Optical Models for Direct Volume Rendering

Nelson Max

University of California, Davis, and

Lawrence Livermore National Laboratory

Abstract

This tutorial survey paper reviews several different models for light interaction with volume densities of absorbing, glowing, reflecting, and/or scattering material. They are, in order of increasing realism, absorption only, emission only, emission and absorption combined, single scattering of external illumination without shadows, single scattering with shadows, and multiple scattering. For each model I give the physical assumptions, describe the applications for which it is appropriate, derive the differential or integral equations for light transport, present calculations methods for solving them, and show output images for a data set representing a cloud. Special attention is given to calculation methods for the multiple scattering model.

1. Introduction

A scalar function on a 3D volume can be visualized in a number of ways, for example by color contours on a 2D slice, or by a polygonal approximation to a contour surface. Direct volume rendering refers to techniques which produce a projected image directly from the volume data, without intermediate constructs such as contour surface polygons. These techniques require some model of how the data volume generates, reflects, scatters, or occludes light. This paper presents a sequence of such optical models with increasing degrees of physical realism, which can bring out different features of the data.

In many applications the data is sampled on a rectilinear grid, for example, the computational grid from a finite difference simulation, or the grid at which data are reconstructed from X-ray tomography or X-ray crystallography. In other applications, the samples may be irregular, as in finite element or free lagrangian simulations, or with unevenly sampled geological or meteorological quantities. In all cases, the data must be interpolated between the samples in order to use the continuous optical models described here. For example, linear interpolation can be used on tetrahedra, and trilinear or tricubic interpolation can be used on cubes. A number of other interpolation methods are given in Nielson and Tvedt [1]. Here I will just assume the interpolation is done somehow to give a scalar function $f(X)$ defined for all points X in the volume.

Optical properties like color and opacity can then be assigned as functions of the interpolated value $f(X)$. (The physical meaning of these optical properties will be discussed in detail below.)

Interpolating f first permits the optical properties to change rapidly within a single volume element, to emphasize a small range of scalar values. It is possible to compute the optical properties only at the grid vertices, and then interpolate them instead, but this may eliminate fine detail. The situation is analogous to the superiority of Phong shading (interpolating the normal) over Gouraud shading (interpolating the shaded color) for representing fine highlight detail.

To compute an image, the effects of the optical properties must be integrated continuously along each viewing ray. This does not mean that only ray tracing can be used. Mathematically equivalent integration can be performed with polyhedron compositing (Shirley and Tuchman [2], Max *et al*. [3], Wilhelms and van Gelder [4], Williams and Max [5]). If the integral is approximated by a Riemann sum, as discussed below, then the plane-by-plane compositing methods of Dreben *et al*. [6] and Westover [7] can also produce equivalent approximations. In this paper, I will not be concerned with the distinctions between these methods. Instead, I will deal with the mathematical forms that the continuous integral takes, depending on the optical model. Siegel and Howell [8] is a good general reference for the physics behind these models.

The optical properties which affect the light passing through a “participating medium” are due to the absorption, scattering, or emission of light from small particles like water droplets, soot or other suspended solids, or individual molecules in the medium. For the models below, I will describe the geometric optics effects of the individual particles, and then derive a differential equation for the light flow in the medium. The differential equations are for a continuous medium, in the limit where the particles are infinitesimally small, so that the absorption, emission, and scattering take place at every infinitesimal segment of the ray. I will write the equations taking the intensity and optical properties to be scalars, for black-and-white images. For multiple wavelength bands (*e. g*. red, green, and blue) in a color image, the equations are repeated for each wavelength, so these quantities become vectors.

2. Absorption only

The simplest participating medium has cold perfectly black particles which absorb all the light that they intercept, and do not scatter or emit any. For simplicity, assume that the particles are identical spheres, of radius r and projected area $A = \pi r^2$, and let ρ be the number of particles per unit volume. Consider a small cylindrical slab with a base B of area E, and thickness Δs, as shown in figure 1, with the light flowing along the direction Δs, perpendicular to the base. The slab has volume $E\Delta s$, and thus contains $N = \rho E \Delta s$ particles. If Δs is small enough so that the particle projections on the base B have low probability of overlap, the total area that they occlude on B is approximated by $NA = \rho AE\Delta s$. Thus the fraction of the light flowing through B that is occluded is $\rho AE\Delta s/E = \rho A\Delta s$. In the limit as Δs approaches zero, and the probability of overlap approaches zero also, this gives the differential equation

$$\frac{dI}{ds} = -\rho(s) A I(s) = -\tau(s) I(s) \tag{1}$$

where s is a length parameter along a ray in the direction of the light flow, and $I(s)$ is the light intensity at distance s. The quantity $\tau(s) = \rho(s)A$ is called the extinction coefficient and defines the rate that light is occluded. The solution to this differential equation is

$$I(s) = I_0 \exp\left(-\int_0^s \tau(t)\, dt \right), \tag{2}$$

where I_0 is the intensity at $s = 0$, where the ray enters the volume. The quantity

$$T(s) = \exp\left(-\int_0^s \tau(t)\, dt \right) \tag{3}$$

is the transparency of the medium between 0 and s. A somewhat different derivation of these equations is given in Blinn [9]. (See also section 2 of Williams and Max [5].)

In the volume rendering literature the extinction coefficient τ is often simply called opacity. However, the opacity α of a voxel of side l, viewed parallel to one edge, is actually

$$\alpha = 1 - T(l) = 1 - \exp\left(-\int_0^l \tau(t)\, dt \right),$$

or, if τ is constant inside the voxel, $\alpha = 1 - \exp(-\tau l) = \tau l - (\tau l)^2/2 + \ldots$. This distinction is important if the voxel is scaled to a different size, or is viewed diagonally, so that the path length through it is different from l. Wilhelms and Van Gelder [4] have a user interface in which α is specified for a unit length l, allowing τ to become infinite when $\alpha = 1$. They also suggest that for small voxels, α can be approximated by $\min(1, \tau l)$, which truncates all but the first term of the above series, but makes sure that α never exceeds 1. Max[10] suggests a quadratic approximation for $\alpha(l)$ arranged to meet the line $\alpha = 1$ smoothly at $l = 2/\tau$.

The mapping which assigns a value for an optical property like τ to each value of the scalar f being visualized is called a transfer function. The simplest transfer function assigns $\tau = \infty$ if f exceeds a threshold K, and $\tau = 0$ otherwise. Thus if f is tomographic density on a medical data set, K can be chosen to make bone completely opaque, and all other tissues completely transparent. Many early medical visualizations were produced in this way. If the "interpolation" for $f(x)$ just sets f to the value at the nearest sample point, so that it is constant inside cubes centered at the sample points, rendering consists of merely projecting the opaque cubes, or their front faces, onto the image plane. This can be done using a z-buffer or back-to-front "painter's algorithm" for visibility determination. If a list of surface voxels can be found [11], the interior voxels, which are guaranteed to be hidden, need not be projected.

This technique has been useful in visualizing bone or other tissues for medical purposes, and various shading schemes have been developed [12]. The simplest uses only the z-buffer depth at a pixel, shading the more distant pixels darker. More realistic surface shading models require the surface normal vector, which must be estimated. One method uses the z-buffer values at neighboring pixels to estimate the slope of the projected surface. Surface normals can also be estimated before projection, according to the orientation of the cube face and its neighbors in the boundary of the opaque volume. Finally, the normals can be determined from the gradient of $f(X)$, estimated

by finite differences of f at neighboring voxels. Note that these shading functions are applied after the thresholded opaque volume has been determined, and are not the same as the shading for direct volume rendering to be discussed below.

The more sophisticated optical models use the transfer function to specify the extinction coefficient as a finite, continuously varying function $\tau(f)$ of the scalar f. When the integration in equation (2) is carried out along each viewing ray, the result is an X-ray-like image, which accounts for the attenuation of the X-rays from the source at $s = 0$ by the density between the source and the film plane. If f is the X-ray absorption density reconstructed by Computed Tomography, $\tau(f)$ can be taken as the identity function, to produce a new simulated X-ray image from any specified direction. Other assignments of $\tau(f)$ can isolate a density range of interest, and render all other densities as transparent. Such images can be used for medical diagnosis and nondestructive testing.

Alternatively, I_0 can represent a background intensity, varying from pixel to pixel, and the resulting image represents the volume density as a cloud of black smoke obscuring the background. To illustrate the optical models in this paper, I have modelled an atmospheric cloud as a sum of ellipsoidal densities. I have added a 3-D noise texture from Perlin [13], to give a natural fractal appearance to its edges. Fig. 2 shows this cloud represented with equation (2), as black smoke hiding the ground, an aerial phhoto of Washington, DC.

The problems of computing these X-ray like images are basically those of computing the integrals appearing in equations (2) and (3), since the exponential function need be done only once per output pixel, and can even be performed by a “gamma correction” table lookup as part of the video output process. Malzbender [14] and Totsuka and Levoy [15] have shown how to use the fourier projection slice theorem and fast fourier transforms to compute these integrals very rapidly.

3. Emission only

In addition to extinction, the medium may also add light to the ray by emission or reflection of external illumination. The simplest case is emission, as by hot soot particles in a flame. Of course, real particles absorb as well as emit light, but in the limit as the particle size or number density approaches zero, while the emission goes to infinity in a compensating manner, we can neglect the absorption. This is the case for a very hot tenuous gas, which glows but is almost transparent. In this section, we will model this case by assuming the small spherical particles discussed above are transparent, and then in the next section we will include their absorption.

If the particles in fig. 1 are transparent, but glow diffusely with an intensity C per unit projected area, their projected area $\rho AE\Delta s$ derived above will contribute a glow flux $C\rho AE\Delta s$ to the base area E, for an added flux per unit area $C\rho A\Delta s$. Thus the differential equation for $I(s)$ is

$$\frac{dI}{ds} = C(s)\,\rho(s)\,A = C(s)\,\tau(s) = g(s) \tag{4}$$

The term $g(s)$ is called the source term, and later we will let it include reflection as well as emission. The solution to this differential equation is simply

$$I(s) = I_0 + \int_0^s g(t)\,dt. \tag{5}$$

Fig. 3 shows the cloud of fig. 2 drawn in this way, with g proportional to f. Note that the fourier methods of Malzbender [14] can also be used to produce such images, which are like the X-ray negatives commonly viewed by radiologists. This sort of image is useful for simulating the glow from fluorescent stains in reconstructed micrographs, or in any situation when the glowing material is not too extensive. However, unlike the exponentials in equations (2) and (3), the integral in (5) has no upper bound, because the intensity can be added across an arbitrary thickness without attenuation. The accumulated intensity can easily exceed the representable range of the output devise. The cloud in figure 3 is too tenuous at the edges because I had to set the constant C very small in order not to exceed the available intensity range at the center of the cloud.

4. Absorption plus emission

The particles in an actual cloud occlude incoming light, as well as add their own glow. Thus a realistic differential equation should include both source and attenuation terms:

$$\frac{dI}{ds} = g(s) - \tau(s)\,I(s)\,. \tag{6}$$

In section 4.2 we will consider as a special case the model where $g(s) = C(s)\,\tau(s)$, as in equation (4), but for now, let the source term $g(s)$ be an arbitrary function of position, perhaps specified by an independent transfer function $g(f)$. The absorption plus emission model is useful for volume rendering continuous scalar fields from numerical simulations, or medical data that has been segmented into different tissue types which can be given different values of τ and g.

Equation (6) can be solved by bringing the $\tau(s)\,I(s)$ term to the left hand side, and multiplying by the integrating factor $\exp\left(\int_0^s \tau(t)\,dt\right)$, giving

$$\left(\frac{dI}{ds} + \tau(s)\,I(s)\right)\exp\left(\int_0^s \tau(t)\,dt\right) = g(s)\exp\left(\int_0^s \tau(t)\,dt\right)$$

or

$$\frac{d}{ds}\left(I(s)\exp\left(\int_0^s \tau(t)\,dt\right)\right) = g(s)\exp\left(\int_0^s \tau(t)\,dt\right).$$

Integrating from $s = 0$ at the edge of the volume to $s = D$ at the eye, we get

$$I(D)\exp\left(\int_0^D \tau(t)\,dt\right) - I_0 = \int_0^D\left(g(s)\exp\left(\int_0^s \tau(t)\,dt\right)\right)ds\,.$$

Bringing the I_0 to the other side, and multiplying by $\exp\left(-\int_0^D \tau(t)\,dt\right)$, we can solve for $I(D)$:

$$I(D) = I_0\exp\left(-\int_0^D \tau(t)\,dt\right) + \int_0^D g(s)\exp\left(-\int_s^D \tau(t)\,dt\right)ds\,. \tag{7}$$

The first term represents the light coming from the background, multiplied by the cloud's transparency, as in equation (2). The second term is the integral for the contribution of the source term $g(s)$ at each position s, multiplied by the transparency $T'(s) = \exp\left(-\int_s^D \tau(x)\,dx\right)$ between s and the eye. Thus

$$I(D) = I_0 T(D) + \int_0^D g(s)\,T'(s)\,ds\,.$$

4.1 Calculation methods

For certain transfer functions and interpolation methods, the integrals in equation (7) can be calculated analytically, as will be discussed in section 4.2. For more general cases, however, numerical integration is required. The simplest numerical approximation to an integral $\int_0^D h(x)\,dx$ is the Riemann sum $\sum_{i=1}^{n} h(x_i)\,\Delta x$. The interval from 0 to D is divided up into n equal segments, of length $\Delta x = D/n$, and a sample x_i is chosen in each segment, so that $(i\text{-}1)\Delta x \le x_i \le i\Delta x$. To simplify the following formulas, I will choose $x_i = i\Delta x$. Then $\exp\left(-\int_0^D \tau(x)\,dx\right)$ is approximated by

$$\exp\left(-\sum_{i=1}^{n} \tau(i\Delta x)\,\Delta x\right) = \prod_{i=1}^{n} \exp\left(-\tau(i\Delta x)\,\Delta x\right) = \prod_{i=1}^{n} t_i$$

where $t_i = \exp(-\tau(i\Delta x)\,\Delta x)$ can be thought of as the transparency of the i^{th} segment along the ray. As noted above, t_i depends not only on $\tau(f)$ but also on the ray segment length Δx.

Similarly, for the final integral in equation (7), we can let $g_i = g(i\Delta t)$, and approximate the transparency $\exp\left(-\int_{i\Delta x}^D \tau(x)\,dx\right)$ between x_i and D by $\prod_{j=i+1}^{n} t_j$. The Riemann sum for $\int_0^D g(s)\exp\left(-\int_s^D \tau(x)\,dx\right)ds$ then becomes $\sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j$. Thus the final estimate is

$$\begin{aligned} I(D) &\approx I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j \\ &= g_n + t_n\left(g_{n-1} + t_{n-1}\left(g_{n-2} + t_{n-2}\left(g_{n-3} + \ldots\left(g_1 + t_1 I_0\right)\ldots\right)\right)\right). \end{aligned}$$

This gives the familiar back-to-front compositing algorithm:

```
I = I0;
for( i = 1; i <= n; ++i )
   I = t[i]*I + g[i];
```

or the front-to-back compositing algorithm:

```
I = 0.;
T = 1.;
i = n;
while( T > small_threshold && i >= 1 )
   {
   I = T*I + g[i];
   T = T*t[i];
   --i;
   }
I = I + T*I0;.
```

4.2 The Particle Model

The derivation of $g(s)$ in equation (4), based on identical spherical particles, defines $g(s) = C(s)\,\tau(s)$. A particularly simple case is when C is constant along the ray, or at least along the segment within a certain material region assigned the color C. This makes the second integral in equation (7) much simpler:

$$\begin{aligned}\int_0^D g(s)\exp\left(-\int_s^D \tau(t)\,dt\right)ds &= \int_0^D C\tau(s)\exp\left(-\int_s^D \tau(t)\,dt\right)ds \\ &= C\int_0^D \frac{d}{ds}\exp\left(-\int_s^D \tau(t)\,dt\right)ds \\ &= C\left(1. - \exp\left(-\int_0^D \tau(t)\,dt\right)\right)\end{aligned}$$

Making the above substitution in equation (7) and using the total transparency $T(D)$ from equation (3), we get

$$I(D) = I_0 T(D) + C(1. - T(D))\,. \tag{8}$$

This is the simple compositing of the color C on top of the background I_0, using the transparency $T(D)$. Conceptually, the opacity $\alpha = (1. - T(D))$ represents the probability that a ray from the eye will hit a particle, and "see" color C.

If $I_0 = 0$, and τ is proportional to f, the result is like an X-ray negative, brightest where there is most density, but saturating at the maximum intensity C, as shown in fig. 4. Fig. 5 shows the cloud over the ground, according to equation (8).

Instead of constant C, a somewhat more general assumption is that $C(s)$ and $\tau(s)$ vary linearly

along a ray segment. This will be the case if both $C(f)$ and $\tau(f)$ are linear or piecewise linear functions of the scalar field f, and if f is interpolated linearly across tetrahedral cells joining points where f is sampled. In this case, $g(s)$ will be a quadratic function on a ray segment, and Williams and Max [5] give a rather complicated analytic formula for the integral, involving tables or subroutines for the normal error integral erf(x), and for $\int_0^x \exp t^2 dt$.

The particle model corresponds to a physical situation with glowing particles, but sometimes it is convenient to define $g(s)$ independently of $\tau(s)$. For example, g could be given directly in terms of the scalar field f, or even in terms of a different scalar field unrelated to the one determining τ. This gives slightly more flexibility than the particle model, because it allows g to be non-zero even when τ is zero, permitting completely transparent glowing gas, without needing an infinitely bright $C(s)$. Even with non-zero τ, it will have different interpolation properties. For example, in the situation in the previous paragraph, the interpolated $g(s)$ was quadratic on a ray segment, while an independently defined and interpolated $g(s)$ would be linear.

5. Scattering and Shading

The next step toward greater realism is to include scattering of illumination external to the voxel. In the simplest model, sometimes called the "Utah approximation" after early shaded images from the University of Utah, the external illumination is assumed to reach the voxel from a distant source, unimpeded by any intervening objects or volume absorption. We will consider this case first, and deal with shadows in the next section.

A general shading rule for the scattered light $S(X, \omega)$ at position X in direction ω is

$$S(X, \omega) = r(X, \omega, \omega') \, i(X, \omega') \tag{9}$$

where $i(X, \omega')$ is the incoming illumination reaching X flowing in direction ω', and $r(X, \omega, \omega')$ is the bidirectional reflection distribution function, which depends on the direction ω of the reflected light, the direction ω' of the incoming light, and on other properties like f or its gradient that vary with position X. For light scattered by a density of particles,

$$r(X, \omega, \omega') = a(X) \, \tau(X) \, p(\omega, \omega'),$$

where $a(X)$ is the particle albedo, giving the fraction of the extinction τ which represents scattering rather than absorption, and p is the phase function, which specifies the directionality of the scattering. For spherical particles, or randomly oriented particles of any shape, p will depend only on the angle α between ω and ω', *i. e.*, on $x = \cos\alpha = \omega \cdot \omega'$. A common formula for p, which can approximate the Mie scattering for spherical particles comparable in size to the light wavelength, is the Henyey-Greenstein function [16]

$$p(\omega, \omega') = \frac{1}{4\pi} \frac{1 - c^2}{\left(1 + c^2 - 2cx\right)^{3/2}}. \tag{10}$$

Here c is an adjustable constant between -1 and 1, which is positive for forward scattering, negative for backward scattering, and zero for isotropic scattering, which is equal in all directions. An even simpler formula (see Blinn[9]) can be derived by geometric optics for a spherical particle much larger than the light wavelength, whose surface scatters diffusely by Lambert's law:

$$p(\omega, \omega') = (8/3\pi)\,(\sin\alpha + (\pi-\alpha)\cos\alpha).$$

In volume rendering, one often wants to produce the visual effect of a shaded contour surface, without actually constructing surface polygons. One can then claim to be rendering directly from the actual data, without introducing artifacts from polygonalization. Such shading is a special case of a general volume scattering term $S(X, \omega)$, and requires the contour surface normal N, which is equal to the direction of the gradient ∇f, *i. e.*, $N = \nabla f / |\nabla f|$. The gradient ∇f can be estimated by central differences between regularly spaced gridded data values, and then interpolated between the grid vertices.

To simulate shading effects from contour surfaces at sharp changes in the scalar function f, one could use $|\nabla f|$ to measure surface "strength". Then a simple Lambert diffuse shading formula $\max(N \cdot \omega', 0)$, multiplied by the "strength", gives

$$r\,(X, \omega, \omega') \;=\; \max\,(\nabla f \cdot \omega', 0)\;. \tag{11}$$

More sophisticated formulas involving ω, ω', and N can be used, like Phong or Cook-Torrance shading. One can also make the "strength" depend on f, in order to localize the surface shading near a contour value for f. Details of such shading algorithms can be found in Levoy [17] and Dreben *et al*. [6]. The most general source term $g(X, \omega)$ is the sum of a non-directional internal glow or emissivity $E(X)$ as in section 3, and the reflection or scattering term $S(X, \omega)$ of this section:

$$g\,(X, \omega) \;=\; E\,(X) \;+ S\,(X, \omega)\;. \tag{12}$$

6. Shadows

The shading effects discussed above are unrealistic, since they replace an internal glow by a reflection of external illumination, but take no account of shadows. If $g(X)$ is to model the reflections from surfaces or particles, one should account for the transparency of the volume density between the light source and the point X, as well as from X to the viewpoint. If L is the intensity from an infinite light source in the direction $-\omega'$, the illumination $i(X, \omega')$ which reaches X is

$$i\,(X, \omega') \;=\; L\,\exp\!\left(-\int_0^\infty \tau\,(X - t\omega')\,dt\right). \tag{13}$$

In practice, the integral does not run to ∞, but only to the edge of the data volume.

At this point, it is convenient to reverse the meaning of the parameter s in equation (7), so that it starts at a viewpoint X and goes out in direction $-\omega$ opposite to the light flow, reaching

X - $s\omega$ at distance s. Rewriting equation (7) with this reversed ray parametrization, $s' = D - s$, $t' = D - t$, we have

$$I(X) = I_0 \exp\left(-\int_0^D \tau(X - t'\omega)\, dt' \right) + \int_0^D g(X - s'\omega) \exp\left(-\int_0^{s'} \tau(X - t'\omega)\, dt' \right) ds' . \qquad (14)$$

Removing the primes on s and t, and substituting equations (9) and (13), we get

$$I(X) = I_0 \exp\left(-\int_0^D \tau(X - t\omega)\, dt \right) + \int_0^D r(X - s\omega, \omega, \omega')\, L \exp\left(-\int_0^\infty \tau(X - sw - t\omega')\, dt \right) \exp\left(-\int_0^s \tau(X - t\omega)\, dt \right) ds$$

The factor $\exp\left(-\int_0^\infty \tau(X - sw - t\omega')\, dt \right)$ corresponds to the "shadow feelers" used in recursive ray tracing, except that a shadow feeler is sent to the light source at each point X - $t\omega$ along the primary ray, and returns a fractional transparency. In Max [10] and [18] I show how these integrals can be evaluated under particular conditions, for example, when τ is constant or varies only along one dimension. Kaneda *et al*. [19] also describe a case of one-dimensional variation, and Nishita *et al*. [20] consider multiple light sources when τ is constant and opaque polygonal objects are present.

A more general two-pass numerical algorithm was suggested by Kajiya and Von Herzen [21]. The first pass computes the illumination $i(X, \omega)$ reaching X, as in equation (2). It propagates the flux from the light source through the volume, one voxel layer at a time, and accounts for the transparency of the layer before propagating to the next one. In the second pass, this illumination is reflected or scattered to the viewpoint by a shading rule $g(X, \omega) = r(X, \omega, \omega')\, i(X, \omega')$. The reflected intensity $g(X, \omega)$ is then gathered along viewing rays according to equation (7). Figure 6 shows the cloud rendered in this way. The shading used a Henyey-Greenstein phase function p as in equation (10), with a peak in the forward scattering direction, consistent with the light scattering properties of small water droplets.

Shadows give useful cues about the shape of opaque objects, and are necessary for photorealistic rendering of opaque objects in the presence of smoke, fog, clouds, turbid water, and other "participating media". The two pass method takes only twice as long as the gathering pass without shadows, and the illumination pass can be amortized over several animated frames if only the viewpoint moves. The "Heidelberg ray tracing model" of Meinzer *et al.* [22] systematically applies this two pass method to medical images. However, its utility in general volume rendering applications has not yet been demonstrated.

7. Multiple Scattering

This two pass method is a single-scattering model, because it accounts for only one reflection or scattering event from the illumination ray to the observer. It is only valid if the albedo or the density is low, so that multiple scattering is unlikely. This is not usually the case in atmospheric clouds, so the side of the cloud away from the light source looks unnaturally dark in figure 6. To

correct this and account for multiple scattering, one may apply the "radiosity" methods originally developed in the field of thermal radiation heat transport. Multiple scattering calculations are important for realistic rendering of high albedo participating media, but are expensive in computer time, and are overkill for most scientific visualization applications.

Multiple scattering involves directionally dependent light flow, so it is necessary to find $I(X, \omega)$, the intensity at each point X in each light flow direction ω. The point at distance s along the viewing ray from X, opposite to the light flow, is $X - s\omega$. Integrating the scattering at $X - s\omega$ of light from all possible incoming directions ω' on the 4π unit sphere, the added scattered intensity gives the source term

$$g(s, \omega) = \int_{4\pi} r(X - s\omega, \omega, \omega') I(X - s\omega, \omega') d\omega' .$$

Substituting this into equation (14) gives

$$I(X, \omega) = I_0(X - D\omega, \omega) \exp\left(-\int_0^D \tau(X - t\omega) dt \right) + \int_0^D \left(\int_{4\pi} r(X - s\omega, \omega, \omega') I(X - s\omega, \omega') d\omega' \right) \exp\left(-\int_0^s \tau(X - t\omega) dt \right) ds , \quad (15)$$

where $X - D\omega$ is the point at the edge of the volume density, reached by the ray from X in direction $-\omega$, and $I_0(X - D\omega, \omega)$ is the external illumination there flowing in direction ω.

7.1 The zonal method

Note that the unknown $I(X, \omega)$ appears on both sides of this integral equation, making its solution more difficult. The situation is simplifies slightly if the scattering is isotropic, so that $g(X,\omega)$ depends only on X. In this case the method of diffuse radiosity for interreflecting surfaces can be extended to volumes. Rushmeier and Torrance [23] and Hottel and Sarofim[24] call this the zonal method, and assume that g(X) is piecewise constant on volume elements. These will usually be the voxels in a volume rendering application. For simplicity, I will assume their volumes are the unit volume.

The total contribution of all voxels X_j to the isotropic scattering $S(X_i)$ at X_i is

$$S(X_i) = a(X_i) \sum_j F_{ij} g(X_j)$$

where the "form factor" F_{ij} represents the fraction of the flux originating at voxel X_j that is intercepted by voxel X_i, and the albedo $a(X_i)$ is the portion of this intercepted flux that is scattered. Rushmeier and Torrance [23] also consider the scattering from surfaces, but for rendering an isolated volume, it is convenient to propagate the external illumination as in the first pass of Kajiya and Von Herzen [21], and include the first bounce of external illumination in the emissivity $E(X_i)$ at X_i. Using equation (12), this then gives a system of simultaneous linear equations for the unknowns $g(X_i)$:

$$g(X_i) = E(X_i) + a(X_i) \sum_j F_{ij} g(X_j) . \quad (16)$$

The form factor F_{ij} is actually a 7 dimensional integral over the voxels X_i, X_j, and the rays between them. For each pair of points, one in X_i and one in X_j, the transparency along the ray between them must be found as in equation (3) by integrating τ across the intervening voxels. Rushmeier approximates this 7D integral using a single 1D integral along the ray between the voxel centers. If a cubic data volume is n voxels on a side, there are O(n) intervening voxels along this ray, and a total of n^3 voxels, so it takes time O(n^7) to compute the O(n^6) necessary form factors. Iterative methods, computing one scattering bounce for each of the O(n^6) form factors per iteration, can converge in O(1) iterations if the albedos $a(X_i)$ are bounded by a constant $r < 1$. Thus the computation time is dominated by the O(n^7) cost of determining the form factors.

Rushmeier combined this volume-to-volume scattering with the earlier surface-to-surface scattering of Goral *et al*. [25], adding surface-to-surface, surface-to-volume, and volume-to-surface terms to equation (16). Sobierajski [26] further generalized this method to include terms from voxels shaded according to equation (11), which scatter diffusely into a hemisphere instead of a full sphere. Hanrahan *et al*. [27] have proposed a hierarchical method to group surface-to-surface interactions to reduce the number of form factors from O(N^2) to O(N), where N is the number of total elements, so that $N = n^3$ in the cubic volume case. Bhate [28] and Sobierajski [26] have extended these hierarchies to volume scattering.

Once the source terms $g(X_i)$ have been determined, equation (7) can be used to produce an output image from any desired viewpoint, with any desired camera parameters. In this pass, the presumed constant voxel values $g(X_i)$ can be interpolated to give a smoother rendering. This final view-dependent pass is also used in surface radiosity algorithms.

7.2 The Monte Carlo method

For directional scattering with a non-isotropic phase function, $g(X, \omega)$ depends on the scattering direction ω, and it is easier to deal directly with equation (15), where the unknown is $I(X, \omega)$. There are three popular methods for solving this integral equation, all explained in Siegel and Howell [8].

The first is the Monte Carlo method, originally developed by physicists for neutron transport, and applied to rendering surface interreflection by Cook *et al*. [29] and Kajiya [30], and to volume applications by Rushmeier [31]. Sample rays are traced from the eye through a pixel, and undergo random absorption or scattering, with probabilities based on the extinction coefficient τ, the albedo a, and the phase function p. Those rays that end up at a light source or volume emitter contribute flux to the pixel intensity. Since these contributing rays are in general a small fraction of all those considered, and many ray samples are required to decrease the variance of the mean of their contributions, the resulting images tend to appear noisy and/or take a very long time to compute.

Rushmeier [31] suggested calculating the $g(X, \omega)$ by the zonal method and then doing the final rendering pass using the Monte Carlo method, for one extra directional bounce toward the

viewpoint. Shirley [32] and Chen *et al*. [33] also use such a final Monte Carlo bounce in rendering images of interreflecting surfaces. These two references, and also Heckbert [34], propose "caustic texture maps" to capture directional interreflection propagated by Monte Carlo means from the light sources, and contributing to the final rendering pass. Thus rays propagating from the lights and rays propagating from the eye meet in the middle at the caustic map. This partially solves the problem that rays originating from the eye rarely end up at a light source, while rays from a light source rarely end up at the eye. For volume rendering, Blasi *et al*. [35] used a similar approach, analogous to the two pass algorithm in [21]. In the first Monte Carlo pass, light was propagated from the light sources, and any light scattering at voxel X_i was added to a texture map, which was used in a final rendering pass with rays from the eye, using equation (7). Blasi *et al*. only stored an isotropic scattering texture, but their methods could be generalized to store a directionally scattered texture $I(X_i, \omega)$.

7.3 The P-N method

The second method, called the P-N, or P_N method in thermal engineering, was originally developed by Chandrasekhar [36] for stellar atmospheres, and was applied to computer graphics by Kajiya and Von Herzen [21]. At each point X, it expands $I(X, \omega)$ in spherical harmonics in the unit sphere direction ω, getting a coupled system of partial differential equations for the spherical harmonic expansion coefficients, which can be solved by finite difference methods.

7.4 The discrete ordinates method

The third alternative is the discrete ordinates method, which uses a collection of M discrete directions, chosen to give optimal Gaussian quadrature in the integrals over a solid angle. Lathrop [37] points out that this process produces ray effects, because it is equivalent to shooting the energy from an element in narrow beams along the discrete directions, missing the regions between them. He presents modifications to avoid these ray effects, but the resulting equations are mathematically equivalent to the P-N method. This implies that M properly placed directions specify the directional intensity distribution to the same detail as M spherical harmonic coefficients. Languénou *et al*. [38] have applied the discrete ordinates method to volume rendering images of clouds.

If the volume is divided into $N = n^3$ cubical voxels, there are a finite number NM of unknown intensities in the discrete ordinates method. These are related by a system of linear equations, whose coefficients are the form factors F_{klij} for $i, k, = 1, \ldots, N$, and $j, l = 1, \ldots, M$. As shown in fig. 7, F_{klij} represents the effect of the intensity $I(X_i, \omega_j)$ in direction ω_j at voxel X_i, on the intensity $I(X_k, \omega_l)$ in direction ω_l at voxel X_k, taking into account the extinction between the voxels. In order to reduce the ray effect, it is necessary to spread the intensity $I(X_i, \omega_j)$ into the solid angle in a direction bin about ω_j, instead of along a discrete ray. Thus every voxel can propagate flux to every other voxel through at least one direction bin.

The flux $I(X_i, \omega_j)$ can hit voxel X_k only if there is a ray in the direction bin ω_j connecting a point in X_i to a point in X_k. For distant pairs of voxels, this is usually only possible for one direction bin ω_j, and even at the bin corners, it is possible for at most four. Thus for fixed i, the M fluxes $I(X_i, \omega_j)$ affect only O(N) other voxels X_k. As in Rushmeier's method, one must compute for each pair of voxels X_i and X_k, an integral for the transparency across the O(n) voxels along the line between their centers (line *CD* in figure 7). Once the flux reaches voxel X_k, it is scattered to each of the reflected directions ω_l, using an M x M bin-to-bin matrix version of the phase function $p(\omega_j, \omega_l)$. This gives O(N^2M) non-zero coefficients, costing O($n^7 + n^6M$) time. As in the case of glossy surface radiosity studied by Immel *et al.* [39], the matrix F_{klij} is sparse, and sparse solution methods apply. Aupperle and Hanrahan [40] have shown that the hierarchical methods of [27] can be applied to glossy surfaces, and presumably they could also be applied to anisotropic volume scattering.

I have found a way to approximate the effects of the coefficients F_{kijl} as the flux in direction bin ω_j propagates from voxel to voxel in the volume. Basically, the flux entering each voxel is multiplied by the voxel's transparency and then distributed to four adjacent voxels, determined by the direction bin ω_j. Since this arithmetic is independent of the location of the shooting voxel X_i, the flux from all voxels X_i in a layer can be propagated simultaneously, effectively computing N^2 interactions in time O(N log$N$). (See Max[41] for details.)

When the flux reaches a voxel X_k, it is deposited into a temporary array of received flux. After the flux in direction bin ω_j from all layers is received at voxel X_k, it is scattered to the M direction bins ω_l, using a row from the M x M matrix version of the scattering phase function. This takes time O(MN). Thus one iteration through all M shooting bins ω_j takes time O(MN log$N$ + M^2N) = O(Mn^3log$n$ + M^2n^3). These iterations must be repeated until convergence, but when the number of iterations required is small compared to N, this is faster then computing all the coefficients F_{klij} in advance. As in the other radiosity methods, once the light flow distribution $I(X,\omega)$ is approximated, a final gathering pass along viewing rays using the right hand side of equation (15) can be performed quickly from any viewpoint, giving one final directional scattering bounce.

In my implementation, I used direction bins arranged on the 96 exterior faces of a 4x4x4 block of cubes. These bins contain unequal solid angles, but this is taken into account in the definition of the M x M phase function matrix $p(\omega_i, \omega_j)$. Fig. 8 was produced by this method, using the same forward scattering function as in fig. 6. The increase in brightness comes from the higher order scattering. The albedo a was .99, but only 15 iterations were needed for convergence, because much of the flux exited at the edges of the cloud. The cloud density was defined on a 24 x 24 x 18 voxel volume, and each iteration took 15 minutes on an SGI Personal Iris 4D/35 with a Mips 3000 processor. The final rendering, at 512 x 384 pixel resolution, took 5 minutes.

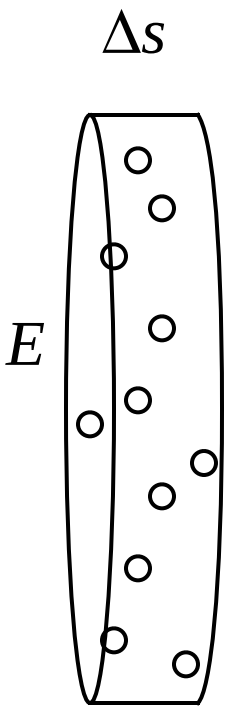

Fig. 1. A slab of base area E and thickness Δs.

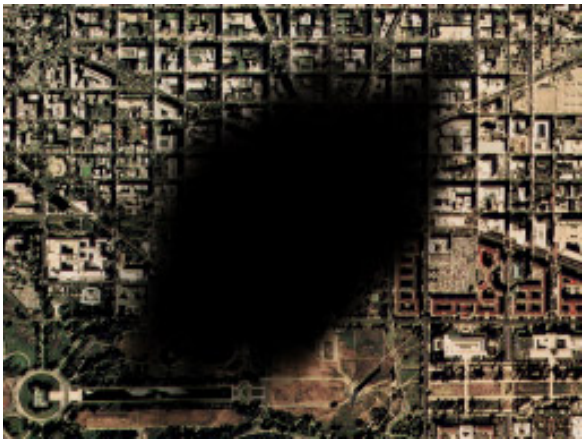

Fig. 2. Black smoke cloud over the ground.

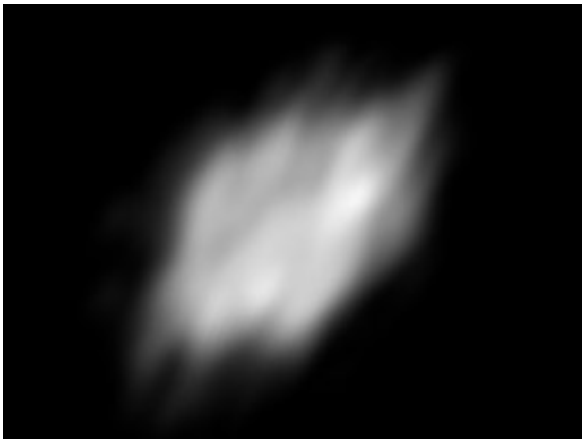

Fig. 3. Emission only cloud.

Fig. 4. Cloud with emission and extinction.

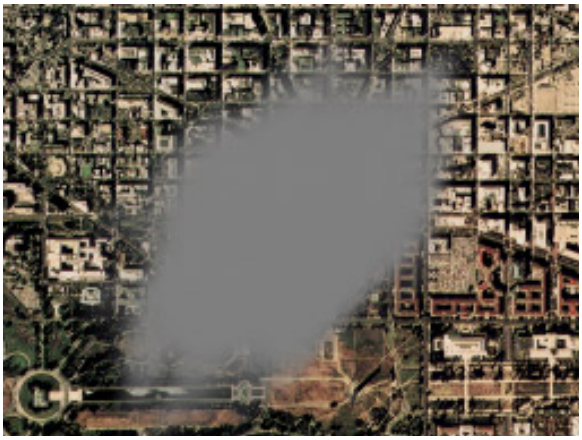

Fig. 5. Cloud of fig. 4 over the ground.

Fig. 6. Cloud with single scattering.

Fig. 8. Cloud with multiple scattering.

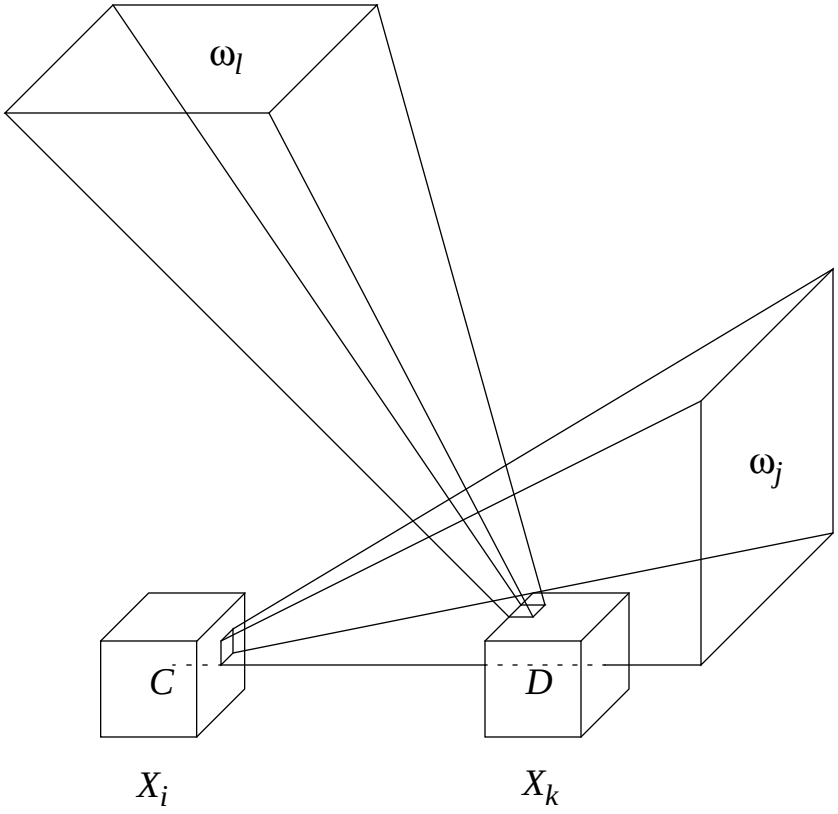

Fig. 7. Geometry for F_{klij} showing direction bin ω_j at pixel X_i and direction bin ω_l at pixel X_k. The flux from X_i to X_k lies in four different direction bins, because X_k is at the corner of bin ω_j.