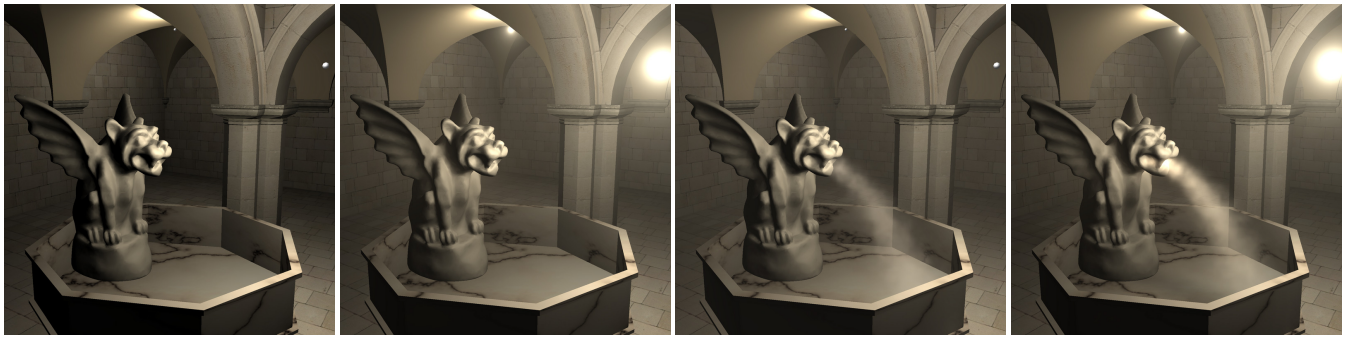
Fogshop: Real-Time Design and Rendering of Inhomogeneous, Single-Scattering Media

Kun Zhou Qiming Hou* Minmin Gong John Snyder† Baining Guo Heung-Yeung Shum

Microsoft Research Asia *Tsinghua University †Microsoft Research



(a) clear day (b) homogeneous media [Sun et al. 2005] (c) inhomogeneous media, simple blending (d) our result

Figure 1: *Rendering of scattering media. A point light source is located at the gargoyle's mouth. Notice the shadowing and haloing effects in (d).*

Abstract

We describe a new, analytic approximation to the airlight integral from scattering media whose density is modeled as a sum of Gaussians. The approximation supports real-time rendering of inhomogeneous media including their shadowing and scattering effects. For each Gaussian, this approximation samples the scattering integrand at the projection of its center along the view ray but models attenuation and shadowing with respect to the other Gaussians by integrating density along the fixed path from light source to 3D center to view point. Our approach handles isotropic, single-scattering media illuminated by point light sources or low-frequency environmental lighting. We also generalize models for reflectance of surfaces from constant-density to inhomogeneous media, using simple optical depth averaging in the direction of the light source or all around the receiver point. Our real-time rendering approach is incorporated into a system for real-time design and preview of realistic, animated fog, steam, or smoke.

Keywords: airlight, Gaussian, radial basis function (RBF).

1 Introduction

Scattering due to light transport in air or water is the source of many important visual effects which determine appearance of the directly-viewed media as well as the surfaces immersed within it. Such effects are critical for realism. Without self-shadowing, dense media such as clouds or smoke appear to emit rather than just reflect light, producing an overall implausible and cartoon-like effect. Lack of scattering ignores phenomena such as halos around lights which also substantially soften the shading on immersed surfaces.

In real-time applications such as 3D games, these scattering effects have been either completely neglected or approximated using restrictive "fog" models which assume the medium is entirely homogeneous or trivially layered. Such models exclude non-constant fog as well as more complex clouds and smoke whose optical density varies greatly over space and time. Traditional volume-rendering approaches support attenuation through inhomogeneous media by accumulating optical density in depth-sorted order over the discretized volume, but neglect scattering/shadowing effects. Full Monte Carlo scattering simulation yields an accurate rendering, but is far too expensive for the real-time demands of designers and end-users.

Our goal is to capture the shadowing and scattering effects of inhomogeneous media in real time. In this work we focus on single scattering: an effect which physically dominates when rendering fairly transparent media but remains useful to visually convey dense media as well. We represent the spatially-varying optical density as a sum of *radial basis functions* (RBFs), each a Gaussian centered around a 3D point. We then derive an analytic formula to approximate single scattering of this RBF model. It is evaluated by direct accumulation over only a few relevant RBFs per pixel and avoids expensive scattering simulation and integration over many samples along each light path.

The RBF model also facilitates real-time design of inhomogeneous media via user interactions such as brush strokes, copy/paste, and erase. Our rendering algorithm supports preview of all scattering effects as the model is edited. Simple particle-based simulation can then animate a model initially specified with our UI. In addition, existing simulations of smoke or clouds can be fit with RBFs and then rendered in our system, allowing real-time change to the lighting and view as the simulation plays back. While our results are not competitive with offline simulations used in feature films, our system provides compelling new content for 3D games, and a new design/preview tool for production-quality content.

Ours is the first analytic model of single scattering in inhomogeneous media, modeled as a sum of Gaussians. Our approximation assumes that variation in the 1D scattering integrand along each

*This work was done while Qiming Hou was an intern at Microsoft Research Asia.

view ray is due to variation in the RBF density, and neglects brightness variation due to close proximity of RBFs to light sources or light rays shining through abrupt density breaks. In the case of smooth media, our results accurately match an offline scattering simulation. In all cases, they are consistent, plausible, and visually capture light glows and self-shadowing effects. Most important, we render these effects in real-time, providing the feedback necessary for interactive design/verification and end-user applications.

2 Previous Work

There has been much previous work on rendering of scattering media dating back to [Blinn 1982]. See [Cerezo et al. 2005] for a detailed review.

Early approaches were based on ray tracing [Kajiya and Herzen 1984; Max 1994; Jensen and Christensen 1998] or radiosity [Rushmeier and Torrance 1987]. They produce a photorealistic image includng both single and multiple scattering effects in non-homogeneous media at the cost of hours of computation. Later techniques reduce computation by focusing on single scattering [Ebert and Parent 1990; Nakamae et al. 1990; Sakas 2004; Nishita et al. 1996]. Recent hardware-accelerated techniques [Harris and Lastra 2001; Dobashi et al. 2002; Riley et al. 2004] decrease running times still further by fixing the medium properties and scene specification, but performance remains unsuitable for real-time applications.

Several analytical models for scattering media have been used in computer graphics. [Blinn 1982] introduced the first of these for rendering homogeneous, single-scattering media, assuming an infinitely distant light source and viewer. [Willis 1987] presented a simple formula for fog consisting of homogeneous layers. Lighting is "baked in" the model, which assumes that each point in the media isotropically emits a constant radiance. [Preetham et al. 1999] described an airlight model for directional light sources to approximate the effects of atmosphere. [Max 1986; Narasimhan and Nayar 2003; Sun et al. 2005; Biri et al. 2006] present analytic expressions for the glows around point light sources. The model in [Sun et al. 2005] also handles the effects of airlight on surface shading (including arbitrary BRDFs), environmental lighting, and precomputed radiance transfer in the presence of homogeneous scattering media. It forms the basis for our work, which generalizes the analytic model to deal with inhomogeneous media.

For static scenes, techniques based on precomputation can be applied. Precomputed radiance transfer was originally applied to volumetric models as well as surfaces [Sloan et al. 2002]. [Premoze et al. 2004] presented an analytic expression for multiple scattering based on point spread functions precomputed from the media properties. Recently, [Szirmay-Kalos et al. 2005] described a multiple scattering method for cloud rendering based on a precomputed illumination network. None of these methods is suitable for rendering dynamic scenes or interactively designing the media, due to their huge precomputation and storage costs.

Using RBFs to model scattering media, such as clouds [Nishita et al. 1996; Dobashi et al. 2002], fog [Perlin 2006] and smoke [Stam 1995a], is not new. RBFs have proven a useful representation for rendering [Stam 1995b] as well as animation [Pighin et al. 2004]. As does our approach, [Stam 1995b] exploits analytic integration through a Gaussian density blob (see appendix), but simulates multiple scattering by an offline PDE solution. It first computes single-scattered radiance at blob centers using slow ray tracing. Our major contribution is an analytic model of single scattering which enables visually accurate rendering in real-time. We also provide a set of easy-to-use tools, such as airbrush and eraser, for interactive design of inhomogeneous media.

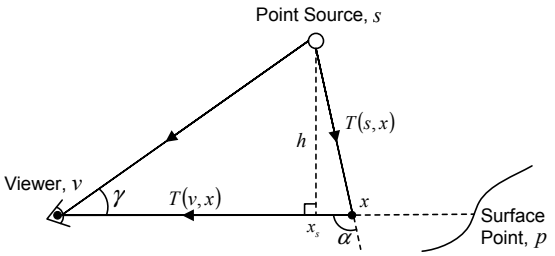


Figure 2: *Airlight.*

A technique for real-time rendering of smoke under low-frequency environmental lighting is proposed in [Zhou et al. 2007]. That technique accounts for multiple scattering as well as single scattering. It is not an analytic model but a technique for high-quality visualization of a given smoke animation sequence based on ray marching, which heavily depends on the volume sizes of the media. The technique also cannot handle local lighting and does not address the important issues of surface reflectance in inhomogeneous media and design of scattering media.

3 Airlight in Inhomogeneous Media

Airlight refers to the appearance of the illuminated scattering medium when viewed directly. The next section will use the principles developed here to render reflective surfaces immersed in the medium. Refer to Figure 2 in the following explanation.

Airlight is governed by *optical density* (or more precisely, the density times the scattering coefficient) denoted $\beta(x)$ where x is the 3D spatial parameter. Along a view ray r parameterized by a distance parameter t, we have

$$x(t) = v + t\,\hat{r} = v + t\,(p-v)/d_r \tag{1}$$

where v is the view point, p is the first surface point hit by the view ray, $d_r = d_{vp} = ||p-v||$ is the distance along the view ray to p, and $\hat{r} = \hat{r}_{vp} = (p-v)/d_{vp}$ is the unit-length view direction. Airlight L_a due to a point light source of intensity I_0 at location s which is scattered in the direction of r is given by the following 1D integral:

$$L_a = \int_0^{d_r} \beta(x)\,k(\alpha(x))\,\frac{I_0}{d^2(x)}\exp\left(-T(v,x)-T(x,s)\right)dt. \tag{2}$$

The function $d(x)$ is the distance of the light source s to x, given by

$$d(x) = d_{sx} = ||x-s|| = \sqrt{(x-x_s)^2+h^2}$$

where x_s is the point along the view ray closest to s, and $h = ||s-x_s||$ is the distance of the source to the view ray. $k(\alpha)$ is the scattering phase function where the scattering angle α is defined by $\cos(\alpha(x)) = (x-x_s)/d(x)$. As in [Sun et al. 2005], we assume isotropic scattering for which $k(\alpha) = \frac{1}{4\pi}$, but our approximation can be applied to anisotropic scattering as well. Since x is a function of the ray parameter t, so are d and α.

The optical depth between two 3D points a and b, $T(a,b)$, is given by the 1D integral of optical density between a and b:

$$T(a,b) = \int_0^{d_{ab}} \beta(a+t\,\hat{r}_{ab})\,dt \tag{3}$$

where $d_{ab} = ||a-b||$ and $\hat{r}_{ab} = (b-a)/d_{ab}$. Direct attenuation of light along the path from a to b is then given by $\exp(-T(a,b))$.

To simplify the notation, we define

$$f(x) = k(\alpha(x))\,\frac{I_0}{d^2(x)}\exp\left(-T(v,x)-T(x,s)\right), \tag{4}$$

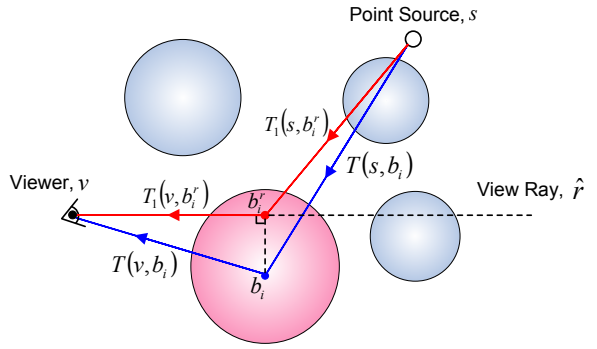


Figure 3: *Light paths for scattering integration. The red path is used for the Gaussian i itself, represented as the pink sphere. The blue path, which is independent of the view ray $\hat{r}$, is used to integrate the rest of the Gaussians $j \neq i$, drawn as blue spheres.*

so that

$$L_a = \int_0^{d_r} \beta(x)\, f(x)\, dt. \tag{5}$$

We model the inhomogeneous medium as a sum of RBFs:

$$\beta(x) = \sum_{i=1}^{n} \beta_i(x) + \beta_0 \tag{6}$$

where

$$\beta_i(x) = c_i \exp\left(-a_i^2\, ||x - b_i||^2\right). \tag{7}$$

a_i represents the Gaussian's scale, b_i its center, and c_i its amplitude.

Then expanding $\beta(x)$ in (5), we obtain

$$L_a = \sum_{i=1}^{n} \int_0^{d_r} \beta_i(x)\, f(x)\, dt + \beta_0 \int_0^{d_r} f(x)\, dt = \sum_{i=1}^{n} L_i + L_0. \tag{8}$$

Gaussian Terms To evaluate the Gaussian terms

$$L_i = \int_0^{d_r} \beta_i(x)\, f(x)\, dt, \tag{9}$$

we assume the variation in $f(x)$ is small with respect to the variation in the RBFs $\beta_i(x)$. According to the mean-value theorem for integration, there exists a $0 \leq t_m \leq d_r$ such that $L_i = f(x_m) \int_0^{d_r} \beta_i(x)\, dt$ where $x_m = x(t_m)$. Since $\beta_i(x)$ is a Gaussian, most of its energy concentrates at the projection of its center to the view ray:[1]

$$b_i^r = v + ((b_i - v) \cdot \hat{r})\, \hat{r}. \tag{10}$$

So as an approximation, we take $x_m = b_i^r$, yielding

$$L_i \approx f(b_i^r) \int_0^{d_r} \beta_i(x)\, dt \tag{11}$$

Equation (31) in the appendix shows how a Gaussian can be analytically integrated along the view ray, allowing evaluation of the second factor $\int_0^{d_r} \beta_i(x)\, dt$.

According to (4), evaluating $f(b_i^r)$ involves computing the optical depths $T(v,b_i^r)$ and $T(s,b_i^r)$ from b_i^r to the view point v and light point s. But it is impractical to compute these by summing over all n RBFs for each view ray. We instead use the correct light path $s \to b_i^r \to v$ for the Gaussian i itself, but simplify it to $s \to b_i \to v$ for the rest of the Gaussians $j \neq i$ (see Figure 3). The second light path is simpler because it no longer depends on the view ray. This yields the approximate factorization $f(b_i^r) \approx f^0(b_i^r)\, f^1(b_i)$ where

$$f^0(b_i^r) = \frac{1}{4\pi} \frac{I_0}{||b_i^r - x_s||^2 + h^2}\, e^{-T_i(s,b_i^r) - T_i(v,b_i^r) + T_i(s,b_i) + T_i(v,b_i)} \tag{12}$$

and

$$f^1(b_i) = \exp\left(-T(s,b_i) - T(v,b_i)\right). \tag{13}$$

Indexed optical depth $T_i(a,b)$ is defined as in (3), but with respect to the i-th Gaussian $\beta_i(x)$ alone; i.e. $T_i(a,b) = \int_0^{d_{ab}} \beta_i(a + t\,\hat{r}_{ab})\, dt$. The addition of the $T_i(s,b_i) + T_i(v,b_i)$ term in the exponent of f^0 compensates for using optical depth T with respect to *all* Gaussians in f^1, when f^0 has already accounted for Gaussian i via the correct light path.

The advantage of this factorization is that f^1 does not vary per view ray, and f^0 can be now computed using four Gaussian line integrals rather than n. Furthermore, only two of these four line intergals vary per view ray; the other two are computed in a separate pass as described in Section 5.1. When the points b_i and b_i^r are close, this factorization is clearly accurate. It is also accurate when they are distant since both L_i and our approximation to it then approach 0.

Homogeneous Term To evaluate the homogeneous term

$$L_0 = \beta_0 \int_0^{d_r} f(x)\, dt, \tag{14}$$

we apply a similar factorization trick based on approximate light paths. We split L_0 into two factors by separately considering the light path $s \to x \to v$ with respect to the homogeneous medium modeled by β_0, and the simpler light path $s \to v$ for the RBF sum modeling the medium inhomogeneity. This yields

$$\begin{aligned} f(x) &\approx \frac{1}{4\pi} \frac{I_0}{d^2(x)}\, e^{-\beta_0 (||v-x|| + ||s-x||)}\, e^{-T(s,v) + \beta_0 ||s-v||} \\ &= C_{sv} \frac{1}{4\pi} \frac{I_0}{d^2(x)} \exp\left(-\beta_0 (||v-x|| + ||s-x||)\right) \end{aligned} \tag{15}$$

where

$$C_{sv} = \exp(-T(s,v) + \beta_0 ||s-v||). \tag{16}$$

With this approximate $f(x)$, the integration in (14) can now be done analytically [Sun et al. 2005], since the only dependence on x in the integrand is with respect to a constant density β_0. Summarizing that method briefly here, homogeneous airlight due to a constant density β, denoted $L_a^h(\gamma, d_{sv}, d_{vp}, \beta)$, is given by

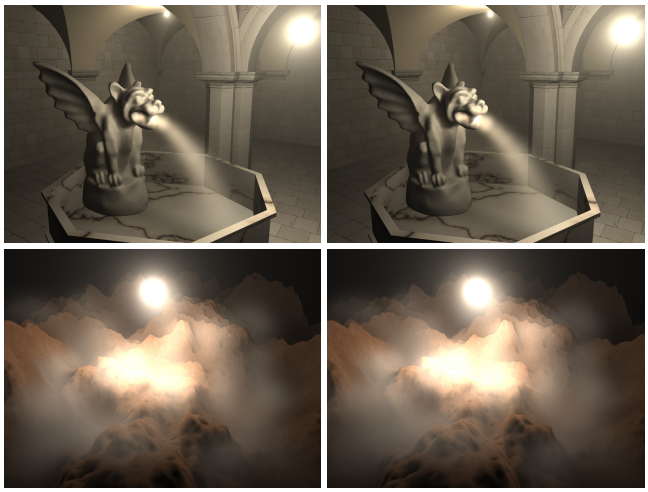
$$L_a^h = A_0 \left[F\left(A_1, \frac{\pi}{4} + \frac{1}{2} \arctan\left(\frac{T_{vp} - T_{sv}\cos\gamma}{T_{sv}\sin\gamma}\right)\right) - F\left(A_1, \frac{\gamma}{2}\right)\right]$$

where $T_{sv} = \beta\, d_{sv}$, $T_{vp} = \beta\, d_{vp}$, $A_0 = \frac{\beta^2 I_0\, e^{-T_{sv}\cos\gamma}}{2\pi T_{sv}\sin\gamma}$, $A_1 = T_{sv}\sin\gamma$, and $F(u,v) = \int_0^v \exp(-u\tan\xi)\, d\xi$. γ is the angle formed by the view direction $\hat{r}$ and the direct path from view point to light point; i.e. $\cos\gamma = \hat{r} \cdot \hat{r}_{sv}$. Using this formula, the homogeneous term in (14) is then given by

$$L_0 \approx C_{sv}\, L_a^h(\gamma, d_{sv}, d_{vp}, \beta_0). \tag{17}$$

Approximation Comparison In most cases, our approximation accurately matches a full, Monte-Carlo simulation of single scattering, as shown in Fig. 4. Figure 7 shows cases in which this match is less accurate. See Section 7 for further discussion.

[1] The projection should be restricted to the segment on the view ray from v to p.



(a) our result (b) ray tracing

Figure 4: *Comparison: our result vs. ray traced simulation.*

4 Surface Reflectance in Inhomog. Media

We denote by L_p the reflected radiance of the surface at point p emitted back to the viewpoint v when illuminated by airlight. L_p can be computed using the *point spread function* or PSF, governing how radiance is blurred and attenuated by the scattering medium before illuminating the surface. Using PSFs will allow our model to be extended to environmental lighting, arbitrary BRDFs, and precomputed radiance transfer (PRT) [Sloan et al. 2002].

As shown in [Sun et al. 2005] for homogeneous media, single-scattered radiance $L_p^{\text{in}-\text{ss}}(\omega)$ incident at a surface point p in all directions ω can be accurately approximated by the following spherical convolution

$$L_p^{\text{in}-\text{ss}}(\omega) = (L_p^{\text{in}} * \text{PSF})(\omega), \;\; \text{PSF}(\gamma) = T_{sp}\, e^{-T_{sp}}\, \text{NPSF}(\gamma), \quad (18)$$

where $L_p^{\text{in}}(\omega)$ is the radiance incident at p neglecting the scattering medium, γ is the angle between the original and scattered lighting directions, and $T_{sp} = \beta\, d_{sp}$ is the optical depth of the medium from s to p. Spherical convolution is denoted by $f * g$ where f and g are spherical functions and g is circularly symmetric about the (canonical) z axis. $\text{NPSF}(\gamma)$ is a spherical function that depends only on the scattering phase function but is independent of the scattering medium:

$$\text{NPSF}(\gamma) = \frac{F(\sin\gamma, \frac{\pi}{2}) - F(\sin\gamma, \frac{\gamma}{2})}{2\pi \sin\gamma \cdot e^{(\cos\gamma - 1)}}.$$

In other words, the scattering effect of the medium on incident radiance can be approximated by a constant convolution with NPSF followed by a multiplication with the scalar $T_{sp}\, e^{-T_{sp}}$.

The total illumination incident at p then sums the singly scattered plus directly attenuated incident illumination:

$$\begin{aligned} L_p^{\text{in}-\text{tot}}(\omega) &= L_p^{\text{in}-\text{ss}}(\omega) + L_p^{\text{in}-\text{att}}(\omega) \quad (19) \\ &= T_{sp}\, e^{-T_{sp}}\, (L_p^{\text{in}} * \text{NPSF})(\omega) + e^{-T_{sp}}\, L_p^{\text{in}}(\omega). \end{aligned}$$

Illuminating the surface using this PSF-based approximation, the outgoing radiance at p in the view direction is given by the scalar

$$L_p = \left\langle L_p^{\text{in}-\text{tot}} | V_p | B_{pv} \right\rangle, \quad (20)$$

where the triple product is defined by the spherical integral

$$\langle f_1 | f_2 | f_3 \rangle = \int_{\omega \in S} f_1(\omega)\, f_2(\omega)\, f_3(\omega)\, d\omega.$$

The spherical function V_p represents visibility of the distant environment at p (due to the presence of scene occluders, not the medium), and B_{pv} represents the BRDF assuming p is being viewed from v [Ng et al. 2004].[2]

We use spherical harmonic (SH) vectors of order 4-6 for lighting, BRDF, and visibility/PRT. Low order vectors represent only low-frequency directional dependence, which is appropriate for fairly matte surfaces or smooth lighting.

SH Review Let $f(\omega)$ be a spherical function, represented by the SH vector $\boldsymbol{f}_{lm}$. An order-n expansion requires n^2 vector components. Let $g(\omega)$ by a function circularly symmetric about z, which can be represented by the SH vector $\boldsymbol{g}_l$ (its symmetry implies its only nonzero coefficients have m=0). Convolving $\boldsymbol{f}$ by $\boldsymbol{g}$ yields:

$$(\boldsymbol{f} * \boldsymbol{g})_{lm} = \sqrt{\frac{4\pi}{2l+1}}\, \boldsymbol{f}_{lm}\, \boldsymbol{g}_l. \quad (21)$$

Evaluating $\boldsymbol{f}$ at the spherical point ω is computed via

$$\boldsymbol{f}(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{+l} \boldsymbol{f}_{lm}\, \boldsymbol{y}_{lm}(\omega). \quad (22)$$

where $\boldsymbol{y}_{lm}$ are the SH basis functions. Rotating $\boldsymbol{g}$ from its canonical center at z to an arbitrary one z' is computed via

$$\text{rot}(\boldsymbol{g}, z \to z')_{lm} = \sqrt{\frac{4\pi}{2l+1}}\, \boldsymbol{g}_l\, \boldsymbol{y}_{lm}(z'). \quad (23)$$

An SH delta function, $\boldsymbol{\delta}$, is the "peakiest" or most directional function that can be produced by a given SH order. If it is canonically centered around z, its coefficients are given by

$$\boldsymbol{\delta}_l = \boldsymbol{y}_{l0}(z). \quad (24)$$

For convenience, we list the first 6 SH coefficients of NPSF (as with any circularly symmetric function about z, only the m=0 components are nonzero): 0.332818, 0.332488, 0.302428, 0.275773, 0.254051, 0.236333. These form the $\boldsymbol{g}_l$ coefficients used in the convolution formula (21).

4.1 Point Lighting

For a point light source:

$$\boldsymbol{L}_p^{\text{in}} = \frac{I_0}{d_{ps}^2}\, \boldsymbol{\delta}_{ps},$$

where $\boldsymbol{\delta}_{ps}$ is the delta function in the direction from p to s. Its SH coefficients can be computed using formulas (24) and (23).

To calculate (19), we make the approximation that the optical density equals the average density from s to p. This simply replaces the optical depth $T_{sp} = \beta\, d_{sp}$, in that formula assuming a homogeneous density β, with the integrated version, $T(s,p)$, with respect to the inhomogeneous medium along the path from s to p as defined in (3). We thus obtain the SH vector

$$\boldsymbol{L}_p^{\text{in}-\text{tot}} = \frac{I_0 e^{-T(s,p)}}{d_{ps}^2} \left(T(s,p)\, \boldsymbol{\delta}_{ps} * \text{NPSF} + \boldsymbol{\delta}_{ps} \right). \quad (25)$$

[2] Actually, the separation of object visibility V_p from incident radiance $L_p^{\text{in}-\text{tot}}$ implied by this triple product formula is an approximation: the two should properly be integrated together along each light path. This approximation assumes that the shadowing scene objects are nearby with respect to the medium's extent.

This approximation works well because the incident illumination is a delta function in the direction $\hat{r}_{ps}$. Thus, singly-scattered airlight drops to 0 rapidly as the angle γ with respect to $\hat{r}_{sp}$ gets larger. The approximation therefore captures the inhomgeneous medium's variation with direction well, by integrating over the actual medium in the single direction $\hat{r}_{ps}$. Optical depth $T(s,p)$ is computed using an RBF splatting method which will be described in the next section.

4.2 Environmental Lighting

We model distant environmental lighting using a spatially invariant SH vector $\boldsymbol{L}^{\text{in}}$. To model how this light is scattered before hitting a receiver point p, we make the approximation that the optical depth equals the average depth in all directions around p, defined by

$$\bar{T}(p) = \frac{1}{4\pi}\int_{\omega\in S} T(p+d_\omega\omega, p)\,d\omega. \tag{26}$$

where $S = \{\omega | \omega_x^2+\omega_y^2+\omega_z^2 = 1\}$. Then we simply replace the optical depth T_{sp} in (19) with this average depth $\bar{T}(p)$, yielding

$$\boldsymbol{L}_p^{\text{in-tot}} = \bar{T}_p\, e^{-\bar{T}_p}\,(\boldsymbol{L}^{\text{in}} * \text{NPSF}) + e^{-\bar{T}_p}\,\boldsymbol{L}^{\text{in}}. \tag{27}$$

To compute $\bar{T}(p)$, we have

$$\begin{aligned}\bar{T}(p) &= \tfrac{1}{4\pi}\textstyle\int_{\omega\in S}\int_0^D \beta(p+t\omega)\,dt\,d\omega \\ &= \beta_0 D + \textstyle\sum_{i=1}^n \left(\tfrac{1}{4\pi}\int_{\omega\in S}\int_0^D \beta_i(p+t\omega)\,dt\,d\omega\right) \\ &= \beta_0 D + \textstyle\sum_{i=1}^n \bar{T}_i(p),\end{aligned}$$

where $D > d_\omega$ bounds the distance of p to the environment. We use a fixed and large value for all points and all directions, which assumes the size of the object is small compared to the distance to the environment map.

$\bar{T}_i(p)$ is the average optical depth from the i-th Gaussian β_i. To calculate it, we first tabulate the average optical depth of a special Gaussian with $a = c = 1$ and $b = 0$ as a 1D table:

$$T(\|u\|) = \frac{1}{4\pi}\int_{\omega\in S}\int_0^\infty \exp\left(\|u+t\omega\|^2\right)\,dt\,d\omega,$$

where u is a point on the z axis. Since D is large, we obtain

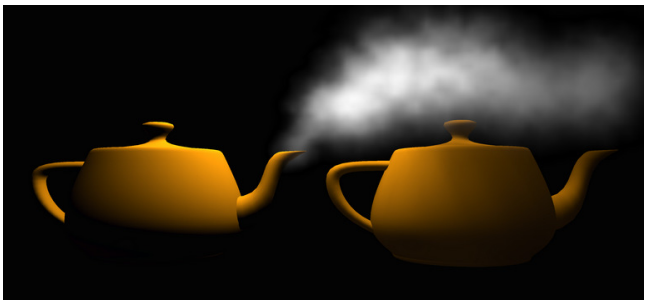
$$\bar{T}_i(p) = \frac{c_i\,T\,(a_i\|p-b_i\|)}{a_i}.$$

$\bar{T}(p)$ is then computed by summing each Gaussian's contribution $\bar{T}_i(p)$.

4.3 Shading with PSF-Scattered Radiance

Given the total scattered radiance incident at p, $\boldsymbol{L}_p^{\text{in-tot}}$ defined in (25) or (27), we can shade by applying (20). Efficient methods for computing the SH triple product are described in [Snyder 2006].

We can also specialize (20) in two important cases: when shading with an arbitrary BRDF but without PRT shadowing, or with a diffuse receiver with PRT. We denote the SH vector representing the BRDF weighting assuming a view point v as $\boldsymbol{B}_{pv}$. A PRT vector represents how the object shadows and inter-reflects light onto itself at receiver point p with respect to a low-frequency, distant lighting basis and is represented by the SH vector $\boldsymbol{P}_p$. Then the resulting shading in either case is obtained simply by dotting $\boldsymbol{L}_p^{\text{in-tot}}$ with either $\boldsymbol{B}_{pv}$ or $\boldsymbol{P}_p$. This requires only a simple dot product rather than an expensive SH triple product.



(a) diffuse, point light source

(b) Phong model (exponent=5), point light source

(c) fitted BRDF from measured data, environmental lighting

Figure 5: *Surface reflectance in inhomogeneous media.*

If the view ray does not hit any surface point, we would still like to see glows around bright sources in the environment. The PSF-based approximation (18) can be used to calculate the environmental airlight via

$$L_a(\hat{r}) = T(v,p)\,e^{-T(v,p)}\,(L^{\text{in}} * \text{NPSF})(\hat{r}), \tag{28}$$

where $T(v,p)$ is the screen optical depth computed as described in Section 5.2. In this case, the integration depth $d_r \to \infty$ since no surface is hit. We precompute the convolution of the lighting environment with NPSF and store this as a cube map.

Note that the PSF method can easily be specialized to diffuse or Phong BRDF models. On the other hand, it is also possible to generalize the model in [Sun et al. 2005] (equations 17,18) for reflectance of a Lambertian plus specular Phong surface in airlight, using the same approach of replacing its $T_{sp} = \beta\, d_{sp}$ (which assumes constant optical depth) with the inhomogeneous depth integrated along the path, $T(s,p)$. While this method is restricted to the diffuse+Phong surface reflectance model, it is theoretically more accurate in that case. We find the results of the two methods almost indistinguishable for diffuse surfaces.

Fig. 5 illustrates scattering effects on surface reflectance. Steam emitted from the teapot on the left scatters light which affects the appearance of the teapot on the right. Notice the softening in the shading and specular highlights, and the steam's shadow.

5 Rendering Pipeline

As in [Sun et al. 2005], the total radiance arriving at the view ray r, denoted L, is modeled via

$$L = L_a + \exp(-T(v,p))\, L_p. \tag{29}$$

This equation supports attenuation through the medium but neglects scattering effects once the light leaves the surface point p. Since surfaces are typically much dimmer than light sources, capturing just this first-order effect is a reasonable approximation.

5.1 Computing $T(v,b_i)$ and $T(s,b_i)$

Computing airlight L_a requires the factor $f^1(b_i)$, which in turn requires exponentiating $T(v,b_i)$ and $T(s,b_i)$ at each of the n Gaussian centers b_i. We describe an algorithm for computing $T(v,b_i)$; substituting the light source position s for v as the ray origin then allows computation of $T(s,b_i)$.

A plane sweep is performed on the CPU to find the subset of RBFs that contribute along each of the n rays from v to b_i. Each ray direction is represented by the spherical point $\hat{b}_i = (b_i - v)/||b_i - v||$ which is converted to 2D spherical coordinates, (θ_i, ϕ_i). We then bound each RBF using an interval over 2D spherical coordinates such that the line integration result for any ray with origin v is sufficiently small outside this bound. From (31), it can be seen that the line integral declines exponentially as a function of distance $||b_i - v||$ and the sine of the angle ξ between $\hat{r}$ and $\hat{b}_i$, due to the factor

$$c_i\, e^{-a_i^2\, ||\hat{r}\times(b_i-v)||^2} = c_i\, e^{-a_i^2\, ||b_i-v||^2 \sin^2\xi}.$$

Thus, we base the bounding box on the threshold

$$\sin\xi \le \frac{\sqrt{\ln c_i - \ln\varepsilon}}{a_i\, \|b_i - v\|} = \sin\xi_i \tag{30}$$

where $\varepsilon = e^{-9}$. This represents a cone around the central direction $\hat{b}_i$.

A segment search tree algorithm [O'Rourke 1998] is then used to query the subset of RBFs whose 2D intervals cover each spherical point (θ_i, ϕ_i), producing a list of $n_i \le n$ RBFs denoted $\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_{n_i}}$ which have a contribution on the ray from v to b_i. The complexity of the algorithm is $O(n \log n + k)$ where n is the number of RBFs and k is the number of intervals reported. The list for each i is then sent to the GPU, which performs the 1D Gaussian integration using equation (31) for each of the β_{i_j}, yielding $T_{i_j}(v,b_i)$. Finally, the results over all n_i Gaussians are summed to obtain $T(v,b_i)$.

5.2 Integrating Optical Depth in All Directions

We integrate optical depth around the view point v and each light point s. This is similar to the computation of the previous section, except that the integration is done in all directions around each ray origin instead of to n Gaussian centers b_i, and the integration proceeds from the ray origin until the first intersected surface instead of stopping at the Gaussian center. Maps of optical depth are computed around the view point for attenuating airlight in and (29), and around each light point for rendering surface reflectance in (20).

We use an RBF splatting techique on the GPU. The screen is used as the render target when accumulating optical depth around the view point; 6 images forming a cube map are used for light sources. For each Gaussian i, we first compute a bounding sphere with radius $r_i = \|b_i - v\| \sin\xi_i$ around its center b_i. This threshold from (30) ensures that $||x - b_i|| > r_i \Rightarrow \beta_i(x) < \varepsilon$. The bounding sphere is then projected to the near plane of the view frustum and its 2D bounding box rasterized. A pixel shader is invoked to compute the 1D integral along that pixel's view ray using (31). All Gaussians are then accumulated using alpha blending hardware to produce the per-pixel integrated result.

5.3 Accumulating Airlight

We use the following algorithm to simultaneously accumulate optical depth around the view point as well as integrate airlight. Here, L_a, T, and d_{vp} denote maps over all screen pixels. Map references are thus implicitly evaluated at each pixel's view ray.

$(L_a, T) \leftarrow (0,0)$
For each pixel
    T += $\beta_0\, d_{vp}$
    For each point light source
        Compute the homogeneous term of airlight, L_0, via (17)
        L_a += L_0
For each Gaussian i
    For each pixel covered by its bounding box
        Compute line integral along view ray, $T_i(v,p)$, via (31)
        $L_i \leftarrow 0$
        For each point light source
            L_i += $f^0(b_i^r)\, f^1(b_i)\, T_i(v,p)$
        (L_a, T) += (L_i, T_i) // accumulate to airlight target
For each pixel covered by the environment map
    L_a += $(L^{\text{in}} * \text{NPSF})(\hat{r})\; T \exp(-T)$

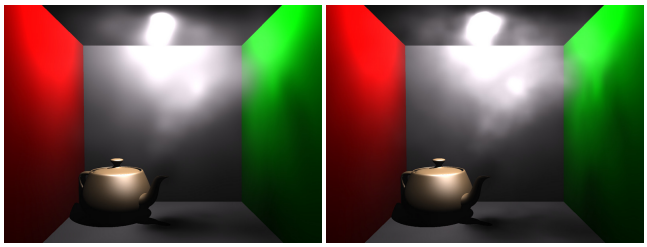
5.4 Rendering Summary

The following steps summarize our entire rendering pipeline:

1. Render view distance and light distance maps, d_{vp} and d_{sp}.
2. Accumulate the optical depth map around each light source, $T(s,p)$, using the RBF splatting described in Section 5.2.
3. If there is a environment map, accumulate the average optical depth for each vertex, $\bar{T}(p)$.
4. Render the scene (i.e. compute the vertex shading L_p) using incident lighting from (25) or (27). as described in Section 4.3.
5. Compute $T(v,b_i)$, $T(s,v)$ and $T(s,b_i)$ using the plane sweep algorithm described in Section 5.1.
6. Accumulate airlight using the algorithm in Section 5.3, yielding the airlight L_a and screen optical depth $T(v,p)$ targets. In our implementation, 4 lights are packed together and treated in a single pass.
7. Attenuate the scene target using the optical depth target and add the airlight target, via (29).

Step 3 forms the bottleneck in our computation. To speed things up, instead of computing $\bar{T}$ for each vertex, we compute it only at the centroid of each object. All the object's vertices then share the same $\bar{T}$. A more sophisticated method could use VQ clustering [Linde et al. 1980] to generate a small set of uniformly-distributed representatives which are blended at each vertex. We deem this as future work.

Step 6 is also computationally expensive. We compute the airlight and screen optical depth targets at lower (e.g. 1/4) resolution. The distance map d_{vp} is first down-sampled. After the airlight and screen optical depth are computed (at reduced resolution), we up-



(a) without noise (b) with noise

Figure 6: *Teapot in a box: adding noise makes the steam look more realistic.*

sample them back to the screen resolution. For pixels whose full-resolution neighborhood spans too great a range in the depth map, we use the low-resolution result having the smallest difference with the desired high-resolution depth value. The rest of the pixels are bilinearly interpolated.

5.5 Adding Noise

Our rendering system can add noise to convey the irregularity of real fog and smoke without unduly increasing the number of RBFs (see Figure 6). This is done by perturbing $T(v,p)$ from Section 5.3. More precisely, when computing $T_i(v,p)$ for each pixel covered by a Gaussian i, we perturb the view ray using a tileable 3D Perlin noise texture and compute the line integral along this perturbed direction. The integration distance d_r is left unchanged.

The noise texture is indexed by the 3D point b_i^r. The result is then scaled by $r_i/||v-b_i||$, transformed to view space by multiplying by the current view matrix, and finally added to the original direction. Adding noise in world space ensures consistency when the camera changes. The scale of the perturbation is user-adjustable. We also add a constant displacement to the b_i^r noise texture index which can be animated.

6 Creating Inhomogeneous Media

Making use of the RBF representation, we develop a set of easy-to-use tools to create inhomogeneous media, including paintbrush, airbrush, eraser, and particle system simulator [Reeves 1983]. Existing animation data of smoke or clouds generated using advected RBFs [Pighin et al. 2004] or a commercial animation system (e.g. Maya) can also be imported and rendered in our system.

Copy/Paste Our system allows the user to select RBFs in the scene, and copy or move them elsewhere. The user simply draws a rectangle on the screen to select RBFs whose center projects inside the rectangle.

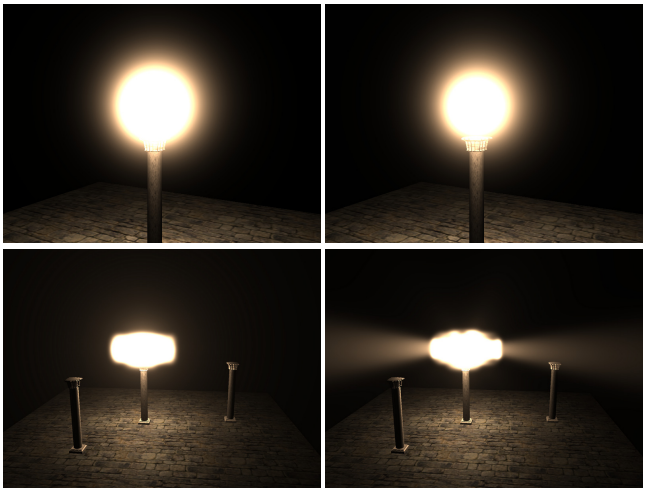
Paintbrush The paintbrush places Gaussians along a stroke drawn by the user. The stroke is projected onto a 3D, user-specified plane. Both the amplitude c and scale a of the Gaussians can be adjusted. The distance between two adjacent Gaussians along the stroke can also be changed ($0.75/a$ by default). We move the Gaussian centers by a random vector lying in the plane perpendicular to the stroke. The length of the offset vector is less than $1/a$.

Eraser The eraser tool reduces the density of those Gaussians it covers. Once a Gaussian's density reaches zero, it is deleted. The radius of the eraser can be adjusted.

Particle Emitter The user can place an emitter at any point in the scene, which then spawns particles. The particle's trajectory is a simple, constant-acceleration (parabolic) path. The spawning rate, acceleration, initial velocity, color, and lifetime of the particles can be adjusted. Gaussians are placed at the centers of the particles.

Scene	# Vertices	# Gaussians	# Lights	FPS
gargoyle (Fig. 1)	78,054	34	3	101
box (Fig. 6)	8,901	3008	1	34
terrain (Fig. 8)	65,047	292	env. map	92
city (Fig. 8)	77,226	353	env. map	162
motorcycle (Fig. 8)	44,430	1223	env. map	31

Table 1: *Statistics and timings.*



(a) our result (b) ray tracing

Figure 7: *Failure cases of our airlight approximation.*

The scale and amplitude of a Gaussian are determined by the particle's lifetime: the longer the particle lives, the smaller its scale and amplitude.

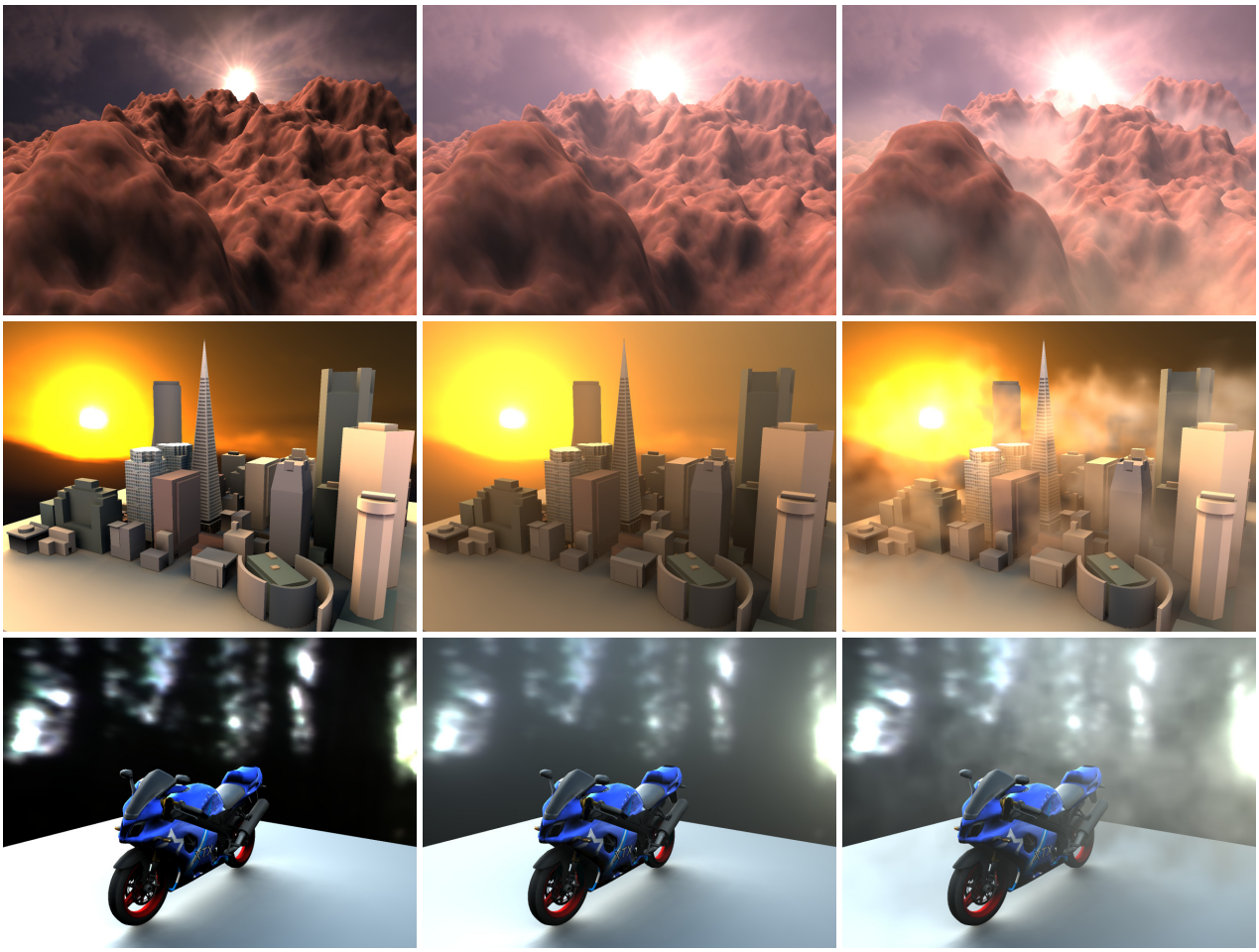
Airbrush The airbrush is similar to the particle emitter, except that its particles have infinite lifetime and bounce off surfaces in the scene. The particles eventually diffuse out randomly, but confined within open regions of the scene bounded by surfaces. When the airbrush is stopped, all of its particles are frozen. Users can employ this tool to generate a foggy area, or fill a 3D model with particles to create a foggy version of the same shape.

7 Results

We have implemented our system on a 3.7Ghz PC with 2GB of memory and an NVidia 8800GTX graphics card. Image generation was done at 800×600 resolution. Table 1 summarizes statistics for the various examples. Please see the video results for animated versions of the figures and other live demos, recorded in real time.

Dynamic smoke in Fig. 1 is generated by the particle emitter. The teapot-in-a-box scene in Fig. 6 is filled with fog using the airbrush tool. Note that our noise scheme disturbs the optical depth consistently in world space and makes the media appear more irregular and realistic. We also import an off-line simulation of steam rising from the teapot's spout. The user is able to visualize simulation results in real time, including interactive lighting and camera change.

The fog in the terrain and city scenes shown in Fig. 8 is created using our paintbrush tool (see the video). The motorcycle scene uses the particle emitter. All three scenes demonstrate how inhomogeneous media enhances realism. We obtain several realistic effects, including soft shadows, glowing of the environmental lighting, and softening of highlights and surface reflectance, all in real time. Combined with PRT, our approach provides an attractive solution for rendering dynamic, inhomogeneous media in applications like 3D games.



(a) clear day (b) homogeneous media (c) inhomogeneous media

Figure 8: *Results combining our scattering model with PRT.*

Discussion of Approximation Limitations Although our approximation of the airlight integral achieves visually realistic results, it is inaccurate in a few cases. Since our approximation samples the integrand at the projection of each Gaussian onto the view ray, it may miss samples where the the view rays' distance $d(x)$ to the light source gets small. This can cause an inaccurate brightness profile for halos around bright lights close to RBFs (top row of Fig. 7). Also, our model fails to obtain the well-known effect of light shafts emanating from a break in dense media (bottom row of Fig. 7, containing a tube-shaped "cloud" pierced by a small hole). Such light shafts can cause an arbitrarily sharp transition in brightness along the view ray which is missed by our approach. The phenomenon requires abrupt changes in the medium's density and disappears when the medium is smooth.

Our use of optical depth averaging can also cause inaccuracies. For example, if a dense cloud is on the opposite side of a receiver point from the light source, then averaging optical depth in all directions (equation 27) will attenuate the radiance more than it should. Good results are obtained in smooth media, such as a patchy fog. Averaging optical depth in a single direction (for point light sources) is an even more robust approximation.

8 Conclusion

Representing complex spatial variation in scattering media is critical for realistic smoke and fog; ours is the first analytic approach capable of rendering such media in real-time. Our new approximation captures many scattering effects, including glows around light sources and shadowing of the media onto itself as well as softening of shading and shadows on surfaces immersed within the media. Results accurately match a full scattering simulation for smooth media, and are consistent and plausible in all cases.

Several ideas make this approximation fast enough for real-time rendering while preserving visual accuracy. We assume that variation of the scattering integrand along the view ray is primarily due to variation in the medium density. This leads us to separate contributions from each Gaussian by modeling a light path through its nearest point along the view ray while modeling the attenuation and shadowing effect of all other Gaussians using a light path independent of the view ray. We also average optical depth in the direction to point light sources or in all directions around a receiver point for environmental sources, to apply constant-fog models of surface reflectance to inhomogeneous media. Our rendering method makes available new, realistic content for real-time applications like 3D games, and also supports interactive preview and design of scattering media.

In future work, we are interested in capturing even more scattering effects, including light shafts and multiple-scattering. Our current solution also assumes that surface shadowing effects are local, so that PRT vectors can be dotted with the airlight radiance at each receiver point as a final step. In fact, the shadowing effect of immersed surfaces is more complicated and must be considered in proper depth order along with the media. This remains a challeng-

ing and unsolved problem for real-time rendering.

References

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

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.

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

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

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

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.

LINDE, Y., BUZO, A., AND GRAY, R. M. 1980. An algorithm for vector quantizer desig. *IEEE Transactions on Communications*, 1, 84–95.

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

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

NAKAMAE, E., KANEDA, K., OKAMOTO, T., AND NISHITA, T. 1990. A lighting model aiming at drive simulators. In *Proceedings of SIGGRAPH 90*, 395–404.

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

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2004. Triple product integrals for all-frequency relighting. *ACM Trans. Gr. 23*, 3, 477–487.

NISHITA, T., DOBASHI, Y., AND NAKAMAE, E. 1996. Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proceedings of SIGGRAPH 96*, 379–386.

O'ROURKE, J. 1998. *Computational Geometry in C, Second Edition*. Cambridge University Press, Cambridge, England.

PERLIN, K. 2006. Using Gabor functions to make atmosphere in computer graphics. research note, NYU. http://mrl.nyu.edu/~perlin/experiments/gabor/.

PIGHIN, F., COHEN, J., AND SHAH, M. 2004. Modeling and editing flows using advected radial basis functions. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 223–232.

PREETHAM, A. J., SHIRLEY, P., AND SMITS, B. 1999. A practical analytic model for daylight. In *Proceedings of SIGGRAPH 99*, 91–100.

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.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1992. *Numerical Recipes in C, Second Edition*. Cambridge University Press, Cambridge, England.

REEVES, W. T. 1983. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Trans. Gr. 2*, 2, 91–108.

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 *Proceedings of SIGGRAPH 87*, 293–302.

SAKAS, G. 2004. Fast rendering of arbitrary distributed volume densities. In *Eurographics 90*, 519–530.

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

SNYDER, J. 2006. Code generation and factoring for fast evaluation of low-order spherical harmonic products and squares. Tech. Rep. MSR-TR-2006-53, Microsoft Corporation.

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

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 Trans. Gr. 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.

WILLIS, P. 1987. Visual simulation of atmospheric haze. *Computer Graphics Forum 6*, 1, 35–42.

ZHOU, K., REN, Z., LIN, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2007. Real-time rendering of smoke using compensated ray marching. *accepted with major revisions to ACM Trans. Graph.*.

A Line Integration of a Gaussian

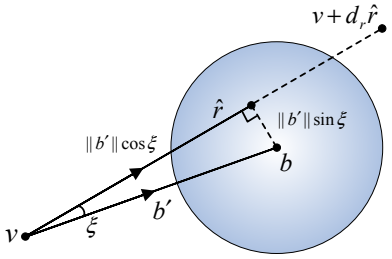


Figure 9: *Line integration of a Gaussian.*

We describe how to integrate a Gaussian

$$G(x) = c\, e^{-a^2 ||x-b||^2}$$

over the ray r in eq. (1). This yields the 1D integral over t given by

$$y = \int_0^{d_r} c\, e^{-a^2 ||x(t)-b||^2}\, dt\,.$$

Letting $b' = b - v$ and $\hat{b}' = b'/||b'||$ where v is the view point, we have

$$\begin{aligned}
y &= \int_0^{d_r} c \exp(-a^2 ||t\,\hat{r} - b'||^2)\, dt \\
&= \int_0^{d_r} c \exp\left(-a^2 \left((t - ||b'|| \cos\xi)^2 + ||b'||^2 \sin^2\xi\right)\right) dt \\
&= c\, e^{-a^2 ||b'||^2 \sin^2\xi} \int_0^{d_r} \exp\left(-a^2 (t - ||b'|| \cos\xi)^2\right) dt \\
&= c\, e^{-a^2 ||b'||^2 \sin^2\xi}\, \frac{\sqrt{\pi}}{2a} \left(\operatorname{erf}(a(d_r - ||b'|| \cos\xi)) - \operatorname{erf}(a(-||b'|| \cos\xi))\right) \qquad (31)
\end{aligned}$$

where ξ is the angle between $\hat{r}$ and $\hat{b}'$, $\cos\xi = \hat{r} \cdot \hat{b}'$, and $\sin\xi = ||\hat{r} \times \hat{b}'||$.

The *error function*, denoted $\operatorname{erf}(x)$, is a standard mathematical function whose numerical evaluation can be found in many published works, e.g. [Press et al. 1992]. We use a fast Chebyshev approximation given by

$$\operatorname{erf}(x) \approx \begin{cases} \operatorname{sgn}(x), & |x| > 2.629639 \\ \begin{pmatrix} 0.0145688 z^6 - 0.0348595 z^5 \\ +0.0503913 z^4 - 0.0897001 z^3 \\ +0.156097 z^2 - 0.249431 z \\ +0.533201 \end{pmatrix} x, & |x| \le 2.629639 \end{cases}$$

where $z = 0.289226 x^2 - 1$. This approximation has absolute error less than 2×10^{-4} for all x.

A similar method for line integration through a Gaussian is described in [Stam 1995a], though it uses a less accurate, piecewise-linear approximation for $\operatorname{erf}(x)$.