Real-time Smoke Rendering Using Compensated Ray Marching

Kun Zhou* Zhong Ren* Stephen Lin* Hujun Bao† Baining Guo* Heung-Yeung Shum*

*Microsoft Research Asia †Zhejiang University

Abstract

We present a real-time algorithm called *compensated ray marching* for rendering of smoke under dynamic low-frequency environment lighting. Our approach is based on a decomposition of the input smoke animation, represented as a sequence of volumetric density fields, into a set of radial basis functions (RBFs) and a sequence of residual fields. From this low-frequency RBF approximation of density fields, an efficient technique is proposed for computing source radiance distributions throughout the volume. Source radiances from single and multiple scattering are directly computed at only the RBF centers and then approximated at other points in the volume using an RBF-based interpolation. With the computed source radiances, a slice-based integration of radiance along each viewing ray is performed to render the final image. During this ray marching process, the residual fields are compensated back into the radiance integral to generate images of high detail.

The runtime algorithm, which includes both light transfer simulation and ray marching, can be easily implemented on the GPU, and thus allows for real-time manipulation of viewpoint and lighting, as well as interactive editing of smoke attributes such as extinction cross section, scattering albedo, and phase function. With only moderate preprocessing time and storage, this technique generates rendering results that are comparable to those from off-line rendering algorithms like ray tracing.

Keywords: participating media, environment lighting, single scattering, multiple scattering, perfect hashing

1 Introduction

Rendering of smoke presents a challenging problem in computer graphics because of its complicated effects on light propagation. Within a smoke volume, light undergoes absorption and scattering interactions that vary from point to point because of the spatial non-uniformity of smoke. In static participating media, the number and complexity of scattering interactions lead to a substantial expense in computation. For a dynamic medium like smoke, whose intricate volumetric structure changes with time, the computational costs can be prohibitive.

Despite the practical difficulties of smoke rendering, it nevertheless remains a popular element in many applications such as films and games. To achieve the desired visual effects of smoke, a designer should be afforded real-time control over the lighting environment and vantage point, as well as the volumetric distribution and optical properties of the smoke.

In this work, we present a real-time algorithm called *compensated ray marching* for rendering smoke animations with dynamic low-frequency environment lighting and controllable smoke attributes. As in most multiple scattering techniques (e.g., [Kajiya and Herzen 1984]), our runtime algorithm first simulates light transfers to compute the source radiances inside the volume, then integrates the radiance contributions in a ray march along each viewing ray. To expedite this process, we propose a technique based on a decomposition of the smoke volume into a low-frequency approximation and a residual field. In the low-frequency approximation, the smoke volume is modeled by a set of radial basis functions (RBFs). Source radiances in this RBF model can be rapidly evaluated by directly computing the radiances at only the RBF centers, and then approximating the radiances at other points inside the volume using an RBF-based interpolation. For fast computation of radiances at RBF centers, we employ an adaptation of the spherical harmonic exponentiation (SHEXP) technique [Ren et al. 2006] to evaluate single scattering and initialize an iterative multiple scattering procedure. Since evaluation of source radiances comprises the bulk of computation in a participating media simulation, this low-frequency approximation of the smoke density field and source radiances leads to a considerable speedup in rendering, which then enables interactive manipulation.

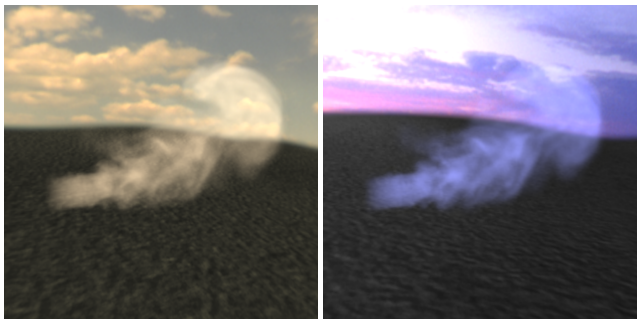


Figure 1: *Real-time rendering of smoke under dynamic environment lighting. The appearance of a smoke volume can change significantly with respect to illumination.*

The appearance of fine-scale smoke details such as vortices depends on the high-frequency density components contained in the residuals. To incorporate these smoke details in an efficient manner, we compensate for the residuals in the ray march by accounting for their extinction effects in the radiance integration along viewing rays. Since residual fields have significant values only at sparse locations, we take advantage of the perfect spatial hashing [Lefebvre and Hoppe 2006] technique to substantially compress and rapidly reconstruct this data.

The runtime algorithm, which includes both light transfer simulation and ray marching, can be easily implemented on the GPU, and thus offers users real-time feedback for changes in viewpoint, lighting, and smoke attributes such as extinction cross section, scattering albedo, and phase function. With practical amounts of preprocessing time and storage, we demonstrate the rendering results of compensated ray marching to be comparable to those from offline rendering algorithms like ray tracing, as shown in Fig. 6 and the supplemental video. With this technique, interactive modifications of scattering properties becomes feasible not only for animated sequences of smoke, but also of other non-emissive media such as mist, steam, and dust.

2 Related Work

Extensive research has been done on realistic simulation of participating media [Cerezo et al. 2005]. While impressive renderings have been generated for static scenes with previous techniques, they do not offer a way to render animated sequences in real time. Here, we limit our discussion to a small number of representative methods for efficient simulation.

Analytic methods Blinn [1982] introduced an analytic technique for rendering single scattering in homogeneous media with low scattering albedo and an infinitely distant light source. For lighting that resides within a homogeneous medium, Narasimhan and Nayar [2003] proposed a multiple scattering model for optically thick media, and [Biri et al. 2004; Sun et al. 2005] presented single scattering formulas that can be evaluated in real time on programmable graphics hardware. The method in [Sun et al. 2005] can be generalized to handle environment illumination and the effects of media on surface reflectance. This approach has been generalized in [Anonymous 2007] to render single scattering in smooth, optically thin, inhomogeneous media in which there exists little multiple scattering. The medium is modelled using radial basis functions, without accounting for the high-frequency residuals needed to represent fine details in media such as smoke. Due to its inhomogeneity and significant multiple scattering of light, smoke is not well-suited for analytic formulation.

Stochastic methods Another approach is to approximate the overall scattering behavior using only a small number of random samples, rather than evaluate the numerous samples of a full participating media simulation. High quality global illumination effects have been produced with Monte Carlo path tracing methods [Lafortune and Willems 1996]. Stam [1994] introduced randomness by incorporating stochastic intensity perturbations according to statistics computed from the density field and an illumination model. Methods based on volume photon maps [Jensen and Christensen 1998; Fedkiw et al. 2001] trace a relatively small number of rays, and determine source radiance at a point based on light interactions within its neighborhood. While these strategies for sampling reduction can significantly decrease computation, considerable simulation time is still needed to render a single image, making this approach inappropriate for interactive applications on animated sequences.

Numerical simulations In contrast to stochastic methods, many techniques compute the radiance transport integral in a deterministic manner. Kajiya and Herzen [1984] computed the source radiance of voxels in a spherical harmonics basis whose coefficients are computed from a system of partial differential equations (PDEs), and then integrated radiances along viewing rays. Rushmeier [1987; 1988] presented zonal finite element methods for isotropic scattering in participating media. Stam [1995] used blobs to model a density volume, and computed energy transport among blobs based on diffusion theory. Geist et al. [2004] also computed multiple scattering as a diffusion process, using a lattice-Boltzmann method as a PDE system solver. These approaches to radiance transport simulation, while faster than conventional path tracing, nevertheless require offline computation.

Simplified volume representations have also been used towards obtaining high performance. Dobashi et al. [2000] represented clouds as a set of metaballs, for which isotropic single scattering is computed at their centers, and then their radiance contributions are composited by a billboard-based blending. For smoke, we utilize an RBF representation of the volume density to facilitate computation of source radiances, including both single and multiple scattering. Moreover, the RBF model is used in conjunction with model residuals in the ray march to obtain real-time rendering results with high-frequency smoke details. While the method in [Dobashi et al. 2000] handles only directional lighting, ours addresses complex environment illumination.

Precomputation techniques Efficient rendering of participating media can also be achieved through precomputation of various scene-dependent quantities. In [Harris and Lastra 2001], static clouds illuminated by multiple directional light sources are rendered using precomputed shading information. Precomputed radiance transfer (PRT) [Sloan et al. 2002] can be applied to volumetric models in addition to surfaces. In PRT, a light transfer vector is precomputed at each voxel, and is used in iteratively forward scattering the radiance until it exits the volume. For a medium represented by a particle system, Szirmay-Kalos et al. [2005] precomputed a visibility network and opacity values between visible particles. With this information, multiple scattering among the particles is iteratively evaluated at run time in a manner that reuses radiance transport that has been computed along the network paths. From the observation that multiple scattering causes blurring and attenuation of light, Premoze et al. [2004] presented an analytic formulation of multiple scattering based on point spread functions precomputed from the optical properties within the medium. This technique was later modified by Hegeman et al. [2005] to perform path integration on programmable graphics hardware. In all of these methods, the precomputed quantities are valid only for the given static participating medium. For dynamic smoke sequences with adjustable smoke parameters, the preprocessing time and storage costs would be prohibitive.

x	A 3D point
s, ω	Direction
x_ω	A point where light enters the medium along direction ω
ω_i, ω_o	Incident, outgoing radiance direction
S	Sphere of directions
$D(x)$	Smoke density
σ_t	Extinction cross section
σ_s	Scattering cross section
Ω	Single-scattering albedo, computed as σ_s/σ_t
$\kappa_t(x)$	Extinction coefficient, computed as $\sigma_t D(x)$
$\tau(u,x)$	Transmittance from u to x, computed as $\exp(-\int_u^x \kappa_t(v)dv)$
$\tau_\infty(x,\omega)$	Transmittance from infinity to x from direction ω, computed as $\exp(-\int_x^{\infty_\omega} \kappa_t(v)dv)$
$L_{in}(\omega)$	Environment map
$L_{out}(x,\omega)$	Outgoing radiance
$J(x,\omega)$	Source radiance
$p(\omega_o,\omega_i)$	Phase function
$\boldsymbol{y}(s)$	Set of spherical harmonic basis functions
$y_i(s)$, $y_l^m(s)$	Spherical harmonic basis function
$\boldsymbol{f}_i$, $\boldsymbol{f}_l^m$	Spherical harmonic coefficient vector

Table 1: *Notation*

3 Notation and Background

Let the lighting be represented as a low-frequency environment map L_{in}, described by a vector of spherical harmonic coefficients $\boldsymbol{L}_{in}$. A sequence of volumetric density fields is used to model the input smoke animation. At each frame, the smoke density is denoted as $D(x)$. In the following, we briefly describe light transport in scattering media and review operations on spherical harmonics. Table 1 lists the notation used in this paper.

Light Transport in Scattering Media To describe light transport in scattering media, we utilize several radiance quantities, which we define as in [Cerezo et al. 2005]. As shown in Fig. 2, the radiance at a point x is composed of the reduced incident radiance L_d and the media radiance L_m:

$$L_{out}(x,\omega_o) = L_d(x,\omega_o) + L_m(x,\omega_o).$$

The reduced incident radiance represents incident radiance L_{in} along direction ω_o that has been attenuated by the medium before

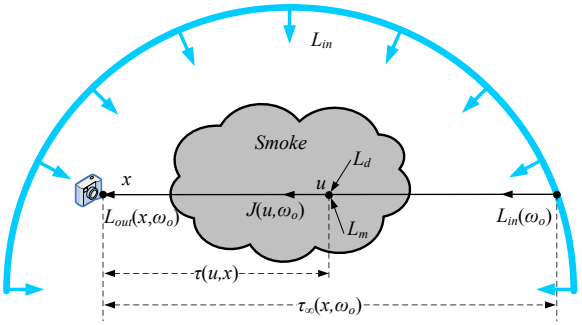


Figure 2: *Light transport in smoke.*

arriving at x:

$$L_d(x,\omega_o) = \tau_\infty(x,\omega_o) L_{in}(\omega_o). \tag{1}$$

The media radiance L_m is the integration of the source radiance J that arrives at x along direction ω_o from points within the medium:

$$L_m(x,\omega_o) = \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J(u,\omega_o) du.$$

In non-emissive media such as smoke, this source radiance J is composed of a single scattering J_{ss} and multiple scattering J_{ms} component:

$$J(u,\omega_o) = J_{ss}(u,\omega_o) + J_{ms}(u,\omega_o).$$

The single scattering term, J_{ss}, represents the first scattering interaction of the reduced incident radiance:

$$J_{ss}(u,\omega_o) = \frac{\Omega}{4\pi} \int_S L_d(u,\omega_i) p(\omega_o,\omega_i) d\omega_i. \tag{2}$$

The multiple scattering term, J_{ms}, accounts for scattering of the media radiance:

$$J_{ms}(u,\omega_o) = \frac{\Omega}{4\pi} \int_S L_m(u,\omega_i) p(\omega_o,\omega_i) d\omega_i. \tag{3}$$

Spherical Harmonics Low-frequency spherical functions can be efficiently represented in terms of spherical harmonics (SHs). A spherical function $f(s)$ can be projected onto a basis set $\boldsymbol{y}(s)$ to obtain a vector $\boldsymbol{f}$ that represents its low-frequency components:

$$\boldsymbol{f} = \int_S f(s)\boldsymbol{y}(s) ds. \tag{4}$$

An order-n SH projection has n^2 vector coefficients. With these coefficients, we can reconstruct a spherical function $\tilde{f}(s)$ that approximates $f(s)$:

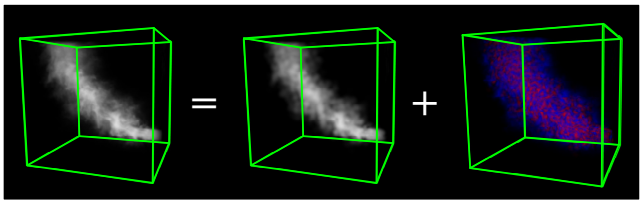
$$\tilde{f}(s) = \sum_{i=0}^{n^2-1} \boldsymbol{f}_i y_i(s) = \boldsymbol{f} \cdot \boldsymbol{y}(s). \tag{5}$$

The *SH triple product*, denoted by $\boldsymbol{f} * \boldsymbol{g}$, represents the order-$n$ projected result of multiplying the reconstructions of two order-n vectors:

$$\boldsymbol{f} * \boldsymbol{g} = \int_S f(s)\, g(s)\, \boldsymbol{y}(s)\, ds \;\Rightarrow\; (\boldsymbol{f} * \boldsymbol{g})_i = \sum_{j,k} \Gamma_{ijk} \boldsymbol{f}_j \boldsymbol{g}_k,$$

where the SH triple product tensor Γ_{ijk} is defined as

$$\Gamma_{ijk} = \int_S y_i(s) y_j(s) y_k(s) ds.$$



(a) original data (b) RBF approx. (c) residual ($\times 16$)

Figure 3: *Density field approximation. (a) Original volume density $D(x)$; (b) RBF approximation $\tilde{D}(x)$; (c) residual field $R(x)$, scaled by 16 for better viewing. Red indicates positive residuals, and blue is negative.*

Γ_{ijk} is symmetric, sparse, and of order 3.

SH convolution, denoted by $\boldsymbol{f} \star \boldsymbol{g}$, represents the order-$n$ projected result of convolving the reconstructions of two order-n vectors:

$$\boldsymbol{f} \star \boldsymbol{g} = \int_S \int_S f(t) g(R_s(t)) \boldsymbol{y}(s)\, dt\, ds \;\Rightarrow\; (\boldsymbol{f} * \boldsymbol{g})_l^m = \sqrt{\frac{4\pi}{2l+1}}\, \boldsymbol{f}_l^m \boldsymbol{g}_l^0,$$

where $g(s)$ is a circularly symmetric function, and R_s is a rotation along the elevation angle towards direction s (i.e., the angle between the positive z-axis and direction s).

SH exponentiation, denoted by $\exp_*(\boldsymbol{f})$, represents the order-n projected result of the exponential of a reconstructed order-n vector:

$$\exp_*(\boldsymbol{f}) = \int_S \exp(f(s)) \boldsymbol{y}(s) ds.$$

This can be efficiently calculated on the GPU using the optimal linear approximation described in [Ren et al. 2006]:

$$\exp_*(\boldsymbol{f}) \approx \exp\left(\frac{\boldsymbol{f}_0}{\sqrt{4\pi}}\right) \left(a(\|\hat{\boldsymbol{f}}\|)\mathbf{1} + b(\|\hat{\boldsymbol{f}}\|)\hat{\boldsymbol{f}}\right),$$

where $\hat{\boldsymbol{f}} = (0, \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{n^2-1})$, $\mathbf{1} = (\sqrt{4\pi}, 0, 0, ..., 0)$, and a, b are tabulated functions of the magnitude of input vector $\hat{\boldsymbol{f}}$.

4 Algorithm Overview

Our approach consists of a preprocessing step and a runtime rendering algorithm.

Preprocessing As shown in Fig. 3, the density field $D(x)$ is first decomposed into a weighted sum of RBFs $B_\ell(x)$ and a residual $R(x)$:

$$D(x) = \tilde{D}(x) + R(x) = \sum_\ell w_\ell B_\ell(x) + R(x).$$

Each RBF $B_\ell(x)$ is defined by its center c_ℓ and radius r_ℓ:

$$B_\ell(x) = G(\|x - c_\ell\|, r_\ell),$$

where $G(t,\lambda)$ is a radial basis function of the following form [Wyvill and Trotman 1990]:

$$G(t,\lambda) = \begin{cases} 1 - \frac{22t^2}{9\lambda^2} + \frac{17t^4}{9\lambda^4} - \frac{4t^6}{9\lambda^6}, & \text{if } t \le \lambda; \\ 0, & \text{otherwise.} \end{cases}$$

These RBFs are approximately Gaussian in shape, have local support, and are C^1 continuous.

The RBF model $\tilde{D}(x)$ represents a low-frequency approximation of the smoke density field. In general, the residual field $R(x)$ contains a relatively small number of significant values. To promote compression, we zero the values below a given threshold, such that the residual becomes sparse.

(a) single scattering (b) multiple scattering (c) compensated ray marching

Figure 4: *Source radiance components and compensated ray marching result. The smoke is illuminated with the KITCHEN environment map, shown as insets at the upper-right.*

Runtime In a participating media simulation, the computational expense mainly lies in the evaluation of source radiances J from the density field D. To expedite this process, we compute an approximation $\tilde{J}$ of the source radiances from the RBF model $\tilde{D}$ of the density field. Specifically, we compute $\tilde{J}$ due to single and multiple scattering at only the RBF centers:

$$\tilde{J}(c_\ell) = \tilde{J}_{ss}(c_\ell) + \tilde{J}_{ms}(c_\ell).$$

The source radiance at any point x in the medium is then approximated as a weighted combination of the source radiances at these centers:

$$J(x,\omega_o) \approx \tilde{J}(x,\omega_o) = \frac{1}{\tilde{D}(x)} \sum_\ell w_\ell B_\ell(x) \tilde{J}(c_\ell,\omega_o).$$

An example of single scattering and multiple scattering source radiances in a smoke volume is illustrated in Fig. 4 (a) and (b).

After computation of source radiances, we perform a ray march to compute the media radiance $L_m(x,\omega_o)$ by gathering radiance contributions towards the viewpoint. These radiance contributions consist of a component $\tilde{L}_m$ computed from the approximated source radiances $\tilde{J}$ and the RBF density model, and a component $\tilde{C}_m$ that compensates for the density field residual:

$$\begin{aligned} L_m(x,\omega_o) &\approx \tilde{L}_m(x,\omega_o) + \tilde{C}_m(x,\omega_o) \\ &= \sum_{j=1}^{N} \tilde{\tau}(x_j,x)\sigma_t \tilde{D}(x_j)\tilde{J}(x_j,\omega_o) + \\ &\quad \sum_{j=1}^{N} \tilde{\tau}(x_j,x)\sigma_t R(x_j)\tilde{J}(x_j,\omega_o), \end{aligned} \tag{6}$$

where $\tilde{\tau}$ denotes transmittance values computed from $\tilde{D}$, and x_j indexes a set of N uniformly distributed points between x_{ω_o} and x. The compensation term $\tilde{C}_m$ brings into the ray march the extinction effects of the density residuals, as shown in Fig. 4 (c).

Due to its size, a density field D cannot in general be effectively processed on the GPU. However, the decomposition of a density field into an RBF approximation and a residual field leads to a considerable reduction in data size. Because of its sparsity, we observe that the residual can be highly compressed using the perfect spatial hashing technique [Lefebvre and Hoppe 2006]. In addition to providing manageable storage costs, perfect spatial hashing also offers fast reconstruction of the residual in the ray march. By incorporating high-frequency smoke details in this manner, high-quality smoke renderings can be generated as exemplified in Fig. 6 and Fig. 9.

In this formulation, we obtain real-time performance with high visual fidelity by accounting for the cumbersome residual field only where it has a direct impact on Eq. (6), namely, in the smoke density values. For these smoke densities, the residual can be efficiently retrieved from the hash table using perfect spatial hashing. In the other factors of Eq. (6), the residual has only an indirect effect and also cannot be rapidly accounted for. We therefore exclude the residuals from computations of source radiances and transmittances to significantly speed up processing without causing significant degradation of smoke appearance, as will be shown in Section 5.

5 Algorithm Details

5.1 Density Field Approximation

Given the number n of RBFs, we compute an optimal approximation of the smoke density field by solving the following minimization problem:

$$\min_{c_\ell, r_\ell, w_\ell} \left(\sum_{j,k,l} \left[D(x_{jkl}) - \sum_{\ell=1}^{n} w_\ell B_\ell(x_{jkl}) \right]^2 \right), \tag{7}$$

where (j,k,l) indexes a volumetric grid point at position x_{jkl}. For optimization, we employ the L-BFGS-B minimizer [Zhu et al. 1997], which was used in [Tsai and Shih 2006] to fit spherical RBFs to a lighting environment, and in [Sloan et al. 2005] to fit zonal harmonics to a radiance transfer function.

Since L-BFGS-B is a derivative-based method, at each iteration we need to provide the minimizer with the objective function and the partial derivatives for each variable, which are listed in the Appendix. Evaluation of these quantities requires iterating over all voxels in the volume, $100^3 \sim 128^3$ for the examples in this paper. To reduce computation, we take advantage of the sparsity of the smoke data by processing only voxels within the support range of an RBF. To avoid entrapment in local minima at early stages of the algorithm, we also employ a teleportation scheme similar to that in [Cohen-Steiner et al. 2004], where we record the location of the maximum error during the approximation procedure and then move the most insignificant basis function there. We utilize teleportations at every 20 iterations of the minimizer, and when the minimizer converges we teleport the most insignificant RBF alternatingly to the location of maximum error or to a random location with nonzero data. The inclusion of teleportation into the minimization process often leads to a further reduction of the objective function by $20\% \sim 30\%$.

Depending on the smoke density distribution in each frame, a different number n of RBFs should be utilized to provide a balance between accuracy and efficiency. We start with a large number (1000) of RBFs in each frame, then after optimization by Eq. (7) we merge RBFs that are in close proximity. Specifically, B_ℓ and B_h are merged together if their centers satisfy the condition $\|c_\ell - c_h\| < \varepsilon$. The center of the new RBF is then taken as the midpoint between c_ℓ

and c_h. After this merging process, we fix the RBF centers and optimize again with respect to only r_ℓ and w_ℓ. In our current implementation, ε is set to 2Δ, where Δ is the distance between neighboring grid points in the volume.

To accelerate this process, we take advantage of the temporal coherence in smoke animations by initializing the RBFs of a frame with the optimization result of the preceding frame before merging. For the first frame, the initialization is generated randomly. Performance results for this RBF fitting scheme are listed in Table 2.

5.2 Residual Field Compression

After computing the RBF approximation of the density field, the residual density field $R(x) = D(x) - \tilde{D}(x)$ is then compressed for GPU processing. While the residual field is of the same resolution as the density field, it normally consists of small values. We zero entries in $R(x)$ below a given threshold ($0.005 \sim 0.01$ for the examples in this paper), and then compress the resulting sparse residual field by perfect spatial hashing [Lefebvre and Hoppe 2006], which is lossless and ideally suited for parallel evaluation on graphics hardware.

In our implementation of perfect spatial hashing, we utilize a few modifications tailored to our application. Unlike in [Lefebvre and Hoppe 2006] where nonzero values in the data lie mostly along the surface of a volume, the nonzero values in our residual fields are distributed throughout the volume. Larger offset tables are therefore needed, so we set the initial table size to $\sqrt[3]{K/3}+9$ (K is the number of nonzero items in the volume), instead of $\sqrt[3]{K/6}$ used in [Lefebvre and Hoppe 2006].

In processing a sequence of residual fields, we tile several consecutive frames into a larger volume on which hashing is conducted. Since computation is nonlinear to the number of domain slots, we construct a set of smaller hash volumes instead of tiling all the frames into a single volume. With smaller hash volumes, we also avoid the precision problems that arise in decoding the domain coordinates of large packed volumes. In our implementation, we tile $3^3 = 27$ frames per volume.

5.3 Single Scattering

To promote runtime performance, source radiance values in the smoke volume are calculated using the low-frequency RBF approximation of the density field, $\tilde{D}$. We compute single scattering at the RBF centers according to Eq. (2):

$$\begin{aligned} \tilde{J}_{ss}(c_\ell, \omega_o) &= \frac{\Omega}{4\pi}\int_S \tilde{L}_d(c_\ell, \omega_i) p(\omega_o, \omega_i) d\omega_i \\ &= \frac{\Omega}{4\pi}\int_S L_{in}(\omega_i)\tilde{\tau}_\infty(c_\ell, \omega_i) p(\omega_o, \omega_i) d\omega_i, \end{aligned} \tag{8}$$

where $\tilde{\tau}_\infty(c_\ell, \omega_i) = \exp(-\int_{c_\ell}^{\infty\omega_i} \sigma_t \tilde{D}(u)du)$ is the approximated transmittance along direction ω_i from infinity to c_ℓ.

In scattering media, phase functions are often well-parameterized by the angle θ between the incoming and outgoing directions. For computational convenience, we therefore rewrite $p(\omega_o, \omega_i)$ as a circularly symmetric function $p(z)$, where $z = \cos\theta$. With this reparameterization of the phase function, Eq. (8) can be efficiently computed using the SH triple product and convolution:

$$\tilde{\boldsymbol{J}}_{ss}(c_\ell) = \frac{\Omega}{4\pi}\left[(\boldsymbol{L}_{in} * \tilde{\boldsymbol{\tau}}(c_\ell)) \star \boldsymbol{p}\right]. \tag{9}$$

$\boldsymbol{L}_{in}$ and $\boldsymbol{p}$ are precomputed according to Eq. (4). In the following, we describe how to efficiently compute the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ on the fly.

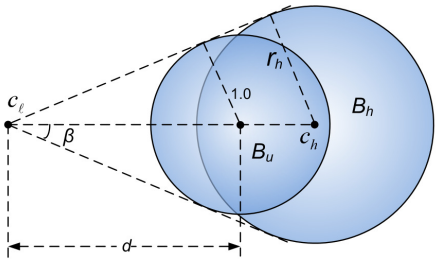


Figure 5: *The accumulative optical depth of an RBF B is determined by the angle β subtended by half the RBF and its absolute radius r. For fast evaluation, the precomputed value for a unit-radius RBF B_u with angle β is retrieved, rotated, and then multiplied by r.*

Computing the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ Expressing transmittance directly in terms of the RBFs, we have

$$\begin{aligned} \tilde{\tau}(c_\ell, \omega_i) &= \exp\left(-\sigma_t \textstyle\sum_h w_h \int_{c_\ell}^{\infty\omega_i} B_h(u)du\right) \\ &= \exp\left(-\sigma_t \textstyle\sum_h w_h T_h(c_\ell, \omega_i)\right), \end{aligned}$$

where $T_h(c_\ell, \omega_i)$ is the accumulative optical depth through RBF B_h. $\tilde{\boldsymbol{\tau}}(c_\ell)$ can then be computed as

$$\tilde{\boldsymbol{\tau}}(c_\ell) = \exp_*\left(-\sigma_t \sum_h w_h \boldsymbol{T}_h(c_\ell)\right),$$

where $\boldsymbol{T}_h(c_\ell)$ is the SH projection of $T_h(c_\ell, \omega_i)$.

For efficient determination of $\boldsymbol{T}_h(c_\ell)$, we tabulate the accumulative optical depth vector $\boldsymbol{T}(\beta)$ with respect to angle β, equal to half the subtended angle of a unit-radius RBF B_u as illustrated in Fig. 5. Note that β ranges from 0 to π:

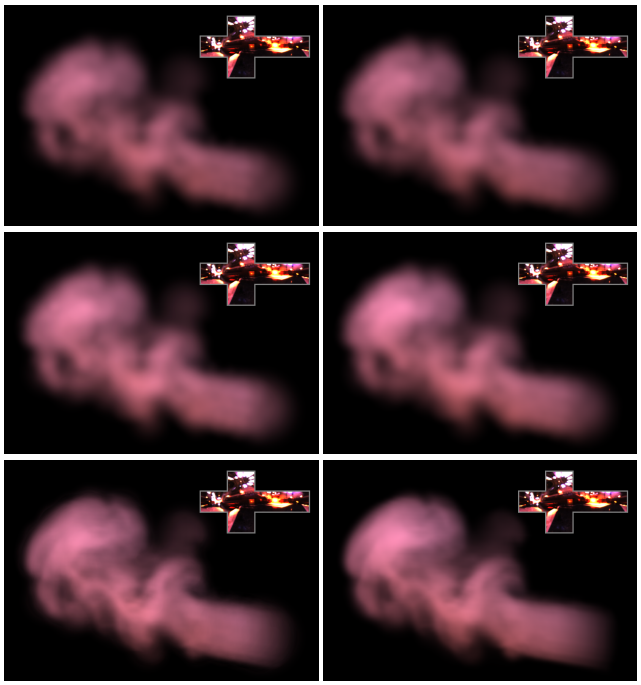
$$\beta = \begin{cases} \arcsin(1/d), & \text{if } d > 1 \\ \arccos(d) + \pi/2, & \text{otherwise} \end{cases}$$

where d is the distance from c_ℓ to the center of B_u. Since the RBF kernel function G is symmetric, this involves a 1D tabulation of zonal harmonic (ZH) vectors.

At runtime, we retrieve ZH vectors $\boldsymbol{T}(\beta_{\ell,h})$ from the table, where $\beta_{\ell,h}$ is half the angle subtended by RBF B_h as seen from c_ℓ. $\boldsymbol{T}_h(c_\ell)$ is then computed by rotating $\boldsymbol{T}(\beta_{\ell,h})$ to the axis determined by c_ℓ and c_h, followed by a multiplication with the radius r_h. We note that this computation is analogous to that in [Ren et al. 2006], except that here we are dealing with accumulative optical depth instead of log space visibility.

Computation of the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ is then straightforward. For each RBF center, we iterate through the RBFs. Their accumulative optical depth vectors are retrieved, rotated, scaled, and summed up to yield the total optical depth vector. Finally, it is multiplied by the negative extinction cross section and exponentiated to yield the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$. With this transmittance vector, the source radiance due to single scattering is computed from Eq. (9).

Comparison This single scattering approximation yields results similar to those obtained from ray tracing, as shown in the top row of Fig. 6. For a clearer comparison, we perform ray tracing with the approximated density field $\tilde{D}(x)$, and compare it with our single scattering result. In rendering the single scattering image, the ray marching algorithm described in Section 5.5 is used without accounting for the residual.



(a) our result (b) ray tracing

Figure 6: *Comparison between our results and ray tracing. Top: single scattering in an approximated density field. Middle: single and multiple scattering in the approximated density field. Bottom: final results for the original density field.*

5.4 Multiple Scattering

Source radiance J_{ms} due to multiple scattering is expressed in Eq. (3). In evaluating J_{ms}, we first group light paths by the number of scattering events they include:

$$J_{ms}(x,\omega_o) = J^2_{ms}(x,\omega_o) + J^3_{ms}(x,\omega_o) + \dots .$$

J^k_{ms} represents source radiance after k scattering events, computed from media radiance that has scattered $k-1$ times:

$$J^k_{ms}(x,\omega_o) = \frac{\Omega}{4\pi}\int_S L^{k-1}_m(x,\omega_i)p(\omega_o,\omega_i)d\omega_i,$$

where

$$L^{k-1}_m(x,\omega_o) = \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J^{k-1}_{ms}(u,\omega_o)du.$$

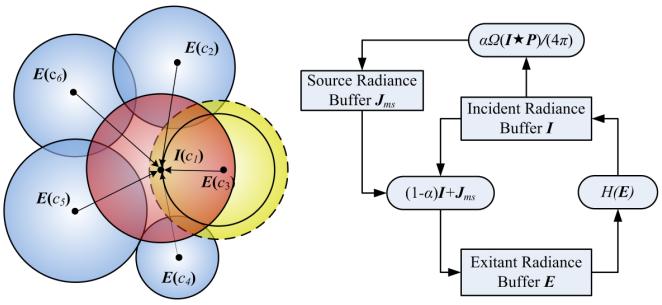
In this recursive computation, we use the single scattering source radiance to initialize the procedure:

$$L^1_m(x,\omega_o) = \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J_{ss}(u,\omega_o)du.$$

This method is similar to that in [Szirmay-Kalos et al. 2005], except that we simulate scattering between RBFs instead of between randomly sampled particles in the volume.

In the SH domain, this scheme proceeds as follows. We first compute at each RBF center the initial radiance distributions from single scattering:

$$\begin{aligned}
&\boldsymbol{I}^1(c_\ell) = \boldsymbol{L}_{in} * \tilde{\boldsymbol{\tau}}(c_\ell)\\
&\tilde{\boldsymbol{J}}^1_{ms}(c_\ell) = \tilde{\boldsymbol{J}}_{ss}(c_\ell) = \tfrac{\Omega}{4\pi}(\boldsymbol{I}^1(c_\ell)\star\boldsymbol{p})\\
&\boldsymbol{E}^1(c_\ell) = \tilde{\boldsymbol{J}}^1_{ms}(c_\ell),
\end{aligned}$$



(a) radiance propagation (b) GPU pipeline

Figure 7: *Multiple scattering simulation. (a) estimation of incident radiance at RBF center c_1. Note that $\theta_{1,3}$ is equal to $\pi/2$ because c_1 is located within RBF B_3. (b) GPU pipeline for multiple scattering simulation.*

where $\boldsymbol{I}$ and $\boldsymbol{E}$ represent incident and exitant radiance, respectively. Then at each iteration k, we update the three SH vectors according to

$$\begin{aligned}
&\boldsymbol{I}^k(c_\ell) = H(\{\boldsymbol{E}^{k-1}(c_h)\})\\
&\tilde{\boldsymbol{J}}^k_{ms}(c_\ell) = \tfrac{\alpha(c_\ell)\Omega}{4\pi}(\boldsymbol{I}^k(c_\ell)\star\boldsymbol{p})\\
&\boldsymbol{E}^k(c_\ell) = (1-\alpha(c_\ell))\boldsymbol{I}^k(c_\ell) + \tilde{\boldsymbol{J}}^k_{ms}(c_\ell).
\end{aligned}$$

Intuitively, we evaluate the incident radiance from the exitant radiance of the preceding iteration using a process H, which will later be explained in detail. Then we compute the scattered source radiance by convolving the incident radiance with the phase function. Here, $\alpha(c_\ell)$ is the opacity of RBF B_ℓ along its diameter (i.e., the scattering probability of B_ℓ), computed as in [Szirmay-Kalos et al. 2005] by $\alpha(c_\ell) = 1 - e^{-3.4r_\ell}$. Finally, we add the scattered source radiance to the transmitted radiance to yield the exitant radiance.

The simulation runs until the magnitude of $\boldsymbol{E}^{k+1}(c_\ell)$ falls below a user-specified threshold, or until a user-specified number of iterations is reached. In our implementation, this process typically converges in 5-10 iterations.

Estimation of Incident Radiance $\boldsymbol{I}^k(c_\ell)$ To estimate the incident radiance at an RBF center c_ℓ, we consider each of the RBFs in its neighborhood as a light source that illuminates c_ℓ, as illustrated in Fig. 7 (a). The illumination from RBF B_h is approximated as that from a uniform spherical light source, whose intensity $E^{k-1}_{\ell,h}$ in direction $c_\ell - c_h$ is reconstructed from the SH vector $\boldsymbol{E}^{k-1}(c_h)$ using Eq. (5):

$$E^{k-1}_{\ell,h} = \boldsymbol{E}^{k-1}(c_h)\cdot\boldsymbol{y}(s(c_\ell,c_h)),$$

where $s(c_\ell,c_h) = \frac{c_\ell - c_h}{\|c_\ell - c_h\|}$ represents the direction from c_h to c_ℓ.

The SH vector for a uniform spherical light source can be represented as a zonal harmonic vector, and tabulated with respect to the angle θ, equal to half the subtended angle by a spherical light source of unit radius as in [Ren et al. 2006]. Unlike β in Section 5.3, θ ranges from 0 to $\pi/2$ such that c_ℓ does not lie within a light source. From this precomputed table, we can retrieve the SH vector using the angle $\theta_{\ell,h}$, which is half the angle subtended by RBF B_h as seen from point c_ℓ:

$$\theta_{\ell,h} = \begin{cases} \arcsin(r_h/\|c_\ell - c_h\|), & \text{if } \|c_\ell - c_h\| > r_h;\\ \pi/2, & \text{otherwise.}\end{cases}$$

We then rotate the vector to direction $c_\ell - c_h$, scale it by $\min(r_h, \|c_\ell - c_h\|)$ to account for RBF radius, and finally multiply it by the intensity $E^{k-1}_{\ell,h}$ to obtain the SH vector $\boldsymbol{I}^k_{\ell,h}$.

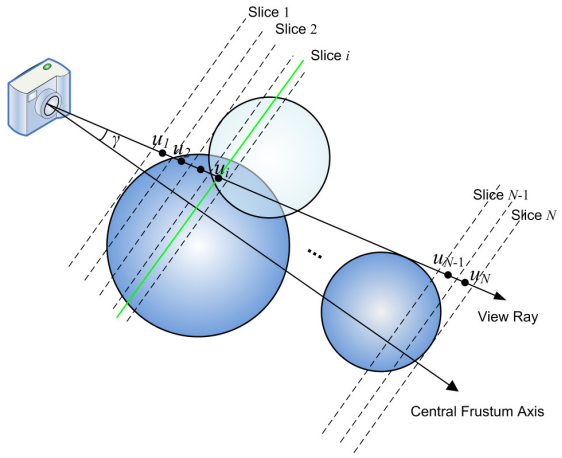


Figure 8: *Ray marching for radiance integration along view rays.*

The incident radiance $\boldsymbol{I}^k(c_\ell)$ at c_ℓ is then computed as:

$$\boldsymbol{I}^k(c_\ell) = \sum_{\|c_\ell - c_h\| < \rho_\ell} \tilde{\tau}(c_\ell, c_h) \boldsymbol{I}^k_{\ell,h},$$

where the parameter ρ_ℓ is used to adjust the neighborhood size. In our current implementation, the default value of ρ_ℓ is $2r_\ell$.

The source radiance for different numbers of scattering events is aggregated to obtain the final source radiance due to multiple scattering: $\tilde{\boldsymbol{J}}_{ms}(c_\ell) = \sum_{k=2} \tilde{\boldsymbol{J}}^k_{ms}(c_\ell)$. Combining this with the single-scattering component of Section 5.3 yields the final source radiance:

$$\tilde{\boldsymbol{J}}(c_\ell) = \tilde{\boldsymbol{J}}_{ss}(c_\ell) + \tilde{\boldsymbol{J}}_{ms}(c_\ell).$$

Comparison The middle row in Fig. 6 compares our multiple scattering result with that from the offline algorithm for volumetric photon mapping [Jensen and Christensen 1998; Fedkiw et al. 2001]. Although our algorithm employs several approximations in the multiple scattering simulation, the two images are comparable. As in [Fedkiw et al. 2001], we use one million photons in computing the volume photon map. A forward ray march is then performed to produce the photon map image.

5.5 Compensated Ray Marching

From the source radiances at the RBF centers, we interpolate the source radiance of each voxel in the volume and composite the radiances along each view ray:

$$\begin{aligned} L(x,\omega_o) &= \tau(x_{\omega_o},x)L_{in}(\omega_o) + \textstyle\int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J(u,\omega_o)du \\ &\approx \tilde{\tau}(x_{\omega_o},x)L_{in}(\omega_o) + \textstyle\int_{x_{\omega_o}}^{x} \tilde{\tau}(u,x)\sigma_t D(u) \tilde{J}(u,\omega_o)du \\ &= \tilde{\tau}(x_{\omega_o},x)L_{in}(\omega_o) + \textstyle\int_{x_{\omega_o}}^{x} \tilde{\tau}(u,x)\sigma_t \tilde{J}_D(u,\omega_o)du \end{aligned} \tag{10}$$

where

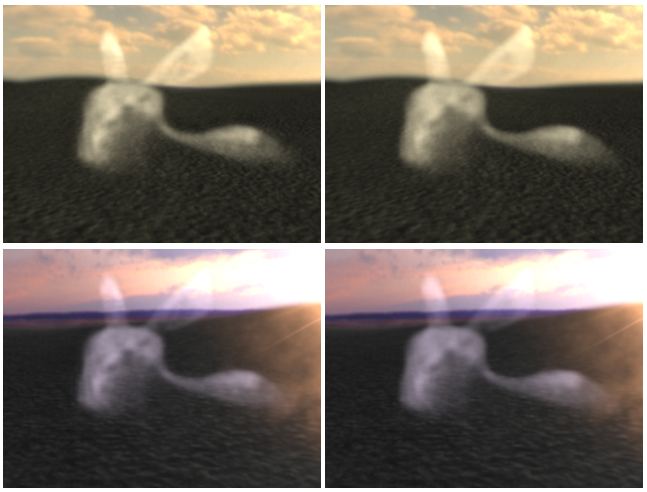
$$\begin{aligned} \tilde{J}_D(u,\omega_o) &= D(u)\tilde{J}(u,\omega_o) \\ &= D(u)\left(\boldsymbol{y}(\omega_o)\cdot \tfrac{1}{D(u)} \textstyle\sum_\ell w_\ell B_\ell(u) \tilde{\boldsymbol{J}}(c_\ell)\right) \\ &= \left(1 + \tfrac{\tilde{R}(u)}{\tilde{D}(u)}\right)\left(\boldsymbol{y}(\omega_o)\cdot \textstyle\sum_\ell w_\ell B_\ell(u) \tilde{\boldsymbol{J}}(c_\ell)\right). \end{aligned} \tag{11}$$

For efficient ray marching, we decompose the RBF volumes into N slices of user-controllable thickness Δu along the current view frustum, as shown in Fig. 8. We calculate the discrete integral of Eq. (10) slice by slice from far to near in a manner similar to [Levoy 1990]:

$$\begin{aligned} L(x,\omega_o) &= L_{in}(\omega_o)\textstyle\prod_{j=1}^{N} \tilde{\tau}_j \; + \\ &\quad \textstyle\sum_{i=1}^{N} \left(\tilde{J}_D(u_i)\sigma_t \Delta u \prod_{j=i+1}^{N} \tilde{\tau}_j\right), \end{aligned}$$

where $\{u_i\}$ contains a point from each slice that lies on the view ray, γ is the angle between the view ray and the central axis of the view frustum, and $\tilde{\tau}_j$ is the transmittance of slice j along the view ray, computed as

$$\tilde{\tau}_j = \exp(-\sigma_t \tilde{D}(u_j)\Delta u / \cos\gamma). \tag{12}$$



(a) our results (b) ray tracing

Figure 9: *Smoke visualization with different lighting conditions. The smoke bunny at dusk (top) has a different appearance from that at dawn (bottom). The compensated ray marching results are similar to the corresponding ray traced images.*

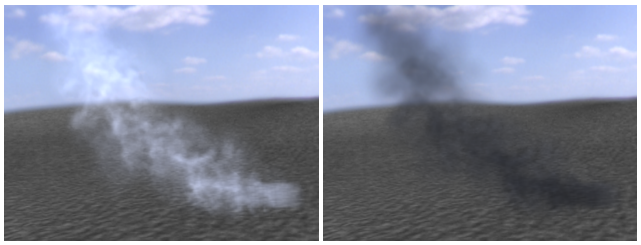
Comparison With compensated ray marching, rendering results are generated with fine details. The bottom row in Fig. 6 compares a rendering result with a ray traced image. In this comparison, ray tracing is performed on the original density field, instead of the approximated density field. The ray tracing result appears slightly smoother than our result, because we account for the residual in the ray march but not in the scattering simulation. The two results are nevertheless comparable. The similarity between our real-time rendering results and offline ray tracing is also exhibited in Fig. 9 for a bunny-shaped smoke volume with different lighting conditions. For a comparison of animation sequences, please view the supplemental video.

5.6 GPU Implementation

All run-time components of our algorithm can be efficiently implemented on the GPU using pixel shaders. In the following, we describe some implementation details.

Single Scattering Source radiances are directly computed at only the RBF centers. For single scattering computation, we rasterize a small 2D quad in which each RBF is represented by a pixel. Since our approximated density model uses at most 1000 RBFs per frame with our implementation settings, a quad size of 32×32 is sufficient. The RBF data (c_ℓ, r_ℓ, w_ℓ) is packed into a texture and passed to the pixel shader.

In the shader, for each pixel (i.e., RBF center) we iterate through the RBFs to compute the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ as described in Section 5.3. Then the source radiance is computed using the SH triple product and convolution according to Eq. (9).



(a) $\sigma_t = 2.46, \Omega = 0.25$ (b) $\sigma_t = 5.64, \Omega = 0.05$

Figure 10: *Changing the optical parameters of smoke.*

Multiple Scattering Fig. 7 (b) depicts an overview of the GPU pipeline for multiple scattering. The incident radiance buffer is initialized with the reduced incident radiance $\boldsymbol{I}^1$ that was calculated for single scattering. Then for each iteration of the simulation, the scattered source radiance $\tilde{\boldsymbol{J}}^k_{ms}$ is accumulated in the source radiance buffer, and the exitant radiance $\boldsymbol{E}^k$ and subsequent incident radiance $\boldsymbol{I}^{k+1}$ are evaluated in alternation. The multiple rendering target and frame buffer object (FBO) of OpenGL extensions are used to avoid frame buffer readbacks.

Note that for the first iteration, the scattered source radiance is not accumulated into $\tilde{\boldsymbol{J}}_{ms}$, and the initial exitant radiance is simply $\tilde{\boldsymbol{J}}^1_{ms}$. As in single scattering, we rasterize a 2D quad of size 32×32 for each operation.

Ray Marching The ray march starts by initializing the final color buffer with the background lighting L_{in}. Then from far to near, each slice i is independently rendered in two passes.

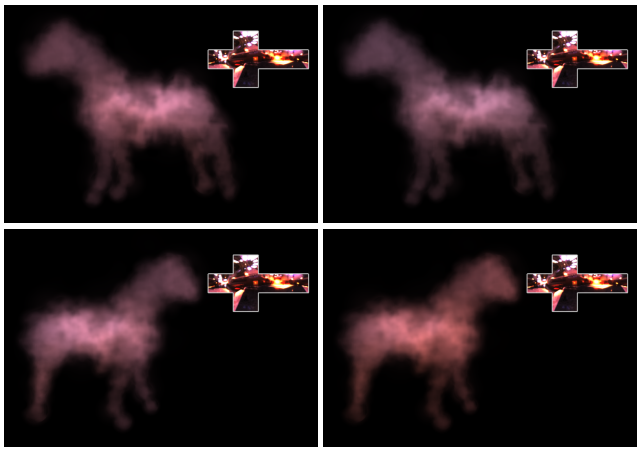
In the first pass, we set the OpenGL blend mode to GL_ONE for both the source and target color buffers, then iterate over all the RBFs. For each RBF B_ℓ, its intersection with the slice is first calculated. This plane-to-sphere intersection can be efficiently computed given the center and radius of the sphere. If an intersection exists, we compute a 2D bounding quad for the circular intersection region, and render this 2D quad. For each pixel in the quad, denote its corresponding point in the volume as u_i. In the pixel shader, the density $w_\ell B_\ell(u_i)$ is evaluated and saved in the alpha channel, and $\tilde{J}_{D,\ell}(u_i) = w_\ell B_\ell(u_i)(\boldsymbol{y}(\omega_o) \cdot \tilde{\boldsymbol{J}}(c_\ell))$ is computed and placed in the RGB channels. With the residual $R(u_i)$ from the hash table, we calculate and store $\tilde{J}_{D,\ell}(u_i)R(u_i)$ in the RGB channels of a second color buffer. After the first pass, we thus have $\sum_\ell \tilde{J}_{D,\ell}(u_i)$, $\tilde{D}(u_i)$, and $R(u_i)\sum_i \tilde{J}_{D,\ell}(u_i)$ in the color buffers, which are sent to the second pass as textures through the FBO.

In the second pass, the OpenGL blend mode is set to GL_ONE and GL_SRC_ALPHA for the source and final color buffers, respectively. Instead of drawing a small quad for each RBF as in the first pass, we draw a large bounding quad for all RBFs that intersect with the slice. In the pixel shader, $\tilde{J}_D(u_i)$ for each pixel is evaluated according to Eq. (11) as

$$\tilde{J}_D(u_i) = \sum_\ell \tilde{J}_{D,\ell}(u_i) + \left(R(u_i) \sum_\ell \tilde{J}_{D,\ell}(u_i) \right) / \tilde{D}(u_i).$$

The RGB channels of the source buffer are then computed as $\tilde{J}_D(u_i)\sigma_t \Delta u / \cos\gamma$, and the alpha channel is set to $\tilde{\tau}_i$, computed using Eq. (12).

The residual $R(u_i)$ is decoded by perfect spatial hashing as described in [Lefebvre and Hoppe 2006]. Eight texture accesses are needed in computing a trilinear interpolation. To avoid divide-by-zero exceptions, residuals are set to zero during preprocessing when $\tilde{D}(u)$ is very small ($< 1.0e-10$).



(a) Constant (b) Henyey-Greenstein

Figure 11: *Changing the phase function of smoke. The lighting, shown as upper-right insets, is fixed. The top and bottom rows are rendered with different viewpoints. The dependence of scattered radiance on the view direction can be seen.*

6 Results and Discussion

We have implemented our algorithm on a 3.7GHz PC with 2GB of memory and an NVidia 8800GTX graphics card. Images are generated at a 640×480 resolution. Please see the supplemental video for live demos. The three sets of smoke animation data used in this paper are all generated by physically-based simulation. The data in Fig. 1 and Fig. 15 are provided by the authors of [Shi and Yu 2005], and the data in Fig. 9 is provided by the authors of [Elcott et al. 2007].

Smoke Visualization As a basic function, our system allows users to visualize smoke simulation results under environment lighting and from different viewpoints. Examples are shown in Fig. 9 and Fig. 15. In Fig. 15, we also compare results with and without residual compensation. With compensated ray marching, we obtain images with fine details.

The optical parameters of the smoke can be edited in real time. Both images of Fig. 10 contain the same smoke density field and lighting. Adjusting the albedo downward and increasing the extinction cross section darkens the appearance of the smoke. In Fig. 11, with fixed lighting we change the phase function from constant to the Henyey-Greenstein (HG) phase function (with eccentricity parameter 0.28). The dependence of the scattered radiance on the view direction can be seen.

Shadow Casting between Smoke and Objects Combined with the spherical harmonic exponentiation (SHEXP) algorithm for soft shadow rendering [Ren et al. 2006], our method can also render dynamic shadows cast between the smoke and scene geometry, as shown in Fig. 12. For this scene containing 36K vertices, we achieve real-time performance at over 20 *fps*.

For each vertex in the scene, we first compute the visibility vector by performing SHEXP over the accumulative log visibility of all blockers. Each RBF of the smoke is regarded as a spherical blocker whose log visibility is the optical depth vector as computed in Section 5.3. Vertex shading is then computed as the triple product of visibility, BRDF and lighting vectors. After the scene geometry is rendered, we compute the source radiance at RBF centers due to single scattering, taking the occlusions due to scene geometry into account using SHEXP. Finally, we run the multiple scattering simulation and compensated ray marching to generate the results.

(a) without shadow casting (b) with shadow casting

Figure 12: *Shadow casting between smoke and scene objects. Top row: the smoke casts a shadow on the terrain. Bottom: the pterosaur casts shadow on the smoke.*

Scene	Fig. 1	Fig. 9	Fig. 11
Volume grid	128^3	100^3	100^3
Frames	600	500	450
Avg. RBFs per frame	460	247	314
RBF approx. RMS error	2.48%	1.03%	1.34%
Decomposition (min)	140	43	85
Hashing (min)	35	18	20
Hash table (MB)	218	57	67
Performance (fps)	$19.1 \sim 95.1$	$36.4 \sim 57.8$	$35.8 \sim 74.8$

Table 2: *Statistics and timings. The total preprocessing time is the sum of the decomposition time and hashing time.*

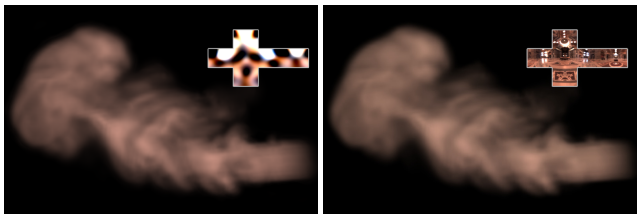
Performance Table 2 lists statistics for the three examples shown in the paper. Note the reasonable preprocessing time, which ranges from 1 to 3 hours. The residual hash tables are significantly smaller than the original density field sequences.

Discussion Although our method currently does not handle all-frequency lighting, it can be reasonably approximated as shown in Fig. 13. On the left is the rendering result from our algorithm using a 4th-order SH approximation of environment lighting. On the right is the corresponding image ray traced using the original environment map.

In our implementation, the user can specify the maximum number of RBFs per frame for density approximation. The default value of 1000 works well for the examples shown in this paper. Fig. 14 exhibits rendering results in which the number of RBFs is fixed at various levels. In principle, the source radiance at each voxel can be exactly reconstructed using a sufficiently large number of RBFs, but clearly a limit on RBFs is needed in practice. This tradeoff between accuracy and performance is an area for future investigation.

7 Conclusion

In this paper, we have presented a method for real-time rendering of smoke animations that allows for interactive manipulation of environment lighting, viewpoint, and smoke attributes. Though a number of techniques have been proposed for efficient rendering of static participating media, they nevertheless require substantial computation or precomputation, making them unsuitable for editing and rendering of dynamic smoke. Based on a presented decomposition of smoke volumes, our method utilizes a low-frequency density field approximation to gain considerable efficiency, while incorporating fine details in a manner that allows for fast processing with high visual fidelity.



(a) 4th-order SH lighting (b) original lighting

Figure 13: *Comparison between the low-frequency lighting used in our implementation and the all-frequency lighting of the original environment map.*

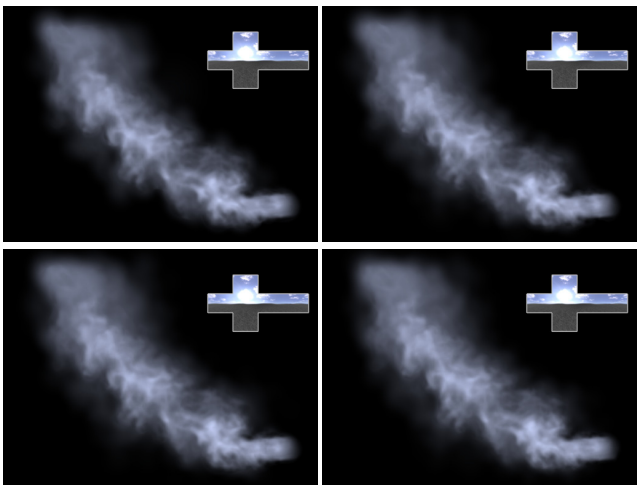


Figure 14: *Rendering results with different numbers of RBFs. Top left: 200. Top right: 400. Bottom left: 600. Bottom right: 800.*

Local light sources, particularly light sources inside the medium, are challenging to process efficiently in our current method. As in [Ren et al. 2006], analytical circular local light sources could be incorporated, but with the cost of evaluating L_{in} (or $\boldsymbol{L}_{in}$) at each RBF center. A sorting problem arises when the local light source enters the smoke volume, since RBFs behind the local light source should not be added into the accumulative optical depth vector. We plan to examine these problems in future work.

Additionally, we are interested in extending our method to handle reflections between an object and the medium. Since our method currently handles shadowing only between the medium and an object outside of it, we would also like to explore solutions for shadowing of objects inside the medium.

Finally, we intend to compare our multiple scattering results with those from the diffusion process described in [Stam 1995], which also simulates multiple scattering among RBFs, given the single scattering result as the boundary condition. This diffusion technique, however, needs to solve a linear system, so would be unsuitable for real-time applications.

Appendix

For the objective function

$$f(\{c_\ell, r_\ell, w_\ell\}_{\ell=1\ldots n}) = \sum_{j,k,l} (D(x_{jkl}) - \tilde{D}(x_{jkl}))^2,$$

the partial derivatives for each parameter are given by

$$\frac{\partial f}{\partial v} = \sum_{j,k,l} 2\left(D(x_{jkl}) - \tilde{D}(x_{jkl})\right)\left(-\frac{\partial \tilde{D}}{\partial v}\right)$$

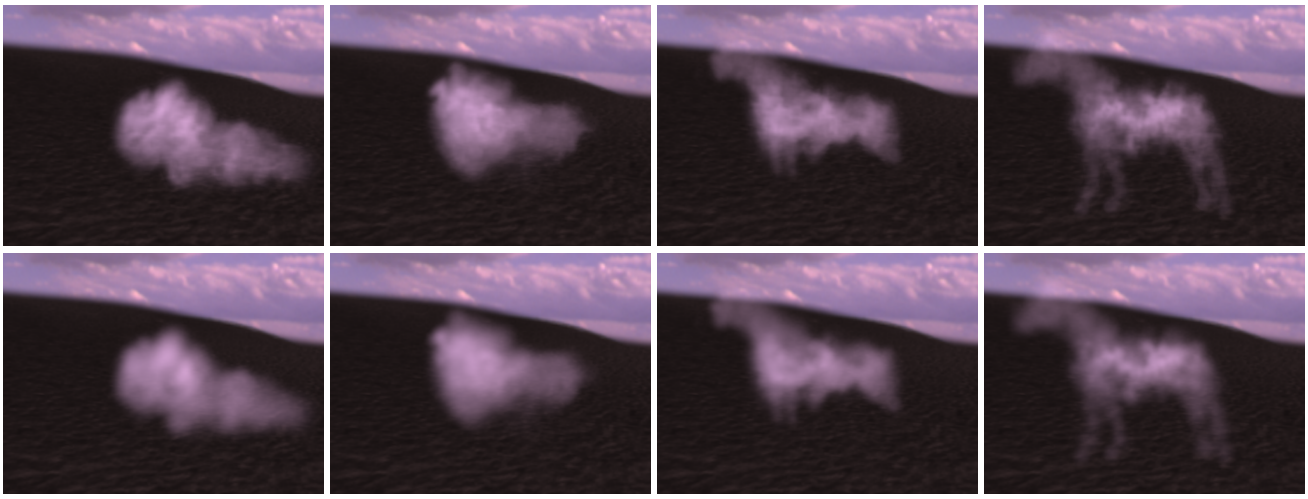


Figure 15: *An initial smoke blob evolves into a smoke horse. Top row: rendering results with residual compensation. Bottom row: results without residual compensation.*

where v denotes a variable in $\{c_\ell, r_\ell, w_\ell\}_{\ell=1\ldots n}$. The partial derivatives of $\tilde{D}$ with respect to each variable are

$$\begin{aligned}
\frac{\partial \tilde{D}}{\partial c_\ell} &= w_\ell \left(\frac{44\Delta_\ell}{9r_\ell^2} - \frac{68\Delta_\ell \|\Delta_\ell\|^2}{9r_\ell^4} + \frac{8\Delta_\ell \|\Delta_\ell\|^4}{3r_\ell^6} \right), \\
\frac{\partial \tilde{D}}{\partial r_\ell} &= w_\ell \left(\frac{44\|\Delta_\ell\|^2}{9r_\ell^3} - \frac{68\|\Delta_\ell\|^4}{9r_\ell^5} + \frac{8\|\Delta_\ell\|^6}{3r_\ell^7} \right), \\
\frac{\partial \tilde{D}}{\partial w_\ell} &= G(\|\Delta_\ell\|, r_\ell),
\end{aligned}$$

where $\Delta_\ell = x_{jkl} - c_\ell$.

References

ANONYMOUS. 2007. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. *Submitted to SIGGRAPH 2007*, Paper ID: 187.

BIRI, V., MICHELIN, S., AND ARQUES, D. 2004. Realtime single scattering with shadows. In *http://citeseer.ist.psu.edu/biri03realtime.html*.

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.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Transactions on Graphics 23*, 3, 905–914.

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 2000*, 19–28.

ELCOTT, S., TONG, Y., KANSO, E., SCHR¨DER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transaction on Graphics*, To Appear.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001*, 15–22.

GEIST, R., RASCHE, K., WESTALL, J., AND SCHALKOFF, R. J. 2004. Lattice-boltzmann lighting. In *Rendering Techniques*, 355–362.

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

HEGEMAN, K., ASHIKHMIN, M., AND PREMOZE, S. 2005. A lighting model for general participating media. In *Symposium on Interactive 3D graphics and games*, 117–124.

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.

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

LEFEBVRE, S., AND HOPPE, H. 2006. Perfect spatial hashing. *ACM Transactions on Graphics 25*, 3, 579–588.

LEVOY, M. 1990. Efficient ray tracing of volume data. *ACM Transactions on Graphics 9*, 3, 245–261.

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

PREMOZE, S., ASHIKHMIN, M., RAMAMOORTHI, R., AND NAYAR, S. 2004. Practical rendering of multiple scattering effects in participating media. In *Eurographics Symposium on Rendering*, 363–374.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Transactions on Graphics 25*, 3, 977–986.

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.

RUSHMEIER, H. E. 1988. *Realistic image synthesis for scenes with relatively participating media*. PhD thesis, Cornell University.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM Transaction on Graphics 24*, 1, 140–164.

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

SLOAN, P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics 24*, 3, 1216–1224.

STAM, J. 1994. Shocastic rendering of density fields. In *Graphics Interface*, 51–58.

STAM, J. 1995. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop*, 41–50.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S., AND NAYAR, S. 2005. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics 24*, 3, 1040–1049.

SZIRMAY-KALOS, L., SBERT, M., AND UMMENHOFFER, T. 2005. Real-time multiple scattering in participating media with illumination networks. In *Rendering Techniques*, 277–282.

TSAI, Y.-T., AND SHIH, Z.-C. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *Proceedings of SIGGRAPH 2006*, 967–976.

WYVILL, G., AND TROTMAN, A. 1990. Ray-tracing soft objects. In *CG International '90*, 469–476.

ZHU, C., BYRD, R. H., LU, P., AND NOCEDAL, J. 1997. Algorithm 778. L-BFGS-B: Fortran subroutines for Large-Scale bound constrained optimization. *ACM Transactions on Mathematical Software 23*, 4 (Dec.), 550–560.