). Implementation details of the algorithm will be given in Section 6.

4 Gradient Based Interpolation

In this section, we present a real-time interpolation algorithm that reconstructs the source radiance throughout the volume from a small set of samples. For heightened accuracy in interpolation, the source radiance at an arbitrary point is evaluated using both the radiance values and radiance gradients of the sample points. We utilize the GPU to expedite this computation by calculating sampled radiance quantities in multiple threads and by splatting the samples into the volume in a manner analogous to [Gautron et al. 2005].

Radiance Samples For gradient-based interpolation, we define a sample j by a point x_j in the media volume, the source radiance L_{x_j} at that point, and the radiance gradient ∇L_{x_j}. In addition, we associate with each sample a *valid radius* R_j that describes the range from x_j within which a sample j may be used for interpolation. The sphere determined by point x_j and valid radius R_j is referred to as the *valid sphere* of sample j.

In our algorithm, the set of sample points is determined using the dynamic sampling method in Section 5. However, to allow comparison of our gradient-based interpolation to RBF-based interpolation, we will instead in this section construct the sample set from the Gaussian centers c_j of the density representation in Eq. (3), such that $x_j = c_j$. The valid radius of each valid sphere is set to the culling radius of the corresponding Gaussian: $R_j = 3r_j$.

Evaluation of Source Radiance and Gradient At each sample point x, we use Eq. (1) to evaluate its source radiance. In computing Eq. (1), we use volume tracing for discrete integration along the ray from x to the light source s at intervals of Δ_1:

$$L_x = \frac{I_0}{4\pi d_{sv}^2} \exp\left(-\sigma_t \Delta_1 \sum_{u \in \mathbb{U}} D(u)\right),$$

$$\mathbb{U} = \{u_k : u_k = x + vk\Delta_1,\ k = 0, 1, \ldots \lfloor \|s - x\|/\Delta_1 \rfloor,\ u_k \in \mathbb{V}\},$$

where $v = (s - x)/\|s - x\|$ represents the ray direction. At each volume tracing step, the density is obtained from the density field and accumulated into the running sum until u exits the volume $\mathbb{V}$. The transmittance is then evaluated and multiplied by $I_0/(4\pi d_{sv}^2)$ to yield L_x.

The gradient is determined numerically from the source radiance values at six points surrounding x along the three axis directions X, Y, Z:

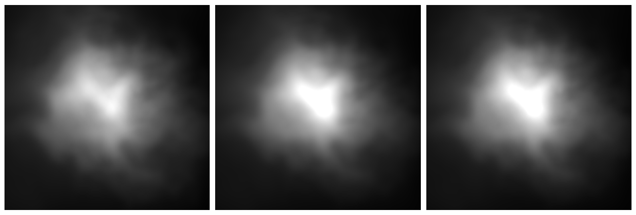
$$\nabla L_x = \left(\frac{L_{x+\Delta_2 X} - L_{x-\Delta_2 X}}{2\Delta_2}, \frac{L_{x+\Delta_2 Y} - L_{x-\Delta_2 Y}}{2\Delta_2}, \frac{L_{x+\Delta_2 Z} - L_{x-\Delta_2 Z}}{2\Delta_2} \right).$$

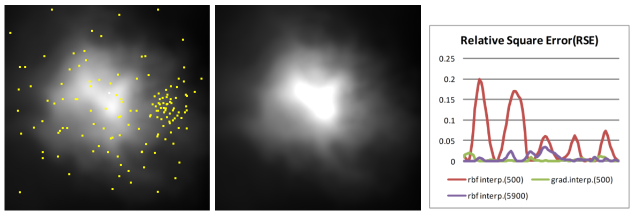
We note that the source radiance at the various sample points are computed in parallel on the GPU. Also, the precision of this numerical evaluation is controlled by the user defined intervals Δ_1 and Δ_2.

Gradient-based Interpolation by Sample Splatting With the computed values of L_{x_j} and ∇L_{x_j} at each sample point x_j, the radiance L_x at an arbitrary point x is computed as a weighted average of the first-order Taylor approximations evaluated from each contributing sample to x. More precisely,

$$\begin{aligned} &L_x = \textstyle\sum_S W_j(x) \left(L_{x_j} + (x - x_j) \cdot \nabla L_{x_j} \right) / \textstyle\sum_S W_j(x), \\ &S = \{j : \|x - x_j\| < R_j\},\ \ W_j(x) = R_j/\|x - x_j\|. \end{aligned} \tag{4}$$

In interpolating the source radiance of a point x, rather than directly retrieve samples whose valid sphere covers x, we utilize the GPU to splat the samples into the volume. First, the valid sphere of each sample is intersected with each $X - Y$ slice of the volume, with $+Z$ aligned to the viewing axis. The bounding quads of the intersection circles are found and grouped by slices. Then, for each slice, these bounding quads are rendered with alpha blending enabled. For each pixel, the weighted approximate radiance $W_j(x) \left(L_{x_j} + (x_j - x) \cdot \nabla L_{x_j} \right)$ and the weighting function $W_j(x)$ are evaluated and accumulated. Rendering all bounding quads for a slice yields the numerator and denominator of Eq. (4), from which we compute L_x. The bounding quad of all intersection circles in the slice is then rendered, with L_x evaluated at each pixel. The result is rendered into a 3D volume texture, using the technique described in [NVIDIA 2007].


(a) RBF Interp. with 500 samples (b) Reference (c) Gradient Interp. with 500 samples



(d) Sample points for Gradient Interp. (e) Dense RBF Interp. with 5900 samples (f) Relative Error

Figure 3: *Comparison of gradient-based interpolation, RBF-based interpolation, and ray tracing. The density values range from 0.0 to 1.0, with $\sigma_t = 1.2$ and $\Omega = 0.6$. The reference image is obtained by ray tracing at each voxel of the $128 \times 128 \times 128$ volume, followed by standard ray marching.*

Result In Fig. 3, we compare the result of gradient-based interpolation, the RBF-based interpolation described in [Zhou et al. 2007b] and a reference ray tracer. The RBF-based interpolation in (a) (relative error of 14.7%) does not adequately capture the fast variation of source radiance near the point source because of its coarse interpolation of the sparse samples. Set to a comparable performance level, the gradient-based interpolation in (c) (relative error of 4.3%) more faithfully approximates the reference solution in (b). Increasing the number of RBFs in (a) can lead to comparable accuracy, as shown in (e) (relative error of 5.4%), but results in slow performance.

5 Dynamic Sampling

With gradient-based interpolation, source radiance throughout the medium can be better reconstructed from a sparse set of samples. However, sharp shading variations tend not to be well modeled without denser sampling. We illustrate this problem with the simple case of a light shaft piercing a medium of uniform density, shown in Fig. 4. With gradient-based interpolation and radiance samples taken at only Gaussian centers, the shape of the shaft is seen to be indistinct in Fig. 4(a). Moreover, in animation sequences, jittering of shading boundaries often appears due to location shifts of sparse samples. For accurate and efficient reconstruction, our method dynamically places additional samples in areas with greater shading error according to the current sampling configuration and the gradient-based interpolation.

The dynamic sampling algorithm consists of two main components. One is a metric for local shading error within the valid sphere of a sample. We formulate this metric to account for discrepancies in interpolated source radiance and the resulting errors in viewed shading. In addition, this measure is designed for rapid evaluation. The second component is a recursive procedure that splits samples into multiple parts that more finely sample the area within a valid sphere if the original sample has a large local shading error. With this adaptive resampling scheme, our method can accurately and efficiently generate high-frequency lighting effects as shown in Fig. 4(c).

(a) Original sampling with 500 samples (b) Reference (c) Dynamic sampling with 2732 samples

Figure 4: *Dynamic sampling for accurate and efficient generation of fast shading variations in a medium. A uniform-density volume is illuminated by a spot light, creating a shaft of light. Sufficient sampling is needed by gradient-based interpolation to adequately capture the shape of the shaft. Sample distributions are shown in upper-right insets, and the reference image is computed by ray tracing a $128 \times 128 \times 128$ volume.*

Local Shading Error The shading error of a given media point due to an approximation $\widetilde{L}_x$ of its source radiance can be derived from Eq. (2) as

$$\delta L_x = \frac{\Omega}{4\pi} \int_{x_{out}}^{x_{in}} D(x)(\widetilde{L}_x - L_x)\tau_{xv} dx.$$

For the local shading error within a valid sphere, we seek an efficiently computable metric that represents the total error over all the points in the sphere. We measure the local shading error of a sample j as

$$E_j = R_j^3 \sum_{i=1}^{n} \frac{|\widetilde{L}_{x_{ij}} - L_{x_{ij}}|}{n} D(x_{ij})\tau_{x_j v} \tag{5}$$

where $\{x_{ij}\}$ is a set of n sampled points within the valid sphere, taken in our implementation as $\{x_j \pm R_j X, x_j \pm R_j Y, x_j \pm R_j Z\}$. The factor $|\widetilde{L}_{x_{ij}} - L_{x_{ij}}|/n$ represents the average approximation error of source radiance among the sampled points. For computational efficiency, the transmittance from each point to the viewer is approximated as that from the sphere center, $\tau_{x_j v}$. In shading, rays are marched through the volume of the sphere, which is proportional to R_j^3. Since this metric measures local shading error with respect to a given sample, we determine $\widetilde{L}_{x_{ij}}$ by computing Eq. (4) using only that sample point:

$$\widetilde{L}_{x_{ij}} \approx L_{x_j} + (x_{ij} - x_j)\nabla L_{x_j}.$$

Volume tracing is used to sample source radiance values at x_j and the sampled points, and density values are determined by sampling the density field.

Recursive Sample Splitting Starting with a sample set $\mathbb{Q}^0 = \{c_j\}$ that contains only the Gaussian centers, we compute the local shading error E_j according to Eq. (5) for each valid sphere, and compare it to a given threshold, ϵ. Within each valid sphere for which $E_j > \epsilon$, additional samples are added for more accurate modeling of the source radiance distribution in the medium.

The set of added samples $\mathbb{Q}_j^1 = \{q : q \in \mathbb{G}_1 \wedge \|q - x_j\| < R_{x_j}\}$ is composed of vertices of a grid $\mathbb{G}_1$ that lie within the valid sphere to be resampled. The vertices from all the split samples are collected into a set $\mathbb{Q}^1 = \bigcup_j \mathbb{Q}_j^1$, with each vertex assigned a valid radius equal to the grid interval of $\mathbb{G}_1$. The sample j that was split is then removed from $\mathbb{Q}^0$.

This recursive process proceeds by iteratively computing the local shading errors for samples in $\mathbb{Q}^k$, and splitting those with errors

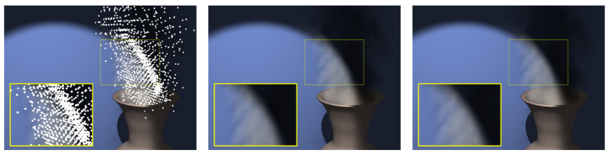

(a) Sampling points (b) Rendering result (c) Reference image

Figure 5: *Capture of sharp shading variations by dynamic sampling.*

greater than ϵ using an increasingly finer grid $\mathbb{G}_{k+1}$. After reaching a specified grid resolution, the final sample set is computed as the union of the sample sets at each grid resolution, $\bigcup_k \mathbb{Q}^k$. The corresponding set of valid spheres covers the volume of the original spheres, such that all points in the volume with significant density will be shaded.

GPU Implementation This algorithm for dynamic sampling can be implemented on the GPU by combining CUDA [NVIDIA 2004] and Cg shaders. The core data structure is a renderable 3D *grid information buffer* that records for each vertex in the corresponding regular grid $\mathbb{G}^k$ an indicator for whether it is currently in the set $\mathbb{Q}^k$. It additionally records the local shading error for the corresponding sample. This data structure can be passed between the CUDA kernel and OpenGL using the interoperability APIs and the pixel buffer object (PBO) extension.

For each iteration, three basic operations are performed: sampling, filtering and splitting. First, the sampling step calls the volume ray tracer and density sampler to compute $L_{x_j}, L_{x_{ij}}, \nabla L_{x_j}, D(x_{ij})$ and $\tau_{x_j v}$, which are used in computing the local shading error (Eq. (5)). Then, a CUDA kernel is invoked to compute the shading error and filter the samples. The scan primitive [Harris et al. 2007] is used to identify samples with errors greater than ϵ. Finally, we split these samples using a standard voxelization of their valid spheres. We implement this splitting using the render-to-3D-texture operation [NVIDIA 2007], with the grid information buffer bound as the rendering target. After splitting, the scan primitive is invoked again to generate the sample points for the next iteration, or to output all the samples if the maximum resolution level is reached.

In our implementation, the maximum resolution for the regular grid is set to half that of the density field, which is $128 \times 128 \times 128$ for all data used in this paper. Specifically, the grid resolutions in our examples are set as follows: $\mathbb{G}_1$ as $16 \times 16 \times 16$, $\mathbb{G}_2$ as $32 \times 32 \times 32$, and $\mathbb{G}_3$ as $64 \times 64 \times 64$. For efficiency in evaluating local shading errors, the value of $\tau_{x_j v}$ for each original valid sphere defined by the Gaussian centers is used for all of its descendant valid spheres in computing Eq. (5).

Result In Fig. 5, a medium is illuminated by a spot light. Our dynamic sampling captures the sharp shading variations well. Starting from the original set of 541 samples, the recursive sample splitting procedure produces a total of 2705 samples, as shown in (a), resulting in a much more faithful capturing of the sharp shading variation(b). A reference image obtained by ray tracing is provided in (c). Please see the accompanying video for a comparison between the original sampling and dynamic sampling.

Like [Ramamoorthi et al. 2007], our sampling scheme operates in a top-down manner. However, instead of computing gradients only at each region center, our method measures source radiance gradients using multiple samples in a valid local neighborhood. Since we deal with shading boundaries instead of soft shadows, this multiple sampling is needed to more reliably detect sharp local changes within a given area.

6 Implementation

In this section, we discuss some implementation details of our rendering pipeline.

Density Field Construction For each frame, the density field is constructed by splatting, with a process similar to the radiance splatting described in Section 4. Here, we splat the weight w_j of each Gaussian instead of the sampled radiance. Unlike for the gradient-based interpolation, no weight normalization is needed. If a residual field hash table exists, we perform splatting with it as well, by retrieving $R(x)$ from the hash table, multiplying it by $\tilde{D}(x)$, and saving it in another color channel. Thus after splatting we have $\tilde{D}(x)$ and $R(x)\tilde{D}(x)$ stored in different color channels. Dividing the latter by the former gives $R(x)$, and adding $R(x)$ to $\tilde{D}(x)$ yields $D(x)$.

Volume Ray Tracing We conduct volume ray tracing for all sample points in a single call. This is done by first packing all the sample points into a 2D texture. A quad of the same size is drawn to trigger the pixel shader, in which volume ray tracing is performed as described in Section 4. To further improve performance, we terminate the tracing of a ray if it exits the volume.

Ray Marching Given the density field and the source radiance field, ray marching is conducted as in [Zhou et al. 2007b]. The RBFs of the density representation are intersected with slices of thickness Δx that are perpendicular to the view direction. Then the slices are rendered from far to near, with alpha blending set to GL_ONE and GL_SRC_ALPHA. The bounding quad of all intersections with the RBFs in each slice is rendered. For each pixel, $D(x)$ and L_x are retrieved from 3D textures, and the RGB channels of the output are set to $D(x)L_x$. The alpha channel is set to the differential transmittance of the slice, computed as $\exp(-\sigma_t D(x)\Delta x)$. After all slices are rendered, we obtain a discrete version of the integration in Eq. (1).

Scene Objects In scenarios where scene objects are present in the medium, we modify Eq. (2) to

$$L = L_s V_{sv} + L_p V_{sp} + \int_p^{x_{in}} \sigma_t D(x) L_x V_{sx} \tau_{xv} dx,$$

where p is the first intersection of the view ray with a scene object, and L_p is the reflected radiance from the surface, computed as $I_0 \tau_{sp} \rho(\overrightarrow{s-p}, \overrightarrow{N})/d_{sp}^2$.[1] The visibility term V_{ab} is a binary function that evaluates to 1 if there exists no scene object blocking a from b, and is equal to 0 otherwise. If the view ray does not intersect a scene object, then p is set to infinity and L_p is zero.

Scene objects affect the computation of L in three ways. First, visibility terms must be incorporated, and can lead to volumetric and cast shadows. Second, they give rise to a new background radiance term L_p. And finally, they may change the starting point of the source radiance integration in the ray march.

To account for the visibility term, we use shadow mapping with a small modification made to the volume tracer. We add a comparison of $\|s - x\|$ to the depth recorded in the shadow map, and exit tracing if $\|s - x\|$ is larger, i.e., x is occluded from s. Note that this modification works for both the dynamic sampling algorithm and the interpolation algorithm. In our implementation, we use variance shadow mapping [Donnelly and Lauritzen 2006] to reduce aliasing.

[1] ρ is the reflectance function and N is the normal of p. The cosine term is merged into ρ for notational simplicity.

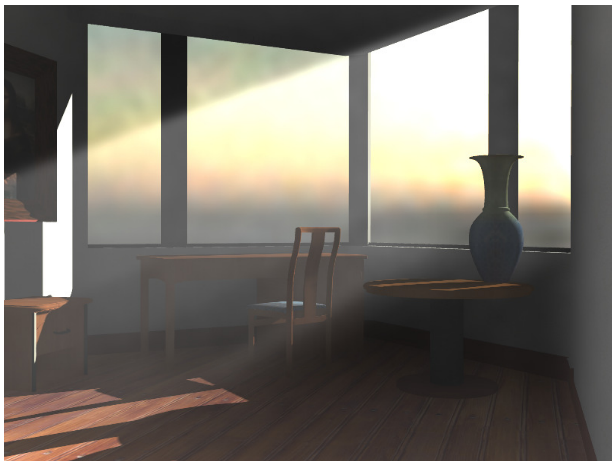


Figure 6: *A dusty room lit by sunlight passing through a window. The light is occluded by the window frame and furniture in the room, generating complex volumetric shadows.*

Scene	# Vertices	# Lights	# Samples	FPS
Fig. 6	24,180	1	3.7k-6.9k	19.2
Fig. 1	16,885	1	1.4k-3.9k	23.2
Fig. 7	16,885	1-30	0.4k-0.8k	28.7

Table 1: *Statistics for the three test scenes. The values for Fig. 1 represent rendering for both daytime and night, using different configurations.*

To compute the reflected radiance L_p on the object surface, the same volume tracer is used to account for the transmittance term. For this, we could use splatting such as in [Gautron et al. 2005]. However, since we compute direct but not indirect illumination, much denser sampling would be required. High curvature regions on the object can also be problematic. Thus, we simply assume that all scene objects are triangulated to a proper scale, and let the graphics hardware linearly interpolate the sampled reflected radiance at vertices.

To account for scene objects in ray marching, we first draw the objects before ray marching, and then leverage the depth culling built into the GPU to correctly attenuate the reflected radiance L_p and exclude slices behind p.

7 Results and Discussion

We implemented our algorithm on a 3.7 GHz PC with 2 GB of memory and an NVidia 8800 GTX graphics card. Images are generated at a 800×600 resolution.

We summarize the statistics of the test scenes in Table 1. For animated versions of the figures, please refer to the supplemental video, which was recorded in real time. In Fig. 1 and Fig. 7, the media animation is generated by simulation, and approximated by a sequence of Gaussian sets and residual fields as in [Zhou et al. 2007b]. For the dusty room scene in Fig. 6, the media data is generated analytically by simply interpolating the density from the floor $(D(x) = 0.6)$ to the ceiling $(D(x) = 0.3)$. Dynamic details are introduced as Perlin noise [2002]. Note that for this example, no preprocessing is required, and the initial samples are obtained by jittered stratified sampling.

The rendering cost is divided into four parts, namely density reconstruction, sampling, gradient interpolation and ray marching. The sampling distribution is dependent on the viewing and lighting conditions, as well as the status of the media data. Sampling and gradient interpolation is the current bottleneck of the algorithm, consuming 60% to 80% of the run time.

Resolution	32^3	64^3	128^3	256^3
ours (512 samples)	73.4	69.1	63.3	45.0
per-voxel	68.0	42.5	11.7	3.5
# Point Lights	1	4	16	64
ours (512 samples)	63.3	57.1	47.6	30.6
per-voxel	11.7	4.2	2.0	0.7
# Spot Lights	1	4	16	64
ours (1.5k$\sim$3.9k samples)	26.4	22.9	17.7	10.4
per-voxel	10.3	4.2	1.9	0.7

Table 2: *Performance (in fps) for different volume resolutions and numbers of light sources. Results are shown for point light sources, which do not require dynamic sampling, and spot light sources.*

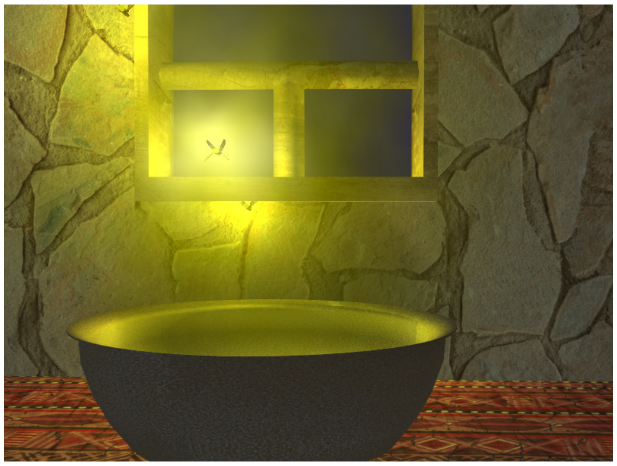


Figure 7: *The scene in Fig. 1 lit by numerous dynamic point light sources. The point light sources are considered local and do not generate shadows. Our algorithm scales well with the number of sources, and renders at* 21.5 *fps when the scene is lit by 50 point light sources.*

Our algorithm scales well with respect to volume resolution and number of light sources. In Table 2, it is shown that the scalability of our algorithm is less sensitive than per-voxel volume tracing to volume resolution and light source number. This results from calling the volume ray tracer at only a very limited number of sample points. Volume tracing, whose performance is directly related to the volume resolution and light number, accounts for a small portion (15%-30%) of the overall cost. In contrast, it can account for up to 90% of the overall cost if volume tracing is performed at each voxel.

For a volume resolution of $128 \times 128 \times 128$, the video memory cost of our algorithm includes a 4 MB density buffer, a 6 MB radiance buffer used as the target for sample splatting, three grid information buffers totaling about 2.5 MB, and a temporary buffer of 10 MB used for loading residual tables of subsequent frames. In the host memory, the main cost is the residual tables, which add up to 79MB for the data in Fig. 1 and Fig. 7. The residual table costs may be avoided by using an analytical+noise representation of the medium, as done for Fig. 6.

8 Conclusion

In this work, we presented a technique for real-time rendering of inhomogeneous media with all-frequency shading effects. The accuracy and efficiency of this system is gained from the gradient-based interpolation and the dynamic sampling of source radiance with respect to local shading errors.

Our current formulation utilizes certain assumptions that limit its generality, such as an isotropic phase function and a finite volume representation. Also, the algorithm does not consider the influence of media on surface shading, i.e., scattering prior to surface reflection. These are issues we intend to examine in future work.

In addition, our method presently addresses only single scattering of radiance in the medium. Within our framework, fast evaluation of multiple scattering might also be possible, by computing it at only the sample points and interpolating throughout the volume. Also, the current technique has demonstrated real-time rendering results for only compact illumination sources. By employing the spherical harmonics domain processing of [Zhou et al. 2007b], we believe that an extension of our method to environment lighting may be within reach as well.

References

BIRI, V., ARQUES, D., AND MICHELIN, S. 2006. Real time rendering of atmospheric scattering and volumetric shadows. *Journal of WSCG 14*, 65–72.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. In *Proceedings of SIGGRAPH 82*, 21–29.

CEREZO, E., PÉREZ, F., PUEYO, X., SERÓN, F. J., AND SILLION, F. X. 2005. A survey on participating media rendering techniques. *The Visual Computer 21*, 5, 303–328.

CHANDRASEKHAR, S. 1960. *Radiative Transfer*. Dover Publications Inc.

DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. 2000. A simple, efficient method for realistic animation of clouds. In *Proceedings of SIGGRAPH 00*, 19–28.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2002. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proceedings of Graphics Hardware Workshop 02*, 99–107.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of I3D 06*, 161–165.

EBERT, D. S., AND PARENT, R. E. 1990. Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. In *Proceedings of SIGGRAPH 90*, 357–366.

GAUTRON, P., KŘIVÁNEK, J., BOUATOUCH, K., AND PATTANAIK, S. 2005. Radiance cache splatting: A gpu-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering 05*, 55–64.

HARRIS, M. J., AND LASTRA, A. 2001. Real-time cloud rendering. In *Proceedings of Eurographics 01*, 76–84.

HARRIS, M., SENGUPTA, S., AND OWENS, J. 2007. Parallel prefix sum (scan) in CUDA. In *GPU Gems 3*, Addison Wesley, Chapter 39.

HEGEMAN, K., ASHIKHMIN, M., AND PREMOZE, S. 2005. A lighting model for general participating media. In *Proceedings of I3D*, 117–124.

JAROSZ, W., DONNER, C., ZWICKER, M., AND JENSEN, H. W., 2007. Radiance caching for participating media. SIGGRAPH Sketch (Conditionally accepted to ACM Transactions on Graphics).

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of SIGGRAPH 98*, 311–320.

KAJIYA, J. T., AND HERZEN, B. P. V. 1984. Ray tracing volume densities. In *Proceedings of SIGGRAPH 84*, 165–174.

KNISS, J., PREMOZE, S., HANSEN, C., SHIRLEY, P., AND MCPHERSON, A. 2003. A model for volume lighting and modeling. *IEEE Trans. Visualization and Computer Graphics 9*, 2, 150–162.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1996. Rendering participating media with bidirectional path tracing. In *Proceedings of Eurographics Workshop on Rendering 96*, 91–100.

LEVOY, M. 1990. Efficient ray tracing of volume data. *ACM Trans. Graph. 9*, 3, 245–261.

MAX, N. L. 1986. Atmospheric illumination and shadows. In *Proceedings of SIGGRAPH 86*, 117–124.

MAX, N. L. 1994. Efficient light propagation for multiple anisotropic volume scattering. In *Proceedings of Eurographics Workshop on Rendering 94*, 87–104.

NARASIMHAN, S. G., AND NAYAR, S. K. 2003. Shedding light on the weather. In *Proceedings of CVPR*, 665–672.

NVIDIA, 2004. CUDA homepage. http://developer.nvidia.com/object/cuda.html.

NVIDIA, 2007. Render to 3d texture. NVIDIA OpenGL SDK 10 Code Samples. http://developer.download.nvidia.com/SDK/10/ opengl/samples.html.

PERLIN, K. 2002. Improving noise. In *Proceedings of SIGGRAPH 02*, 681–682.

PIGHIN, F., COHEN, J., AND SHAH, M. 2004. Modeling and editing flows using advected radial basis functions. In *Proccedings of Eurographics Symposium on Computer Anmiation 04*, 223–232.

RAMAMOORTHI, R., MAHAJAN, D., AND BELHUMEUR, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2.

RILEY, K., EBERT, D. S., KRAUS, M., TESSENDORF, J., AND HANSEN, C. 2004. Efficient rendering of atmospheric phenomena. In *Proceedings of Eurographics Symposium on Rendering 04*, 375–386.

RUSHMEIER, H. E., AND TORRANCE, K. E. 1987. The zonal method for calculating light intensities in the presence of a participating medium. In *Proceedings of SIGGRAPH 87*, 293–302.

SAKAS, G. 1990. Fast rendering of arbitrary distributed volume densities. In *Eurographics 90*, 519–530.

SCHPOK, J., SIMONS, J., EBERT, D. S., AND HANSEN, C. 2003. A real-time cloud modeling, rendering, and animation system. In *Proceedings of Eurographics Symposium on Computer Anmiation 03*, 160–166.

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

STAM, J. 1995. *Multi-Scale Stochastic Modelling of Complex Natural Phenomena*. PhD thesis, Dept. of Computer Science, University of Toronto.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S., AND NAYAR, S. 2005. A practical analytic single scattering model for real time rendering. *ACM Trans. Graph. (SIGGRAPH '05) 24*, 3, 1040–1049.

WARD, G. J., AND HECKBERT, P. 1992. Irradiance Gradients. In *Proceedings of Eurographics Workshop on Rendering 92*, 85–98.

ZHOU, K., HOU, Q., GONG, M., SNYDER, J., GUO, B., AND SHUM, H.-Y. 2007. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. *Proceedings of Pacific Graphics 07*, 116–125.

ZHOU, K., REN, Z., LIN, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2007. Real-time smoke rendering using compensated ray marching. Tech. rep. http://research.microsoft.com/research/pubs/view.aspx?tr_id=1385.