Real-Time Smoke Rendering Using Compensated Ray Marching

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

*Microsoft Research Asia †State Key Lab of CAD&CG, 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. To expedite rendering, the source radiance distribution within the smoke is computed from only the low-frequency RBF approximation of the density fields, since the high-frequency residuals have little impact on global illumination under low-frequency environment lighting. Furthermore, in computing source radiances the contributions from single and multiple scattering are evaluated at only the RBF centers and then approximated at other points in the volume using an RBF-based interpolation. A slice-based integration of these source radiances along each view ray is then performed to render the final image. The high-frequency residual fields, which are a critical component in the local appearance of smoke, are compensated back into the radiance integral during this ray march 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. Only moderate preprocessing time and storage is needed. This approach provides the first method for real-time smoke rendering that includes single and multiple scattering while generating results comparable in quality to offline 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.

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

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 von 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 view ray. To facilitate 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 throughout the volume can be rapidly computed with this RBF model by directly evaluating the radiances at only the RBF centers and then approximating the radiances at other points using an RBF-based interpolation. For fast computation of radiances at RBF centers, we aggregate the optical depths due to each RBF, and then employ an adaptation of the spherical harmonic exponentiation (SHEXP) technique [Ren et al. 2006] to evaluate single scattering and initialize an iterative diffusion equation solver for multiple scattering. 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. Furthermore, this speedup comes at only a minor loss in accuracy, since the high-frequency residuals of the density field have little effect on the global distribution of source radiances for smoke viewed under low-frequency environment lighting.

The appearance of fine-scale smoke details such as vortices, however, depends heavily on the high-frequency density components contained in the residuals. To incorporate these smoke details in an efficient manner, we compensate for the residuals by accounting for their extinction effects (or enlightenment effects for negative residuals) in the radiance integration along each view ray. Since residual fields have significant values only at sparse locations, we take advantage of the perfect spatial hashing technique [Lefebvre and Hoppe 2006] 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 provides 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, this approach offers the first method for real-time rendering of smoke with fidelity comparable to offline algorithms like ray tracing, as shown in Fig. 6, Fig. 12 and the supplemental video. With this technique, interactive modifications of scattering properties become 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. Zhou *et al.* [2007] introduced an analytic model for rendering single scattering in smooth, optically thin, inhomogeneous media which is modeled using radial basis functions. Due to its inhomogeneity, fine-scale details 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] trace a relatively small number of rays, and estimate source radiance at a point by querying the volume photon map. Jarosz *et al.* [2007] proposed a method to cache radiance at sparse locations and interpolate using gradients of the radiative transport equation. While these strategies for sampling reduction can significantly decrease computation, considerable simulation time is still needed to render a single image, making this approach unsuitable 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 von Herzen [1984] computed the source radiance of voxels in a spherical harmonics (SH) basis whose coefficients are computed from a system of partial differential equations (PDEs), and then integrated these radiances along view rays. Rushmeier [1987; 1988] presented zonal finite element methods for isotropic scattering in participating media. Stam [1995] introduced diffusion theory to compute energy transport in the medium using either a multi-grid scheme or a finite-element blob method. Detailed smoke shape in an evolving smoke volume can be accounted for using blob warping [Stam and Fiume 1995]. 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.

Volume rendering has long been applied in rendering gaseous phenomena [Ebert and Parent 1990], and hardware accelerated techniques have been proposed to render single scattering [Kniss et al. 2003] or multiple scattering [Kniss et al. 2003; Riley et al. 2004] in participating media at interactive frame rates. Targeting specific phenomena such as sky or clouds, Riley *et al.* [2004] tabulated the multiple scattering angular distribution function. In contrast, we wish to enable changes in extinction, scattering albedo and even phase function at runtime, making such tabulation impossible.

Precomputation Techniques Efficient rendering of participating media can also be achieved through precomputation of various scene-dependent quantities [Harris and Lastra 2001; Sloan et al. 2002; Premoze et al. 2004; Hegeman et al. 2005; Szirmay-Kalos et al. 2005]. 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. Our method also needs a preprocess to decompose the smoke density into a set of RBFs and a residual. However, such decomposition can be done in moderate time and is independent of lighting and smoke attributes such as scattering albedo and phase function.

Ren *et al.* [2006] presented a GPU-based algorithm to render soft shadows in dynamic scenes with low-frequency environment lighting. Blocker geometries are approximated by sets of spheres, whose spherical harmonic (SH) visibility is accumulated in log space and exponentiated by SHEXP to obtain the final visibility SH vector. We present a framework for real-time rendering of smoke that takes advantage of the SHEXP technique for rapid evaluation of transmittance vectors in our single scattering computation.

3 Light Transport in Scattering Media

In our work, we represent lighting as a low-frequency environment map L_{in}, described by a vector of spherical harmonic coefficients $\boldsymbol{L}_{in}$. The input smoke animation is modeled as a sequence of volumetric density fields. We denote the smoke density at each frame as $D(x)$, the extinction cross section of the smoke as σ_t, the scattering cross section as σ_s, and the single-scattering albedo as Ω. In the following, we briefly review light transport in scattering media. The radiance quantities that we use are listed in Table 1 and are defined as in [Cerezo et al. 2005].

Fig. 2 illustrates the transport of light through smoke. The radiance at a point x along direction ω_o is composed of reduced incident radiance L_d and media radiance L_m:

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

Reduced incident radiance describes source illumination L_{in} that arrives directly at x along direction ω_o with some attenuation by the medium:

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

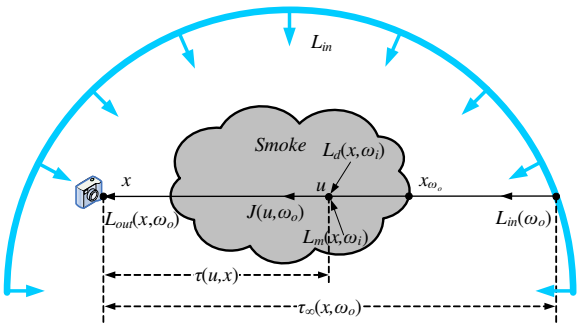


Figure 2: *Light transport in smoke.*

where $\tau_\infty(x,\omega_o)$ is the transmittance from infinity to x along direction ω_o, computed as $\exp(-\int_x^{\infty_\omega}\sigma_t D(v)dv)$.

On the other hand, media radiance L_m consists of light that has scattered at least once in the medium before arriving at x along direction ω_o. It is computed by integrating radiance contributions along the corresponding view ray:

$$L_m(x,\omega_o) = \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J(u,\omega_o)du, \qquad (2)$$

where source radiance $J(u,\omega_o)$ represents light that scatters at a point u toward direction ω_o. In non-emissive media such as smoke, this source radiance is composed of a single scattering J_{ss} and a 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 reduced incident radiance whose first scattering interaction in the medium occurs at u:

$$J_{ss}(u,\omega_o) = \frac{\Omega}{4\pi}\int_S L_d(u,\omega_i)p(\omega_o,\omega_i)d\omega_i, \qquad (3)$$

where the phase function $p(\omega_o,\omega_i)$ describes the directional scattering distribution of the smoke. In computer graphics, the phase function is often assumed to be symmetric about the incident direction, and can be parameterized by the angle between the incident and outgoing direction as $p(\omega_o\cdot\omega_i)$. The first moment of the phase function $\bar{\mu} = 3/2\int_{-1}^{1}\mu p(\mu)d\mu$ provides a simple measure of a medium's anisotropy. With this first moment, we define the transport cross section σ_{tr} as $(1-\Omega\bar{\mu}/3)\sigma_t$.

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

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

In an optically thick medium, media radiance which has undergone multiple scattering can be approximated by a diffusion process [Stam 1995]. Expressing the media radiance by a two-term Taylor expansion gives us

$$L_m(x,\omega) = L_m^0(x) + \boldsymbol{L}_m^1(x)\cdot\omega,$$

where $L_m^0(x) = \frac{1}{4\pi}\int_S L_m(x,\omega)d\omega$ is the average radiance over all angles, and $\boldsymbol{L}_m^1(x) = \frac{3}{4\pi}\int_S L_m(x,\omega)\omega d\omega$ is the principal directional component.

The average radiance $L_m^0(x)$ is determined by a diffusion equation:

$$\nabla\kappa(x)\cdot\nabla L_m^0(x) + \kappa(x)\nabla^2 L_m^0(x) - \alpha(x)L_m^0(x) + S(x) = 0, \qquad (5)$$

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
$\tilde{D}(x)$	RBF approximation of smoke density
$R(x)$	Residual of 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
$L_d(x,\omega)$	Reduced incident radiance
$L_m(x,\omega)$	Media radiance
$J(x,\omega)$	Source radiance
$p(\omega_o,\omega_i)$	Phase function (normalized)
$\boldsymbol{y}(s)$	Set of SH basis functions
$y_i(s)$, $y_l^m(s)$	SH basis function
$\tilde{f}(s)$	Projection of $f(s)$ on the SH basis
$\boldsymbol{f}_i$, $\boldsymbol{f}_l^m$	Spherical harmonic coefficient vector

Table 1: *Notation*

where the following shorthand notations have been used:

$$\begin{aligned}\kappa(x) &= (3\sigma_{tr}D(x))^{-1},\\ \alpha(x) &= (\sigma_t-\sigma_s)D(x),\\ S(x) &= \tfrac{\sigma_t}{\Omega}D(x)J_{ss}^0(x) - \tfrac{\sigma_t}{\sigma_{tr}}\nabla\cdot\boldsymbol{J}_{ss}^1(x).\end{aligned}$$

J_{ss}^0, $\boldsymbol{J}_{ss}^1$ are the first two terms of the Taylor expansion of single scattering source radiance $J_{ss}(x,\omega)$.

The directional component $\boldsymbol{L}_m^1(x)$ can be expressed in terms of $L_m^0(x)$ as

$$\boldsymbol{L}_m^1(x) = \kappa(x)(-\nabla L_m^0(x) + \sigma_t D(x)\boldsymbol{J}_{ss}^1(x)). \qquad (6)$$

4 Algorithm Overview

In our algorithm, we utilize a basic assumption of low-frequency environment lighting. Based on this lighting condition, we can reasonably assume that source radiance varies smoothly in the smoke volume, so that its angular variation can be approximated by low order spherical harmonics and that its spatial variation can be well-modeled by a relatively small number ($300 \sim 1200$) of radial basis functions. We also assume the medium to be optically thick, so that multiple scattering can be approximated by a diffusion process [Stam 1995].

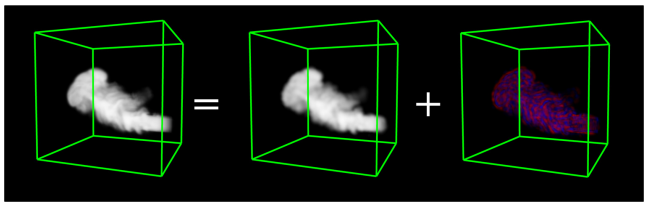
Our approach consists of a preprocessing step and a runtime rendering algorithm.

Preprocessing As shown in Fig. 3, the density field D is decomposed into a weighted sum of RBFs B_ℓ and a residual R:

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

Each RBF B_ℓ is defined by its center c_ℓ and radius r_ℓ:

$$B_\ell(x) = G(\|x-c_\ell\|, r_\ell) = \exp\left(-(\|x-c_\ell\|/r_\ell)^2\right).$$



(a) original data (b) RBF approx. (c) residual (16×)

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

In our implementation, we limit the kernel support to a radius of $3r_\ell$, which reduces both preprocessing and runtime computation.

With this decomposition of the density field, we use the low-frequency RBF representation to efficiently compute global illumination within the medium, and account for the high-frequency residual in rendering local appearance details. The decomposition is computed for the entire animation sequence, as will be discussed in Sec. 5.1. An analysis on rendering quality with respect to the number of RBFs will be presented in Sec. 6.

Due to its size, an original density field D cannot in general be effectively processed on the GPU. However, this decomposition into a low-frequency RBF approximation $\tilde{D}$ and a residual field R leads to a considerable reduction in data size. Further memory savings can be obtained by taking advantage of the relatively small number of significant values that typically exists in a residual field. To promote compression, we obtain a sparse residual field by quantizing the residual values. This sparse residual, we observe, can be highly compressed using perfect spatial hashing [Lefebvre and Hoppe 2006], as will be discussed in Sec. 5.2. In addition to providing manageable storage costs, perfect spatial hashing offers fast reconstruction of the residual at runtime.

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

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

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

$$J(x, \omega_o) \approx \tilde{J}(x, \omega_o) = \frac{1}{\tilde{D}(x)} \sum_\ell w_\ell B_\ell(x) \tilde{J}(c_\ell, \omega_o). \qquad (7)$$

This low-frequency representation of source radiances can be rapidly computed and provides a reasonable approximation with low-frequency environment illumination. An example of single scattering and multiple scattering source radiances in a smoke volume is illustrated in Fig. 4 (a) and (b). Details of the source radiance computation will be presented in Sec. 5.3 and 5.4.

After computing source radiances, we obtain the media radiance $L_m(x, \omega_o)$ by performing a ray march that gathers 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

(a) single scattering (b) multi. scattering (c) comp. ray march.

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. The density ranges from* 0.0 *to* 1.1*,* $\sigma_t = 3.03$*,* $\Omega = 0.73$*, and the smoke volume is a cube of about* 3.5 *meters in length.*

compensates for the density field residual:

$$\begin{aligned} L_m(x, \omega_o) &\approx \tilde{L}_m(x, \omega_o) + \tilde{C}_m(x, \omega_o) \\ &= \sum_{j=1}^{N} \tilde{\tau}(x_j, x) \sigma_t \tilde{D}(x_j) \tilde{J}(x_j, \omega_o) + \\ &\quad \sum_{j=1}^{N} \tilde{\tau}(x_j, x) \sigma_t R(x_j) \tilde{J}(x_j, \omega_o), \end{aligned} \qquad (8)$$

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 or enlightenment effects of the density residuals, as shown in Fig. 4 (c). By incorporating high-frequency smoke details in this manner, high-quality smoke renderings can be generated. In contrast to the work of Schpok *et al.* [2003] which adds procedural 3D noise to enrich coarse lighting in generated clouds, the high-frequency residuals used in our compensated ray march represent true smoke features, whose fine distinctive details and motion are difficult to convincingly approximate with noise animation.

In this formulation, we obtain real-time performance with high visual fidelity by accounting for the residual field only where it has a direct impact on Eq. (8), 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. (8), 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 Sec. 5.

5 Algorithm Components

5.1 Density Field Approximation

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

$$\min_{c_\ell, r_\ell, w_\ell} \left(\sum_{j,k,m} \left[D(x_{jkm}) - \sum_{\ell=1}^{n} w_\ell B_\ell(x_{jkm}) \right]^2 \right), \qquad (9)$$

where (j, k, m) indexes a volumetric grid point at position x_{jkm}. For this minimization, we employ the L-BFGS-B minimizer [Zhu et al. 1997], which has been used in [Sloan et al. 2005] to fit zonal harmonics to a radiance transfer function.

L-BFGS-B is a derivative-based method, so at each iteration we provide the minimizer with the objective function and the partial derivatives for each variable. We bound the RBF radii to within $0.015 \sim 0.09$ (with volume size normalized to 1) and the weights to $0.01 \sim 1.0$ times the largest density value. 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 RBF with the most insignificant integration value 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 non-zero data. The algorithm terminates when both teleportation strategies fail to reduce the objective function. The inclusion of teleportation into the minimization process often leads to a further reduction of the objective function by $20\% \sim 30\%$.

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. This increases convergence speed and improves performance. However, a large number of teleportations may possibly reduce temporal coherence and cause flickering in the final animation. We avoid this problem by employing an adaptive insertion scheme, which begins with the aforementioned teleportation and minimization procedure on the initial frame. The relative error of this first frame is then recorded as

$$\sum_{j,k,m} \left[D(x_{jkm}) - \sum_{\ell=1}^{n} w_\ell B_\ell(x_{jkm}) \right]^2 / \sum_{j,k,m} D(x_{jkm})^2.$$

At each of the subsequent frames in the sequence, we first delete a user-specified number of the most insignificant RBFs, then perform L-BFGS-B minimization without teleportations. The converged relative error is compared to that of the first frame, and if their ratio lies above a given threshold, which is $1.05 \sim 1.2$ for all data used in this paper, we insert an RBF of fixed radius at the location of maximum error and minimize again. Additional RBFs are inserted in this manner until the relative error ratio is driven below the threshold or a specified maximum number of RBFs have been inserted. In our current implementation, 1000 RBFs are used in the initial frame, and in each subsequent frame, 3 RBFs are removed and a maximum of 8 RBFs are inserted.

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 quantitize $R(x)$ to further reduce storage. For all the data used in our paper, we found 8-bit quantization to be sufficient for visually plausible results. Then, we 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 avoid the precision problems that arise in decoding the domain coordinates of large packed volumes, and facilitate loading of the hash tables. In our implementation, we tile $3^3 = 27$ frames per volume.

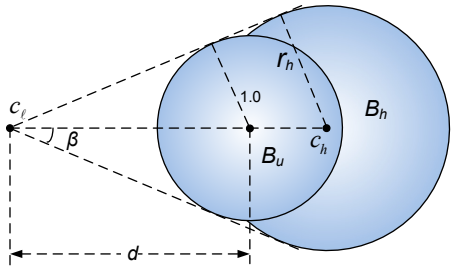


Figure 5: *The optical depth of an RBF B_h is determined by the angle β and its absolute radius r_h. For fast evaluation, we tabulate in a preprocessing step a set of zonal harmonics vectors $\boldsymbol{T}(\beta)$ that represent optical depth for unit-radius RBFs B_u. At runtime, the corresponding precomputed ZH vector is retrieved, rotated, and then multiplied by r_h to obtain the optical depth of B_h.*

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. (3):

$$\begin{aligned} \tilde{J}_{ss}(c_\ell, \omega_o) &= \tfrac{\Omega}{4\pi} \textstyle\int_S \tilde{L}_d(c_\ell, \omega_i) p(\omega_o, \omega_i) d\omega_i \\ &= \tfrac{\Omega}{4\pi} \textstyle\int_S L_{in}(\omega_i) \tilde{\tau}_\infty(c_\ell, \omega_i) p(\omega_o, \omega_i) d\omega_i, \end{aligned} \tag{10}$$

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. (10) can be efficiently computed in the spherical harmonics domain using the SH triple product and convolution operators:

$$\tilde{\boldsymbol{J}}_{ss}(c_\ell) = \frac{\Omega}{4\pi} \left[(\boldsymbol{L}_{in} * \tilde{\boldsymbol{\tau}}(c_\ell)) \star \boldsymbol{p} \right]. \tag{11}$$

The SH projections $\boldsymbol{L}_{in}$ and $\boldsymbol{p}$ are each computed once and reused until the lighting or phase function is modified by the user. For a review of spherical harmonic operations, we refer readers to the Appendix. In the following, we describe how to efficiently compute the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ on the fly.

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

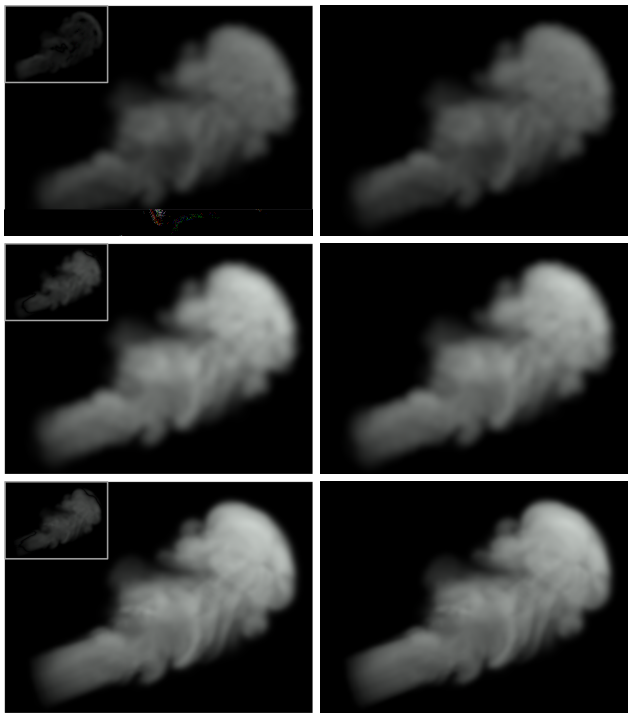
$$\begin{aligned} \tilde{\tau}(c_\ell, \omega_i) &= \exp\left(-\sigma_t \textstyle\sum_h w_h \int_{c_\ell}^{\infty_{\omega_i}} B_h(u) du \right) \\ &= \exp\left(-\sigma_t \textstyle\sum_h w_h T_h(c_\ell, \omega_i) \right), \end{aligned}$$

where $T_h(c_\ell, \omega_i)$ is the optical depth through RBF B_h along the path described by c_ℓ and ω_i. We can project $\tilde{\tau}(c_\ell, \omega_i)$ to the SH domain to obtain $\tilde{\boldsymbol{\tau}}(c_\ell)$:

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

where $\boldsymbol{T}_h(c_\ell)$ is the SH projection of $T_h(c_\ell, \omega_i)$ and $\exp_*$ is the SH exponentiation operator described in the Appendix.

For efficient computation of $\boldsymbol{T}_h(c_\ell)$, we utilize tabulated optical depth vectors for unit-radius RBFs at various distances. As illustrated in Fig. 5, the optical depths through RBF B_h can be simply



(a) our result (b) ray tracing

Figure 6: *Comparison between our results and ray tracing. Top: single scattering in an approximated density field (root mean square (RMS) error:4.3%). Middle: single and multiple scattering in the approximated density field (RMS error:13.8%). Bottom: final results for the original density field (RMS error:14.1%). Lighting comes mainly from the top-right corner. The ray tracing result takes 47 minutes to compute, while ours renders at real-time rates. The smoke density ranges from* 0.0 *to* 1.1*,* $\sigma_t = 2.49$*,* $\Omega = 0.66$*, and the smoke volume width is about* 3.0 *meters. Absolute differences are included as insets.*

computed as r_h times that through a unit-radius RBF B_u with the same subtended half-angle β with respect to c_ℓ. So we build a table of optical depth vectors for unit-radius RBFs of various angles β:

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

where d is the distance from c_ℓ to the center of B_u, and instances where c_ℓ exists within B_u are handled by extending the definition of β to a smoothly varying, monotonic quantity in the range $[0, \pi]$. Since the RBF kernel function is symmetric, the construction of optical depth vectors is equivalent to a 1D tabulation of zonal harmonic (ZH) vectors [Sloan et al. 2005]. In tabulation, we obtain satisfactory results by uniformly sampling β at 256 values in $[0, \pi]$.

To compute $\boldsymbol{T}_h(c_\ell)$ at runtime, we first retrieve from the table the corresponding ZH vector $\boldsymbol{T}(\beta_{\ell,h})$, where $\beta_{\ell,h}$ is the angle β evaluated at c_ℓ with respect to RBF B_h. This ZH vector $\boldsymbol{T}(\beta_{\ell,h})$ is then rotated to the axis determined by c_ℓ and c_h, and multiplied by the radius r_h to obtain $\boldsymbol{T}_h(c_\ell)$. We note that this computation is analogous to that in [Ren et al. 2006], except that here we are dealing with optical depth instead of log space visibility.

Computation of the transmittance vector $\tilde{\boldsymbol{\tau}}(c_\ell)$ is then straightforward. For each RBF center c_ℓ, we iterate through the set of RBFs. Their 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. (11).

Comparison This single scattering approximation yields results similar to those obtained from a volume ray tracer [Kajiya and von Herzen 1984], as shown in the top row of Fig. 6. The error is more significant at density boundaries, where the distance to the samples is large and interpolation error rises. For a clearer comparison, ray tracing is performed with the approximated density field $\tilde{D}(x)$. In rendering the single scattering image, the ray marching algorithm described in Sec. 5.5 is used without accounting for the residual.

5.4 Multiple Scattering

Given the single scattering source radiance, the multiple scattering media radiance is obtained by solving the diffusion equations in Eq. (5) and Eq. (6). As in [Stam 1995], the diffusion equation is rewritten in terms of the RBF representation and constructed at the RBF centers. A linear system is formed, and solved through a least squares optimization to yield the media radiances at the RBF centers.

The solution of the diffusion equation can be trivially converted into order-2 SH media radiance vectors, and then convolved with the phase function to yield the multiple scattering source radiance $\tilde{\boldsymbol{J}}_{ms}(c_\ell)$. The final source radiance is computed as

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

Linear System Construction Like the source radiance, the average media radiance $L^0_m(x)$ can be approximated with RBF interpolation:

$$L^0_m(x) \quad \approx \quad \tfrac{1}{\tilde{D}(x)} \textstyle\sum_\ell w_\ell B_\ell(x) L^0_m(c_\ell). \tag{12}$$

Substituting the RBF representation of L^0_m, J^0_{ss}, and $\boldsymbol{J}^1_{ss}$ into Eq. (5), and approximating $D(x)$ by $\tilde{D}(x)$ gives a linear equation of the unknowns $L^0_m(c_\ell)$:

$$\begin{aligned} &\textstyle\sum_\ell w_\ell \left(\frac{\tilde{D}^2\nabla^2 B_\ell - 3\tilde{D}\nabla\tilde{D}\cdot\nabla B_\ell + \left(3\nabla\tilde{D}\cdot\nabla\tilde{D} - \tilde{D}\nabla^2\tilde{D}\right)B_\ell}{3\sigma_{tr}\tilde{D}^4} - \sigma_a B_\ell \right) L^0_m(c_\ell) \\ &= \textstyle\sum_\ell w_\ell \left(\frac{\sigma_t(\nabla B_\ell \tilde{D} - B_\ell \nabla\tilde{D})}{\sigma_{tr}\tilde{D}^2} \cdot \boldsymbol{J}^1_{ss}(c_\ell) - \frac{\sigma_t}{\Omega} B_\ell J^0_{ss}(c_\ell) \right). \end{aligned} \tag{13}$$

For simplicity, we have omitted the variable x in the above equation. The values of $B_\ell(x), \nabla B_\ell(x)$ and $\nabla^2 B_\ell(x)$ can be analytically computed. Then $\tilde{D}(x), \nabla\tilde{D}(x), \nabla^2\tilde{D}(x)$ can be calculated according to the RBF representation. We also note that the two-term Taylor expansion $(J^0_{ss}, \boldsymbol{J}^1_{ss})$ of single-scattering source radiances can be trivially converted to and from order-2 SH vectors.

Enforcing Eq. (13) at all RBF centers $c_h (h = 1 \ldots n)$ (i.e., replacing x with c_ℓ) results in a linear system

$$\mathsf{A}\boldsymbol{L}^0_m = \boldsymbol{b},$$

where A is an $n \times n$ matrix, and $\boldsymbol{L}^0_m = \{L^0_m(c_\ell)\}$. Solving it gives the average media radiance $L^0_m(c_\ell)$ at RBF centers. The directional component $\boldsymbol{L}^1_m(c_\ell)$ is then obtained by substituting Eq. (12) into Eq. (6).

Conjugate Gradient Solver The above linear system can be solved through a least squares optimization: $\mathsf{A}^T\mathsf{A}\boldsymbol{L}^0_m = \mathsf{A}^T\boldsymbol{b}$, which can be done using a conjugate gradient (CG) solver. The solution is iteratively updated until a user-defined error threshold is reached.

A useful observation is that the solution of the equation is often temporally coherent as the user rotates the lighting or adjusts the smoke attributes or as a smoke animation proceeds to the next frame. In these typical cases, the solution of the previous frame is a good starting point for the iterations of the current frame. When the user alters scene or smoke attributes, the RBF set remains unchanged, and the solution of the previous frame can be directly used to initialize the CG solver. For temporal progressions, we reconstruct the initial solution at the current RBF centers by an RBF-based interpolation of the solution for the previous frame. This simple optimization significantly improves convergence speed, and 20∼40 iterations are usually sufficient for a visually plausible result. In cases when no temporal coherence can be exploited, e.g., when the user switches the environment from one to another, it is necessary to start the solver from a default initialization.

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]. The diffusion approximation generates errors at the RBF centers, which are then propagated into the volume. Nonetheless, the two images are comparable. As in [Fedkiw et al. 2001], we use one million photons in computing the volume photon map for the comparison result. A forward ray march is then performed to produce the photon map image. Fig. 12 shows another example of multiple scattering.

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:

$$\begin{aligned} L(x,\omega_o) &= \tau(x_{\omega_o},x)L_{in}(\omega_o) + \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t D(u) J(u,\omega_o)du \\ &\approx \tau(x_{\omega_o},x)L_{in}(\omega_o) + \int_{x_{\omega_o}}^{x} \tau(u,x)\sigma_t \tilde{J}_D(u,\omega_o)du, \end{aligned} \tag{14}$$

where

$$\begin{aligned} \tilde{J}_D(u,\omega_o) &= D(u)\tilde{J}(u,\omega_o) \\ &= D(u)\left(\boldsymbol{y}(\omega_o)\cdot\tfrac{1}{\tilde{D}(u)}\textstyle\sum_\ell w_\ell B_\ell(u)\tilde{\boldsymbol{J}}(c_\ell)\right) \\ &= \left(1+\tfrac{R(u)}{\tilde{D}(u)}\right)\left(\textstyle\sum_\ell w_\ell B_\ell(u)(\boldsymbol{y}(\omega_o)\cdot\tilde{\boldsymbol{J}}(c_\ell))\right). \end{aligned} \tag{15}$$

We compute $L(x,\omega_o)$ with two passes: a volume pass to obtain the density field $D(u)$ and the approximate source radiance field $\tilde{J}(u,\omega_o)$, and a view pass to composite the radiance.

The volume pass renders at volume resolution. It iterates over each slice of the volume to reconstruct the density field $\tilde{D}(u)$. The residual $R(u)$ is obtained from the hash table, and $D(u)$ is computed as $\tilde{D}(u)+R(u)$. At the same time, the approximate source radiance field $\tilde{J}(u,\omega_o)$ is computed, and then multiplied by $D(u)$ to yield $\tilde{J}_D(u,\omega_o)$.

The view pass then renders at screen resolution. The RBF volume is divided along the current view direction into N slices of user-controllable thickness Δu. We iterate through these slices to retrieve both $D(u)$ and $\tilde{J}_D(u,\omega_o)$ output by the volume pass, and composite the final radiance of each screen pixel, as depicted in Fig. 7. For all the results in the paper, Δu is set to half of a voxel length.

More concretely, we calculate the discrete integral of Eq. (14) slice by slice from far to near in a manner similar to [Levoy 1990]:

$$L(x,\omega_o) = L_{in}(\omega_o)\prod_{j=1}^{N}\Delta\tau_j + \sum_{i=1}^{N}\left(\tilde{J}_D(u_i)\sigma_t\Delta u\prod_{j=i+1}^{N}\Delta\tau_j\right).$$

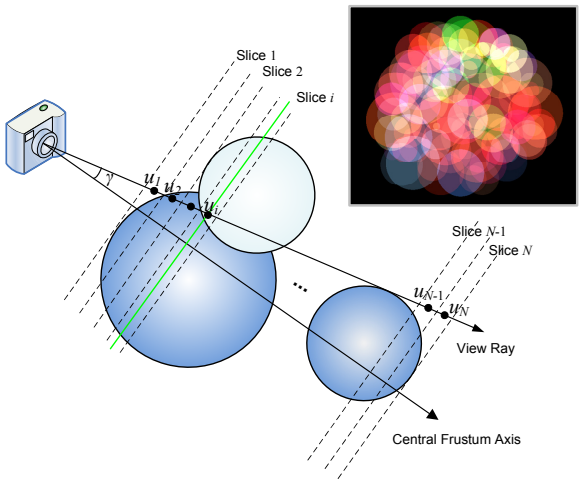


Figure 7: *A 2D illustration of the view pass in ray marching. At the top right corner is a typical slice taken from the data of Fig. 3.*

Here, $\{u_i\}$ contains a point from each slice that lies on the view ray, and $\Delta\tau_j$ is the transmittance of slice j along the view ray, computed as

$$\Delta\tau_j = \exp(-\sigma_t D(u_j)\Delta u/\cos\gamma), \tag{16}$$

where γ is the angle between the view ray and the central axis of the view frustum. Details of the GPU implementation will be discussed in the following subsection.

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 two results are seen to be comparable. In comparing with the middle row, the additional error caused by compensated ray marching is small. For a comparison of animation sequences, please view the supplemental video.

5.6 GPU Implementation

All runtime components of our algorithm can be efficiently implemented on the GPU. For our GPU implementation, a good tradeoff between performance and quality is obtained using order-4 SHs. 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 to trigger a pixel shader, in which the RBF information is retrieved from a texture and used to query the optical depth ZH vector table $\boldsymbol{T}(\beta)$ described in Sec. 5.3. The resulting ZH vectors are then rotated, scaled, accumulated and exponentiated in a manner identical to [Ren et al. 2006]. Finally, the source radiance SH vector $\tilde{\boldsymbol{J}}_{ss}$ is computed using the SH triple product and convolution according to Eq. (11), and rendered into four textures using the OpenGL extensions for multiple render targets and frame buffer objects.

Multiple Scattering We implement the multiple scattering algorithm using CUDA [NVIDIA 2007]. First, to construct the matrix A and vector $\boldsymbol{b}$, $B_\ell(c_h)$, $\nabla B_\ell(c_h)$ and $\nabla^2 B_\ell(c_h)$ are computed for all pairs of (ℓ,h) in parallel. Then, for each c_h in parallel, the set of RBFs is iterated to compute $\tilde{D}(c_h)$, $\nabla\tilde{D}(c_h)$ and $\nabla^2\tilde{D}(c_h)$. Finally, the entries of A and $\boldsymbol{b}$ are computed according to Eq. (13).

The matrix-matrix multiplication $\mathsf{A}^T\mathsf{A}$ and matrix-vector multiplication $\mathsf{A}^T\boldsymbol{b}$ can be performed efficiently using CUBLAS [NVIDIA 2007], an optimized CUDA implementation of BLAS included with the CUDA SDK.

(a) $\sigma_t = 0.77,\ \Omega = 0.63$ (b) $\sigma_t = 3.34,\ \Omega = 0.17$

Figure 8: *Changing the optical parameters of smoke. The light comes mainly from the bottom of the volume, and the width of the smoke data is about* 4.0 *meters.*

A GPU CG solver for sparse matrices was described in [Bolz et al. 2003]. However, since the matrix $\mathsf{A}^T\mathsf{A}$ may be dense, we simply implemented a basic CG solver for dense matrices. The major computation in the CG solver involves matrix-vector multiplications and vector inner products, which can be efficiently done using CUBLAS.

Ray Marching The ray march starts from a volume pass that reconstructs the density field $D(u)$ and approximates the source radiance field $\tilde{J}(u, \omega_o)$ in Eq. (15).

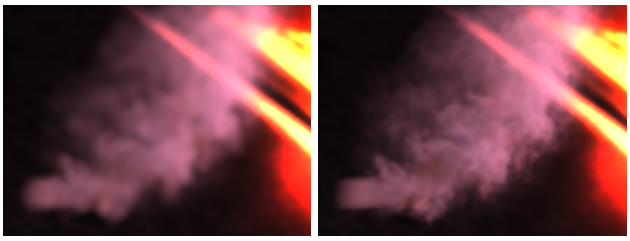
The projection matrix is set to be orthogonal, and the view direction is set to $-$Z. Each XY slice that intersects with at least one RBF is filled in one-by-one. First, a temporary frame buffer object (FBO) is bound, and alpha blending is enabled, with both the source and target blending factors set to GL_ONE. We then iterate over all the RBFs and find their intersections with the current slice, which can be trivially obtained given the center and radius of each RBF. A 2D bounding quad is then computed for the circular intersection region, and rendered to trigger a pixel shader. In the pixel shader, $w_\ell B_\ell(u_j)$ is evaluated and saved in the alpha channel, and $\tilde{J}_{D,\ell}(u_j) = w_\ell B_\ell(u_j)(\boldsymbol{y}(\omega_o) \cdot \tilde{\boldsymbol{J}}(c_\ell))$ is saved in the RGB channels. The residual $R(u_j)$ is retrieved from the hash table, multiplied with $w_\ell B_\ell(u_j)$, and then stored in the RGB channels of a second color buffer.

When all RBFs are processed, we have $\sum_\ell \tilde{J}_{D,\ell}(u_j)$, $\tilde{D}(u_j)$, and $R(u_j)\tilde{D}(u_j)$ in the color buffers, which are then bound as textures. We disable alpha blending and render the bounding quad of all the intersections, with the corresponding slice of a 3D texture bound to the render target. At each pixel, $\tilde{J}_D(u_j)$ is evaluated according to Eq. (15) as

$$\tilde{J}_D(u_j) = \left(1 + \frac{R(u_j)}{\tilde{D}(u_j)}\right) \sum_\ell \tilde{J}_{D,\ell}(u_j),$$

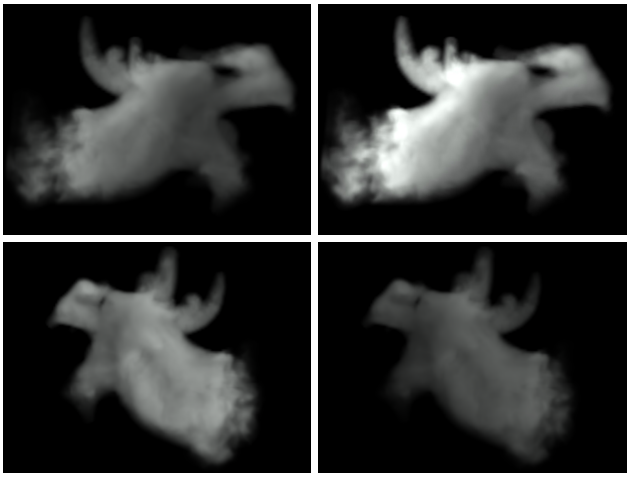
and saved in the RGB channels of the color buffer. To avoid division-by-zero exceptions, residuals are set to zero when $\tilde{D}(u)$ is very small ($< 1.0e-10$). The alpha channel is set to $D(u_i) = \tilde{D}(u_i) + R(u_i)$.

Once $D(u)$ and $\tilde{J}_D(u, \omega_o)$ have been reconstructed as a 3D texture, conventional ray marching can be conducted in the view pass to composite the radiance at screen pixels. First, the final color buffer is initialized with the background lighting L_{in}. The projection and model-view matrices are restored according to the current camera setting. Alpha blending is enabled again, with the source blending factor set to GL_ONE and the target blending factor set to GL_ALPHA. The slices perpendicular to the central frustum axis are processed in a far-to-near order. Each slice is intersected with all the RBFs. A bounding quad is found for the intersections, and is drawn to the final color buffer. $D(u)$ and $\tilde{J}_D(u, \omega_o)$ are retrieved from the 3D texture, and the RGB channels and alpha channels of the output color are set to $\tilde{J}_D(u_i)\sigma_t \Delta u / \cos\gamma$ and $\Delta\tau_i$ (Eq. (16)), respectively.



(a) without residual (b) with residual

Figure 9: *Fine smoke details generated by compensated ray marching. The density field is approximated by 647 RBFs.*



(a) constant (b) HG, $g = 0.42$

Figure 10: *Changing the phase function of smoke. The smoke is illuminated by a fixed light with a solid angle of* 10 *degrees. In the top row, the lighting direction is opposite from the viewing direction. In the bottom row, the viewing and lighting directions are roughly aligned.*

Separation of the ray marching procedure into a view-independent volume pass and a view pass leads to improved performance. Faster processing is obtained from reconstructing the density and source radiance at volume resolution, which is typically much lower than the screen resolution. It also allows graphics hardware to be used for trilinear interpolation of the residuals. Furthermore, if no changes are made to the density and source radiance field, e.g., when the user rotates the view and the phase function is constant, the 3D texture from the volume pass can be reused. The cost would be an additional 3D FBO of 16MB for a 128^3 volume.

6 Results and Discussion

We implemented our algorithm on a 3.7GHz PC with 2GB of memory and an NVidia 8800GTX graphics card. Images are generated at a 800×600 resolution. Please see the supplemental video for live demos. The three sets of smoke animation data used in this paper are all generated through physically-based simulation. For all the smoke data used in this paper, the volume resolution is $128 \times 128 \times 128$.

Smoke Visualization As a basic function, our system allows users to visualize smoke simulation results under environment lighting and from different viewpoints. With the proposed compensated ray marching, smoke with fine details can be visualized, as shown in Fig. 9. More examples are shown in the supplemental video.

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

Figure 11: *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. 8	Fig. 10
Frames	600	135	300
Avg. RBFs per frame	612	530	683
RBF approx. RMS error	2.13%	1.57%	2.20%
Decomposition (min)	87	22	51
Hashing (min)	35	9	24
Hash table (MB)	218	37	173
Performance (fps)	$19.1 \sim 95.1$	$36.4 \sim 57.8$	$35.8 \sim 74.8$

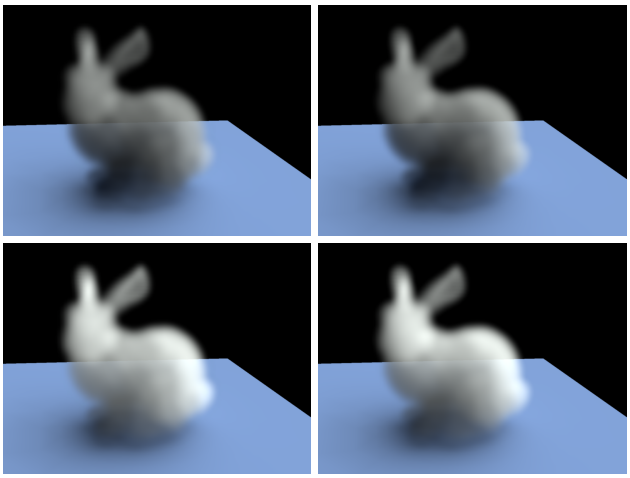
Table 2: *Statistics and timings. The total preprocessing time is the sum of the decomposition time and hashing time.*

The optical parameters of the smoke can be edited in real time. Both images of Fig. 8 contain the same smoke density field and environment lighting. Adjusting the albedo downward and increasing the extinction cross section darkens the appearance of the smoke and increases the shading variance. In Fig. 10, with fixed lighting we change the phase function from constant to the Henyey-Greenstein (HG) phase function (with eccentricity parameter 0.42). 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 objects, as shown in Fig. 11. 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 Sec. 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 solver and the compensated ray marching to generate results. Fig. 12 shows another example of shadow casting.

Performance Table 2 lists statistics for the three examples shown in the paper. Reasonable preprocessing times were needed, from 0.5 to 1.5 hours. In addition, the residual hash tables are significantly smaller than the original density field sequences.



(a) our results (b) ray tracing results

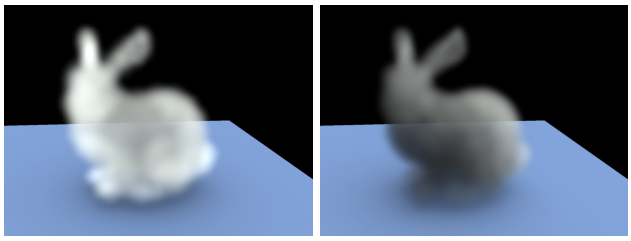
Figure 12: *Single scattering (top row) and multiple scattering (bottom row). Light arrives from the top-right corner of the volume, as indicated by the shadows on the plane below the smoke. The single scattering results appear dark in the occluded lower-left regions of the smoke. With multiple scattering, light transport through the volume yields a more realistic result. The smoke density ranges from* 0.0 *to* 2.0*,* $\sigma_t = 1.39$*,* $\Omega = 0.69$*, and the smoke volume width is about* 1.6 *meters.*

The bottleneck of our runtime algorithm is the source radiance computation, which takes up to 64% of the execution time. Single scattering comprises 20-30% of this time, and multiple scattering accounts for the rest. The cost of ray marching depends linearly on the number of slices N.

The cost of the single scattering simulation grows quadratically with the SH order. According to our experiments, spherical harmonics of order 4 produce plausible rendering results for all data shown in this paper. Lower orders of SH lighting, such as ambient (SH order 1) lighting, may produce unsatisfactory results though. Fig. 13 demonstrates single scattering results with lower order SH lighting for the scene shown in Fig. 12. With ambient lighting, shadows due to self-occlusion are nearly absent. With SH order 2, most occlusion effects are captured, but the lower portion of the smoke appears brighter than in Fig. 12.

The number of RBFs, n, is a dominant factor since it has a quadratic influence on the source radiance computation. For the example shown in Fig. 1, we generate in Fig. 14 approximations with different numbers of RBFs. And the average frame rates are 68.3 *fps* for 100 RBFs, 24.5 *fps* for 400 RBFs, and 12.9 *fps* for 800 RBFs. With 100 RBFs, the smoke appears thin and does not adequately capture the detail present in the original density field. At 400 RBFs, the rendering result reaches a level where it does not improve appreciably with greater numbers of RBFs.

Discussion In our implementation, the user can specify the number of RBFs for density approximation in the initial frame. The default value of 1000 works well for all examples shown in this paper. In principle, the source radiance at each voxel can be exactly reconstructed using a sufficiently large number of RBFs. A greater number of RBFs for approximating the density field leads to a better compression rate for the hash table and a higher precision in the represented shading variance. However, more RBFs also entails a higher computational cost, especially for the source radiance computation, whose performance depends roughly on the square of n. This tradeoff between accuracy and performance is an area for future investigation.



(a) ambient lighting (b) order-2 SH lighting

Figure 13: *Single scattering results using lower-order SH lighting.*

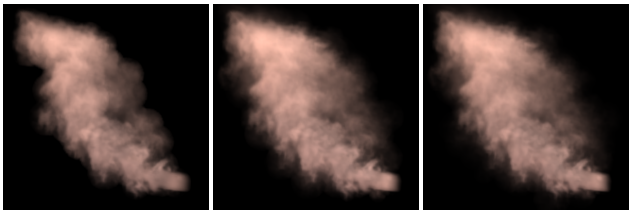


Figure 14: *Rendering results with different numbers of RBFs. From left to right: 100, 400 and 800.*

A limitation imposed by the assumption of low-frequency environment lighting is that our method cannot handle strongly directional phase functions. Their effects would be smoothed out by the low-frequency lighting, because convolving an order-4 SH vector with SH vectors of higher order would zero out the higher-frequency bands. According to our experiments, our method can effectively handle HG phase functions with values of g up to about 0.6. As the eccentricity of the phase function increases to 0.9, the low order SH representation becomes unable to capture the directional variance of the source radiance as shown in Fig. 15.

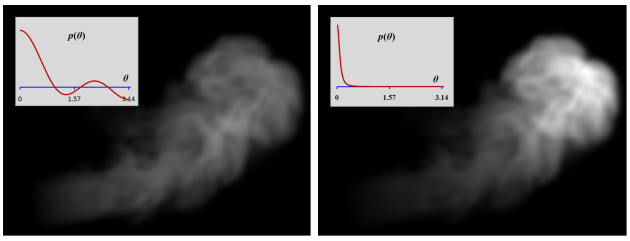
Our method also has difficulty in rendering sharp illumination effects such as shafts of light. A much greater number of RBFs and a high-frequency representation of lighting would be needed for a close visual approximation. In addition, convergence of the multiple scattering simulation would require many additional iterations.

We compute the source radiance due to multiple scattering by solving the diffusion equation, which is a reasonable approximation for optically dense media, i.e., the ratio between the mean free path and the characteristic length of the medium is less than $1/4$. On the other hand, thin media can also be handled by our algorithm by rendering single scattering alone. Media of intermediate optical density, however, may not be generated as accurately.

7 Conclusion and Future Work

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 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. Indeed, this method represents the first to render single and multiple scattering of smoke that is both real-time and a close match to offline ray tracing.

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



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

Figure 15: *Comparison between the low-frequency lighting used in our implementation and the all-frequency lighting of the original environment map. The medium is strongly forward scattering with a Henyey-Greenstein phase function of $g = 0.9$. The lighting comes from the back side of the volume, and the phase angle range is marked on the phase function curve.*

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 issues in future work.

Additionally, we are interested in reducing the cost of precomputation, or developing further approximations that would possibly eliminate the need for precomputation. Combining our method with the latest real-time smoke simulation technique [Crane et al. 2007] is also an interesting direction, which would make our technique directly useful for interactive simulations or games.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. Special thanks to Linjie Luo and Yizhou Yu for providing the smoke animation data. Hujun Bao was partially supported by the NSF of China (No. 60633070) and the 973 Program of China (No. 2002CB312102).

References

BIRI, V., MICHELIN, S., AND ARQUÈS, D., 2004. Real-time single scattering with shadows. http://igm.univ-mlv.fr/˜biri/indexCA_en.html.

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

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖODER, P. 2003. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Trans. Graph. 22*, 3, 917–924.

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 Trans. Graph. 23*, 3, 905–914.

CRANE, K., LLAMAS, I., AND TARIQ, S. 2007. Real-time simulation and rendering of 3d fluids. *GPU Gems 3*, Chapter 30.

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

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

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *ACM SIGGRAPH*, 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*, 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.

JAROSZ, W., DONNER, C., ZWICKER, M., AND JENSEN, H. W. 2007. Radiance caching for participating media. In *ACM SIGGRAPH 2007 Sketches*.

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

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray tracing volume densities. In *ACM SIGGRAPH*, 165–174.

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

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

LEFEBVRE, S., AND HOPPE, H. 2006. Perfect spatial hashing. *ACM Trans. Graph. 25*, 3, 579–588.

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

NARASIMHAN, S. G., AND NAYAR, S. K. 2003. Shedding light on the weather. In *IEEE Comp. Vision Patt. Rec.*, 665–672.

NVIDIA, 2007. CUDA homepage. http://developer.nvidia.com/object/cuda.html.

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 Trans. Graph. 25*, 3, 977–986.

RILEY, K., EBERT, D. S., KRAUS, M., TESSENDORF, J., AND HANSEN, C. 2004. Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering*, 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 *ACM SIGGRAPH*, 293–302.

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

SCHPOK, J., SIMONS, J., EBERT, D. S., AND HANSEN, C. 2003. A real-time cloud modeling, rendering, and animation system. In *ACM SIGGRAPH/Eurographics Symp. Computer Animation*, 160–166.

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

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

STAM, J., AND FIUME, E. 1995. Depicting fire and other gaseous phenomena using diffusion processes. In *ACM SIGGRAPH*, 129–136.

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

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

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S., AND NAYAR, S. 2005. A practical analytic single scattering model for real time rendering. *ACM Trans. Graph. 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.

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. In *Pacific Graphics*, 116–125.

ZHU, C., BYRD, R. H., LU, P., AND NOCEDAL, J. 1997. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Trans. Math. Softw. 23*, 4, 550–560.

Appendix: Spherical Harmonic Operations

Low-frequency spherical functions can be efficiently represented in terms of spherical harmonics (SHs). In this appendix, we briefly review spherical harmonic operations that are used in our algorithm.

Projection 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:

$$\boldsymbol{f} = (\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{n^2}) = \int_S f(s)\boldsymbol{y}(s)ds.$$

The first several SH basis functions are

$$\boldsymbol{y}_0(s) = \sqrt{\tfrac{1}{4\pi}},$$
$$\boldsymbol{y}_1(s) = -\sqrt{\tfrac{3}{4\pi}}y,\ \boldsymbol{y}_2(s) = \sqrt{\tfrac{3}{4\pi}}z,\ \boldsymbol{y}_3(s) = -\sqrt{\tfrac{3}{4\pi}}x$$

where x, y, z is the Cartesian coordinate of s. Note that the first two terms of a spherical function's Taylor expansion can be trivially converted to its SH coefficients of the first two orders, and vice versa.

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

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

Triple product Denoted by $\boldsymbol{f} * \boldsymbol{g}$, the SH triple product 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

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

Γ_{ijk} is symmetric, sparse, and of order 3 [Ren et al. 2006].

Convolution The convolution of two spherical functions $f(s)$ and $g(s)$ is given by

$$(f \star g)(s) = \int_S f(t) g\left(R_s(t)\right) dt$$

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 convolution*, denoted by $\boldsymbol{f} \star \boldsymbol{g}$, represents the order-$n$ projected result of convolving the reconstructions of two order-n vectors [Sloan et al. 2005]:

$$\begin{aligned} \boldsymbol{f} \star \boldsymbol{g} \quad &= \textstyle\int_S \int_S f(t) g\left(R_s(t)\right) \boldsymbol{y}(s)\, dt\, ds \ \Rightarrow \\ (\boldsymbol{f} \star \boldsymbol{g})_l^m \quad &= \sqrt{\tfrac{4\pi}{2l+1}}\, \boldsymbol{f}_l^m\, \boldsymbol{g}_l^0. \end{aligned}$$

Exponentiation Denoted by $\exp_*(\boldsymbol{f})$, SH exponentiation 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 $\hat{\boldsymbol{f}}$.