Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Lattice-Boltzmann Lighting

Robert Geist, Karl Rasche, James Westall [†] and Robert Schalkoff [‡]

Clemson University, Clemson, South Carolina USA 29634

Abstract

A new technique for lighting participating media is suggested. The technique is based on the lattice-Boltzmann method, which is gaining popularity as alternative to finite-element methods for flow computations, due to its ease of implementation and ability to handle complex boundary conditions. A relatively simple, grid-based photon transport model is postulated and then shown to describe, in the limit, a diffusion process. An application to lighting clouds is provided, where cloud densities are generated by combining two well-established techniques. Performance of the new lighting technique is not real-time, but the technique is highly parallel and does offer an ability to easily represent complex scattering events. Sample renderings are included.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Lattice-Boltzmann (LB) methods are computational alternatives to finite-element methods for solving coupled systems of partial differential equations. The LB methods have recently provided significant successes in modeling fluid flows and associated transport phenomena [HCD98]. The methods simulate transport by tracing the evolution of a single particle distribution through synchronous updates on a discrete grid. Although the LB methods deliver stability, accuracy, and computational efficiency comparable to the finite-element methods, the advantages lie in ease of implementation, straightforward parallelization, and an ability to handle inter-facial dynamics and complex boundaries. LB methods have become quite popular, and an extensive literature has developed. The reader is directed to any of [CD98, HCD98, FHP86, GdL98, HSB89] for additional detail.

The purpose of this note is to suggest that LB methods may be effectively applied to certain lighting problems, in particular, those requiring accurate representation of multiple, anisotropic scattering, e.g., in lighting participating media such as clouds, dust, and smoke.

At the heart of any LB simulation is a lattice, which is tiled across the space of interest. In two dimensions, hexagonal or rectangular grids are most often used. Each node of the lattice has an associated set of directional densities, where each density flows toward a specific neighbor node in the lattice. Thus for a hexagonal lattice, each node will have six directional densities, and for a rectangular lattice, each will have eight. Many models incorporate an additional density, called a “rest density,” that flows from each node to itself. These are useful in capturing certain multi-step effects such as visco-elasticity for non-Newtonian fluids. Each lattice direction has an associated speed at which the density flows. This is used as a weighting, to ensure an isotropic base, in cases where the distance between lattice neighbors is not equal. Thus, all directions in a hexagonal lattice have equal speeds, but a rectangular lattice has speeds of $\sqrt{2}$ between diagonal neighbors and unit speeds otherwise. In three dimensions, a wide variety of lattice configurations are used, but the most common choice is a rectangular lattice with neighbors given by the non-corner points of a cube of unit radius, $\{-1,0,1\}^3$. This yields a system with 18 neighboring directions and one rest direction, as illustrated in Figure 1.

All computations in an LB simulation are performed locally at each lattice point. The density flowing into a point is redistributed to lattice neighbors using a set of “collision rules”. The collision rules are most conveniently expressed

[†] Dept. of Comp. Sci., email:{rmg,rkarl,westall}@cs.clemson.edu
[‡] Dept. of Elec. & Comp. Eng., email:{rjschal}@clemson.edu

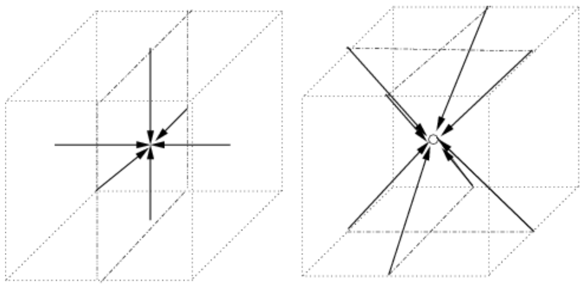

Figure 1: *The 18 neighbors used for 3-dimensional lattice computations, with the 6 neighbors at distance 1 unit (left) and the 12 neighbors (8 shown) at distance $\sqrt{2}$ (right)*

as a fixed matrix, Θ, where entry $\Theta_{i,j}$ contains the fraction of flow in direction j that will be diverted to direction i. For each time step, the redistribution of density at all nodes is performed synchronously. That is, we can think of the redistribution, or update, as comprising two steps. In the first step, the collision phase, the densities at each lattice point are locally redistributed to new directions according to the collision matrix. In the second step, the redirected densities at each site flow to neighboring sites, in their respective post-collision directions. This two-step, synchronous update sequence repeats until the system converges to a steady state. Initially, the directional densities in the lattice are set to random values except at the boundaries. Specific directional densities are often injected at certain boundaries on each time step to represent incoming flow. Directional density that would exit the lattice on any time step is either reflected (to represent hard boundaries) or captured in a sink to test for system equilibrium (injected flow = exit flow).

In terms of a matrix operation, the synchronous density update is simply

$$\Theta\vec{I} = \vec{O} \tag{1}$$

where $\vec{I}$ is the vector of current directional densities at a single site and $\vec{O}$ is the vector of density flowing out of that site after the distribution. Note that the components of $\vec{I}$ represent all directional flows at a given site at a given time, whereas the components of $\vec{O}$ are each sent to a different (neighbor) site at the next time step. Pseudocode describing the update of the lattice at each time step is given in Appendix B.

The model formulation expressed by (1) is attractive in its simplicity and the ease with which intricate boundary conditions, in additional to simple reflection or pass-through, might be incorporated. The essential difficulty in any application of the lattice-Boltzmann technique, and its true limitation, lies in verifying that a nearly trivial update equation, such as our equation (1), has, as its limiting behavior, a specific target system of partial differential equations that describes the system of interest. Our LB approach, which might be regarded as a computationally-trivial discrete ordinates method, will be seen to have an underlying diffusion process that emerges as lattice spacing and time step approach zero.

The remainder of the paper is organized as follows. In the next section we describe related work, both in lattice-Boltzmann modeling applied to graphics and in lighting participating media. In section 3 we describe the new LB model and derive the underlying diffusion process. This is the heart of the paper, since such derivations represent the only significant obstacles to application. In section 4, we apply the model to lighting clouds, where the cloud density models are obtained by a novel combination of two previously suggested techniques. Some implementation details and potential limitations of the approach are provided in section 5. Conclusions follow in section 6.

2. Related Work

LB methods have been successfully applied in other areas of computer graphics. Wei *et al.* [WLMK04] and Li *et al.* [LWK03] describe methods for implementing small LB simulations on graphics hardware for simulations of gases. Harris *et al.* [HISL03] have used a similar concept, the coupled map lattice, for hardware simulation of the formation and evolution of clouds.

Because lattice-Boltzmann methods compete directly with finite-element methods on many problems domains, multi-scale LB methods, e.g. [FH98], have been developed in the same spirit as the multi-grid FE methods.

The process of light scattering in a volume filled with some medium can be described by the standard volume radiative transfer equation

$$(\vec{\omega}\cdot\nabla+\sigma_t)\,L(\vec{x},\vec{\omega}) = \sigma_s\int p(\vec{\omega},\vec{\omega}')L(\vec{x},\vec{\omega}')d\vec{\omega}' + Q(\vec{x},\vec{\omega}) \tag{2}$$

where $\vec{x}$ is a position in space, $\vec{\omega}$ is a spherical direction, $p(\vec{\omega},\vec{\omega}')$ is the phase function, σ_s is the scattering coefficient, σ_a is the absorption coefficient, $\sigma_t = \sigma_s + \sigma_a$ is the extinction coefficient, and $Q(\vec{x},\vec{\omega})$ is the emissive field in the volume [Arv93]. Over an infinitesimal path, the left side of Equation 2 represents the attenuation due to both absorption and scattering as characterized by $\sigma_t = \sigma_a + \sigma_s$. The right side represents the amount of light scattered into this path from outside and emitted from within the path. In the steady state they must be equal.

Radiative transfer in volumes is a well-studied topic. Early approaches to the simulation of light scattering in participating media assumed that propagating rays encountered at most one scattering event [EP90, Sak90]. Several techniques have been suggested to capture multiple scattering. Rushmeier and Torrance [RT87] used a radiosity (finite-element) technique to model energy exchange between environmental zones and hence capture isotropic scattering. Max

[Max94] and Languénou *et al.* [LBC94] were able to capture anisotropic effects by extending the *discrete ordinates* method for radiation transfer in which all transfer is limited to a few discrete directions, chosen for optimal Gaussian quadrature in integrals over solid angles. A fast approximation to multiple, anisotropic scattering that uses half-angle (between light and viewpoint) slicing has recently been suggested by Kniss *et al.* It is limited to forward scattering.

Analytical models of multiple scattering have invariably led to diffusion processes. Kajiya and Von Herzen [KH84] used spherical harmonics to expand both the light intensity field and the scattering phase function. They obtained a coupled set of partial differential equations in the spherical harmonic coefficients whose solution would yield intensity at each spatial coordinate. Stam [Sta95] observed that an approximate solution was a diffusion process and provided substantial detail regarding this process, including a suggested multi-grid solution technique. More recently, Jensen *et al.* [JMLH01] showed that a simple, two-term approximation of radiance naturally leads to a diffusion approximation that is appropriate for a highly scattering medium. We will see that the simple LB update (1) can also provide a diffusion process.

3. Transport Model

Our transport model is based upon a discrete representation of time, space, and direction within a scattering volume. We evaluate a discrete approximation of equation (2) locally at each point within the volume. As the size of the time step and the distance between spatial points in our approximation approach 0, the approximation will be shown to converge toward a solution of (2).

Formally, we postulate a three-dimensional photon density transport in terms of a spatial and temporal Markovian update on a lattice:

$$f_i(\vec{r}+\lambda\vec{c}_i, t+\tau) = \Theta_i(f(\vec{r},t)), \qquad \text{i = 0,1, ..., 18} \tag{3}$$

where $f_i(\vec{r},t)$ is the density arriving at lattice site $\vec{r} \in \Re^3$ at time t in direction $\vec{c}_i$, λ is lattice spacing, τ is a time step, and Θ_i is the i^{th} row of the update matrix, Θ, to be specified. The directions $\vec{c}_i$, i = 0,1,...,18, are all the non-corner lattice points of a cube of unit radius, $\{-1,0,1\}^3$. We take $\vec{c}_0 = (0,0,0)$, and $\vec{c}_1$ - $\vec{c}_6$ to be the axis directions.

As with all lattice-Boltzmann models, the advantage of this approach is the speed, storage, and simplicity of the update (3). The challenge is showing that the limiting behavior (as $\lambda, \tau \to 0$) is a target differential equation that adequately describes the system of interest.

Consider first the isotropic case. Our update matrix is given as follows. For row 0:

$$\Theta_{0j} = \begin{cases} 0 & j=0 \\ \sigma_a & j>0 \end{cases} \tag{4}$$

For the axial rows, $i = 1,\ldots,6$:

$$\Theta_{ij} = \begin{cases} 1/12 & j=0 \\ \sigma_s/12 & j>0, \quad j \neq i \\ 1-\sigma_t+\sigma_s/12, & j=i \end{cases} \tag{5}$$

For the non-axial rows, $i = 7,\ldots,18$:

$$\Theta_{ij} = \begin{cases} 1/24 & j=0 \\ \sigma_s/24 & j>0, \quad j \neq i \\ 1-\sigma_t+\sigma_s/24, & j=i \end{cases} \tag{6}$$

Directional density f_0 is the absorption/emission component, and $\sigma_t \in [0,1]$ is the extinction coefficient of the medium. As usual, $\sigma_t = \sigma_a + \sigma_s$, where σ_a represents extinction due to absorption and σ_s represents extinction due to scattering. The first row of Θ (4) indicates that fraction σ_a of incoming density in any direction $j > 0$ is absorbed on each (synchronous) update step. The first column ($j = 0$ in Equations 5 and 6) indicates that all previously absorbed density will be emitted uniformly on the next (synchronous) step, except that we weight axis directions twice as heavily as non-axis directions to ensure isotropic flow in the case that the scattering coefficient is independent of direction [CD98]. The remaining non-diagonal entries indicate that the scattered fraction of the incoming density, σ_s, is isotropically distributed to neighboring lattice points. The diagonal entries account for the transmission of that density which is not scattered. Note that the matrix is stochastic, i.e., $\sum_{i=0}^{18} \Theta_{i,j} = 1$, all j. We denote total site density by $\rho(\vec{r},t) = \sum_{i=0}^{18} f_i(\vec{r},t)$.

To explore behavior in the limit, it will be useful to work with increments, and so we rewrite (3) as:

$$f_i(\vec{r}+\lambda\vec{c}_i, t+\tau) - f_i(\vec{r},t) = \Omega_i(f(\vec{r},t)) \tag{7}$$

where $\Omega = \Theta - I$. Expanding the left side of (7) in a Taylor series, we have:

$$[(\lambda\vec{c}_i,\tau)\cdot\nabla]f_i(\vec{r},t) + \frac{[(\lambda\vec{c}_i,\tau)\cdot\nabla]^2}{2!}f_i(\vec{r},t) + \ldots = \Omega_i(f(\vec{r},t)) \tag{8}$$

where

$$\nabla = (\partial/\partial\vec{r}, \partial/\partial t) = (\partial/\partial x, \partial/\partial y, \partial/\partial z, \partial/\partial t) \tag{9}$$

For the diffusion behavior we seek, it will be important for the time step to approach 0 faster than the lattice spacing. Specifically, we write

$$t = \frac{t_0}{\varepsilon^2} \qquad \text{where} \quad t_0 = o(\varepsilon^2)$$

$$\vec{r} = \frac{\vec{r}_0}{\varepsilon} \qquad \text{where} \quad \|\vec{r}_0\| = o(\varepsilon)$$

Then

$$\frac{\partial}{\partial t} = \varepsilon^2 \frac{\partial}{\partial t_0}$$

$$\frac{\partial}{\partial r_\alpha} = \varepsilon \frac{\partial}{\partial r_{0\alpha}} \qquad \text{for} \quad \alpha \in \{x,y,z\}$$

As is standard practice in lattice-Boltzmann modeling, we also assume that we can write $f(\vec{r},t)$ as a small perturbation on this same scale about some local equilibrium, i.e.,

$$f(\vec{r},t) = f^{(0)}(\vec{r},t) + \varepsilon f^{(1)}(\vec{r},t) + \varepsilon^2 f^{(2)}(\vec{r},t) + \ldots \tag{10}$$

where the local equilibrium carries the total density, $\rho(\vec{r},t) = \sum_{i=0}^{18} f_i^{(0)}(\vec{r},t)$. Equation (10) is the Chapman-Enskog expansion from statistical mechanics [CD98], wherein it is assumed that any flow that is near equilibrium can be expressed as a perturbation in the so-called Knudsen number, ε, which represents the mean free path (expected distance between successive density collisions) in lattice spacing units.

Equation (7) now becomes:

$$\left[[\varepsilon\lambda(\vec{c}_i \cdot \tfrac{\partial}{\partial \vec{r}_0}) + \varepsilon^2 \tau \tfrac{\partial}{\partial t_0}] + \frac{[\varepsilon\lambda(\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) + \varepsilon^2 \tau \frac{\partial}{\partial t_0})]^2}{2} + \ldots \right] \times$$
$$(f_i^{(0)} + \varepsilon f_i^{(1)} + \ldots) = \Omega_i (f^{(0)} + \varepsilon f^{(1)} + \ldots) \tag{11}$$

Equating coefficients of ε^0 in (11), we obtain:

$$0 = \Omega_i (f^{(0)}(\vec{r},t)) \tag{12}$$

i.e., $f^{(0)}$ is indeed a local equilibrium. In general, a local equilibrium need not be unique, and the choice can affect the speed of convergence [KFO99]. Nevertheless, in this case it turns out that Ω has a one-dimensional null space. We observe that

$$v = (\sigma_a, 1/12, \ldots, 1/12, 1/24, \ldots, 1/24)$$

(where entries 1 - 6 are $1/12$) satisfies $\Omega_i v = 0$, all i, and so we must have

$$f_i^{(0)} = K v_i$$

where the scaling coefficient, K, is determined by the requirement that $\rho = \sum_i f_i^{(0)} = K \sum_i v_i = K(1+\sigma_a)$. Thus we have:

$$f_i^{(0)}(\vec{r},t) = \frac{v_i}{1+\sigma_a} \rho(\vec{r},t) \tag{13}$$

Similarly, equating coefficients of ε^1 in (11), we obtain:

$$\lambda(\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) f_i^{(0)}(\vec{r},t) = \Omega_i f^{(1)}(\vec{r},t) \tag{14}$$

that is,

$$\frac{\lambda v_i}{1+\sigma_a} (\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) \rho(\vec{r},t) = \Omega_i f^{(1)}(\vec{r},t) \tag{15}$$

We would like to solve (15) for $f^{(1)}$, but we cannot simply invert Ω, since it is singular. Nevertheless, we can observe that any of

$$(c_{0_\alpha}, c_{1_\alpha}, \ldots, c_{18_\alpha}) \quad \text{where } \alpha \in \{x,y,z\}$$

as well as any of

$$(v_0 c_{0_\alpha}, v_1 c_{1_\alpha}, \ldots, v_{18} c_{18_\alpha}) \quad \text{where } \alpha \in \{x,y,z\}$$

is an eigenvector of Ω with eigenvalue $-\sigma_t$. Thus, if we write

$$f_i^{(1)}(\vec{r},t) = K v_i (\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) \rho(\vec{r},t)$$

and substitute into (15), we can determine that $K = -\lambda/((1+\sigma_a)\sigma_t)$ and so

$$f_i^{(1)}(\vec{r},t) = \frac{-\lambda v_i}{(1+\sigma_a)\sigma_t} (\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) \rho(\vec{r},t) \tag{16}$$

Finally, we need to equate ε^2 terms in (11), but here it will suffice to sum over all directions. We obtain:

$$\sum_{i=0}^{18} \left[\tau \frac{\partial f_i^{(0)}}{\partial t_0} + \lambda(\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0}) f_i^{(1)} + \frac{\lambda^2}{2} (\vec{c}_i \cdot \frac{\partial}{\partial \vec{r}_0})^2 f_i^{(0)} \right] = 0 \tag{17}$$

Substituting expressions (13) and (16) into equation (17) and observing that

$$\sum_{i=0}^{18} v_i c_{i_\alpha} c_{i_\beta} = (1/2)\delta_{\alpha\beta} \quad \text{for } \alpha, \beta \in \{x,y,z\}$$

we obtain

$$\frac{\partial \rho}{\partial t_0} - \frac{\lambda^2 (1/\sigma_t - 1/2)}{2\tau(1+\sigma_a)} \left(\frac{\partial^2 \rho}{\partial r_{0_x}^2} + \frac{\partial^2 \rho}{\partial r_{0_y}^2} + \frac{\partial^2 \rho}{\partial r_{0_z}^2} \right) = 0 \tag{18}$$

which is the standard diffusion equation,

$$\frac{\partial \rho}{\partial t_0} = D \nabla_{\vec{r}_0}^2 \rho \tag{19}$$

with diffusion coefficient

$$D = \left(\frac{\lambda^2}{\tau} \right) \left[\frac{(2/\sigma_t) - 1}{4(1+\sigma_a)} \right]$$

4. Lighting Clouds

To illustrate application of our transport model to lighting clouds, we obviously need a cloud density generator. The most successful density generators in the literature use a two-stage, macro-structure/micro-structure model [EMP*02, KPH*03, DKY*00, HL01]. We follow this lead, but because we want to stay within the realm of physically-based modeling, we employ a variation on these approaches. For the macroscopic structure, we use the model of Miyazaki *et al.* [MYDN01], which can be regarded as an approximation of the momentum and energy equations for fluid flow at the Navier-Stokes level. We then treat the shape outputs of this model as masks for humidity seeding in the percolation model of Nagel and Raschke [NR92]. The most compelling argument in favor of the Nagel-Raschke model is the excellent agreement with real clouds in fractal analysis. To retain this characteristic, we avoid explicit Gaussian smoothing of the binary, percolation model output in computing density at grid nodes. Instead, we first expand the target density grid (of size N^3, where, for the examples shown, $N = 128$) by a factor of K in each dimension. Because the Nagel-Raschke model can be implemented with efficient bit

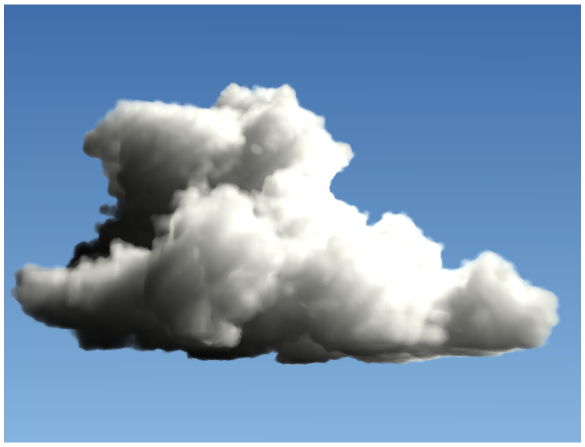

Figure 2: *Sample cloud rendering with isotropic scattering ($g = 0.0$) and single scattering albedo 0.9.*

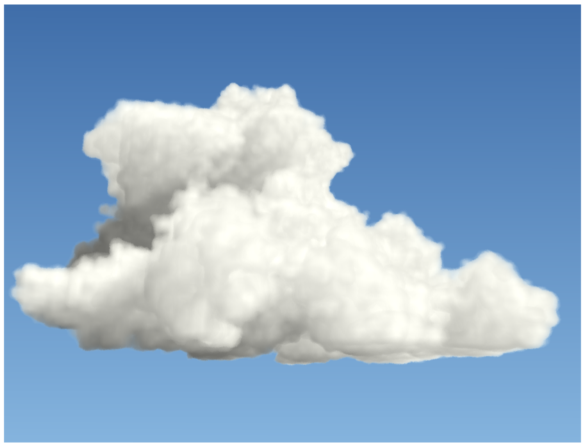

Figure 3: *Sample cloud rendering with forward scattering ($g = 0.85$) and single scattering albedo 0.9.*

operators and storage, this does not lead to inordinate storage requirements. The binary output is then averaged over each cube of edge dimension K to provide a real density with resolution $1/K^3$ at each of the original N^3 sites. In the examples shown, we used $K = 5$.

To provide for anisotropic, non-homogeneous scattering, we modify the entries of Θ. First, because $\Theta_{i,j}$ controls scattering from direction $\vec{c}_j$ into direction $\vec{c}_i$, we scale σ_s in $\Theta_{i,j}$ by

$$\frac{p_{i,j}}{\sum_{j=1}^{18} p_{i,j}/18}$$

where $p_{i,j}$ is a discrete version of the the Henyey-Greenstein phase function [HG40],

$$p_{i,j} = \frac{1-g^2}{(1-2g\vec{n}_i \cdot \vec{n}_j + g^2)^{3/2}}$$

Here $\vec{n}_i$ is the normalized direction, $\vec{c}_i$, and $g \in [-1,1]$ is a parameter that allows for the anisotropic scattering. Note that $g > 0$ provides forward scattering, $g < 0$ provides backward scattering, and $g = 0$ returns us to the isotropic case. To provide density-dependence, the σ values in each entry of Θ are scaled by the density of the scattering medium at the lattice point before applying the update (7). Note that a density of zero at a lattice site yields a simple pass-through of photon density.

For rendering, the directional densities at each grid location are summed to represent the illuminate at the location. We then march rays through the volume to form images as in [KH84]. Note that the display could, instead, be made directionally dependent by resolving the viewing direction into the minimal grid-directional positive components and using only those $f_i(\vec{r},t)$. In Figure 2 we show a sample rendering with a single scattering albedo (σ_s/σ_t) of 0.9, a per-lattice-site extinction coefficient $\sigma_t = 0.25$, and isotropic scattering ($g = 0$). In Figure 3 we show the same cloud density from the same viewpoint with significant forward scattering ($g = 0.85$).

With this method, it is easy to simulate non-homogeneous materials. Figure 4 shows an example of a rainbow gradient applied to σ_s over a cloud shaped like the Stanford Bunny (www-graphics.stanford.edu/data/3Dscanrep/). The cloud density was generated by applying the percolation model to the Bunny interior. With this figure surface effects are not considered, so the resulting simulation is not directly comparable to current subsurface scattering techniques.

5. Details and Limitations

At the beginning of each time step, photon density corresponding to light entering the system must be added to the boundary nodes of the lattice. Such boundary conditions are handled by resolving the illumination source (sun) direction into a minimal collection of grid-directional, positive components at each boundary site. For this step we use a procedure, similar to Gram-Schmidt orthogonalization, wherein we repeatedly select, from those lattice directions not yet selected, that grid direction having the largest dot product with the remaining light direction and then subtract that contribution from the remaining light direction. Pseudocode for this operation is listed in Appendix A. For those selected lattice directions that point interior to the grid, directional flow $(f_i(\vec{r},t))$ is fixed at a constant value on each time step. Grid outflow is captured in a sink to test for equilibrium, but it is otherwise unused. We note that it could be sampled to implement shadows cast by the cloud density on surfaces external to the grid.

In many LB simulations, the boundaries of the lattice are periodic. Thus, as density flows out one side, it flows back in

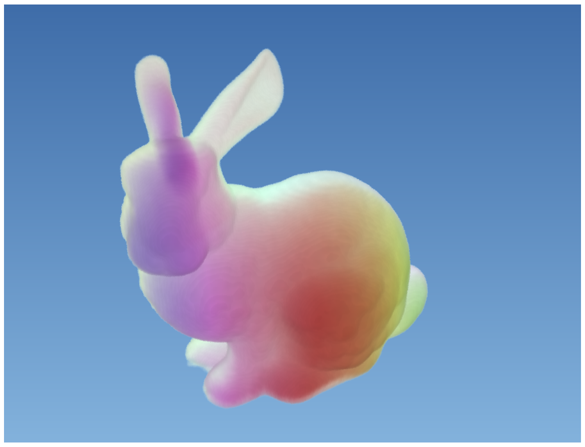

Figure 4: *Bunny-shaped cloud with anisotropic scattering ($g = 0.25$) and a wavelength dependent σ_s smoothly varying with position.*

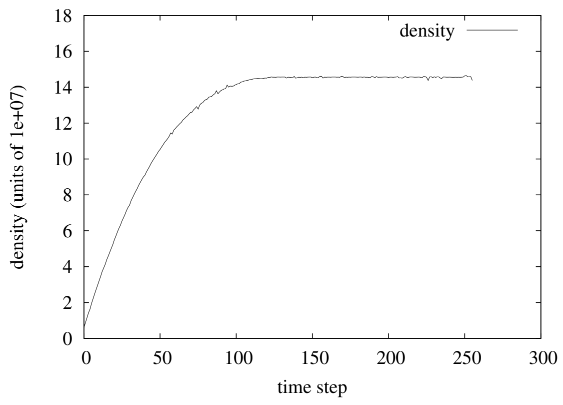

Figure 5: *The total density in the lattice at each time step for the 128x128x128 lattice shown in Figure 2*

the other. For many applications, it is appropriate to obstruct some of the boundaries with some sort of barrier or wall and modify the collision rules at such sites to include reflection or absorption. For simulations of light through participating media, these methods do not relate to the physical processes being simulated. Instead, we allow density to freely flow out of the boundaries of the lattice. Since we always inject density from a light source at the beginning of each time step, the system eventually converges to a steady state, as long as the eigenvalues of Ω are $\in (-2, 0)$.

While we have used this procedure for simulations involving external light sources, there is nothing preventing us from simulating emissive media. For an emissive medium, photon density would be added at appropriate lattice points at each time step.

By experiment, convergence generally requires a number of iterations of (7) equal to twice the longest dimension of the grid. This can be seen in Figure 5 where the total density in the lattice is plotted for each time step with a 128^3 lattice.

Our method does not provide the real-time performance seen by other approaches [HL01, KPH*03], but the quality of the images, the simplicity of the algorithm (7), and the ability to handle complex scattering events make it attractive. To estimate the performance available through parallelization, we implemented a single-channel, $N = 128$ model on a 64-processor Beowulf-class cluster with 1.6GHz Pentium IV processors and 100Mb Ethernet links. In this implementation, the algorithm required 0.093 seconds per iteration of (7), and thus approximately 24 seconds to convergence, which is not unreasonable for a system with more than two million nodes. The single CPU time was 6.15 seconds per iteration. Note that the speedup was slightly super-linear.

A disadvantage of this method is the amount of storage required to hold the lattice. We need to represent N^3 sites, each with 19 directional densities (floats). If N is as large as 256, this can be extensive, although space requirements can be partitioned among multiple computational nodes in a straightforward manner. A few conventional PCs would suffice for a very large model.

Work is progressing on multi-resolution LB methods, such as [FH98], in order to help lower the resource usage of the simulation.

The current simulation does not take into account any interaction between light and surfaces. This limits the class of objects modeled to participating media such as clouds, dust and smoke. An extension to include interaction with surfaces and BRDF-based boundary conditions would be both an interesting and useful exercise and might allow for the simulation of subsurface scattering in complex, non-homogeneous media such as skin.

6. Conclusions

We have suggested a new technique, based on a lattice-Boltzmann method, for lighting participating media. Although the technique does not deliver real-time performance, it does offer a very simple implementation, high-quality images, an ability to capture complex scattering events, and an underlying analytic (diffusion process) model. The essential obstacle to any application of the lattice-Boltzmann technique lies in verifying that a discrete system of relatively simple, synchronous updates has, as its limiting behavior, a specific target system of partial differential equations that describe the system of interest. We have applied this technique to a model of photon transport and provided this verification.

The ease with which lattice-Boltzmann methods can handle complex boundary conditions suggests that applications to modeling subsurface scattering may be available.

7. Acknowledgment

This work was supported in part by the ERC Program of the U.S. National Science Foundation under award EEC-9731680 and the ITR Program of the National Science Foundation under award ACI-0113139.

References

[Arv93] ARVO J.: Transfer equations in global illumination. In *Global Illumination, SIGGRAPH '93 Course Notes* (August 1993). 2

[CD98] CHOPARD B., DROZ M.: *Cellular Automata Modeling of Physical Systems*. Cambridge Univ. Press, Cambridge, UK, 1998. 1, 3, 4

[DKY*00] DOBASHI Y., KANEDA K., YAMASHITA H., OKITA T., NISHITA T.: A simple, efficient method for realistic animation of clouds. In *Proc. of SIGGRAPH 2000* (July 2000), pp. 19 – 28. 4

[EMP*02] EBERT D., MUSGRAVE F., PEACHEY D., PERLIN K., WORLEY S., MARK W., HART J.: *Texturing and Modeling: A Procedural Approach*, third ed. Morgan Kaufmanm, December 2002. 4

[EP90] EBERT D., PARENT R.: Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. *ACM Computer Graphics (SIGGRAPH '90) 24*, 4 (August 1990), 357–366. 2

[FH98] FILIPPOVA O., HANEL D.: Grid refinement for lattice-bkg models. *Journal of Computational Physics 147* (1998), 219–228. 2, 6

[FHP86] FRISCH U., HASSLACHER B., POMEAU Y.: Lattice-gas automata for the navier-stokes equation. *Physical Review Letters 56* (April 1986), 1505–1508. 1

[GdL98] GIRAUD L., D'HUMIERES D., LALLEMAND P.: A lattice boltzmann model for jeffreys viscoelastic fluid. *Europhysics Letters 42* (June 1998), 625–630. 1

[HCD98] HE X., CHEN S., DOOLEN G.: A novel thermal model for the lattice boltzmann method in incompressible limit. *Journal of Computational Physics 146* (1998), 282–300. 1

[HG40] HENYEY G., GREENSTEIN J.: Diffuse radiation in the galaxy. *Astrophysical Journal 88* (1940), 70–73. 5

[HISL03] HARRIS M., III W. B., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Graphics Hardware* (2003). 2

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. In *EG 2001 Proceedings* (2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3), Blackwell Publishing, pp. 76–84. 4, 6

[HSB89] HIGUERA F., SUCCI S., BENZI R.: Lattice gas dynamics with enhanced collisions. *Europhysics Letters 9* (June 1989), 345–349. 1

[JMLH01] JENSEN H., MARSCHNER S., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001* (August 2001), pp. 511–518. 3

[KFO99] KARLIN I., FERRANTE A., OTTINGER C.: Perfect entropy functions of the lattice boltzmann method. *Europhysics Letters 47* (July 1999), 182–188. 4

[KH84] KAJIYA J., HERZEN B. V.: Ray tracing volume densities. *ACM Computer Graphics (SIGGRAPH '84) 18*, 3 (July 1984), 165–174. 3, 5

[KPH*03] KNISS J., PREMOŽE S., HANSEN C., SHIRLEY P., MCPHERSON A.: A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (April 2003), 150–162. 4, 6

[LBC94] LANGUÉNOU E., BOUATOUCH K., CHELLE M.: Global illumination in presence of participating media with general properties. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 69–85. 3

[LWK03] LI W., WEI X., KAUFMAN A.: Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer 19*, 7-8 (2003), 444–456. 2

[Max94] MAX N. L.: Efficient light propagation for multiple anisotropic volume scattering. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 87–104. 3

[MYDN01] MIYAZAKI R., YOSHIDA S., DOBASHI Y., NISHITA T.: A method for modeling clouds based on atmospheric fluid dynamics. In *Proc. Ninth Pacific Conference on Computer Graphics and Applications (PG'01)* (Tokyo, Japan, October 2001), pp. 363 – 372. 4

[NR92] NAGEL K., RASCHKE E.: Self-organizing criticality in cloud formation? *Physica A 182* (1992), 519–531. 4

[RT87] RUSHMEIER H., TORRANCE K.: The zonal method for calculating light intensities in the presence of a participating medium. In *Proc. SIGGRAPH '87* (July 1987), pp. 293–302. 2

[Sak90] SAKAS G.: Fast rendering of arbitrary distributed volume densities. In *Proc. Eurographics '90* (September 1990), pp. 519–530. 2

[Sta95] STAM J.: Multiple scattering as a diffusion process. In *Proc. 6th Eurographics Workshop on Rendering* (Dublin, Ireland, June 1995), pp. 51–58. 3

[WLMK04] WEI X., LI W., MUELLER K., KAUFMAN A.: The lattice-boltzmann method for gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004). 2

Appendix A: Resolving Directions

This section presents pseudocode for resolving a light direction, $\vec{L}$, with intensity 1, into a lattice of 18 directional densities. This does not include the "rest density" which does not have an associated direction. The parameter $\vec{f}$ stores the resulting light in terms of lattice directions.

```
void light_to_latt($\vec{L}$, $\vec{f}$) {
  int i, j, dirs[18];
  // Normalized light intensity
  float intens = 1.0;
  float latt_dirs[] = {1, 0, 0, -1, ...};

  for (i=0; i<18; i++) dirs[i] = i;

  // Sort lattice dirs on dot product
  // with the light direction, in
  // desc order; store idxs in dirs
  sort_by_dp(dirs, $\vec{L}$, latt_dirs);

  for (i=0; i<18; i++) {
     // If 0 intensity remains, stop
     if (intens < $\varepsilon$) break;
     dp = dot(latt_dirs[dirs[i]], $\vec{L}$);
     $\vec{f}$[dirs[i]] = min(dp,intens);
     intens -= dp;
  }
}
```

Appendix B: Lattice Updates

Here we present pseudocode for the main lattice update step, where directional densities are scattered and distributed to neighboring locations. Θ is the matrix describing the redistribution of density (see Equations 3-5 for the isotropic case)

```
void update(node *src, node *dst,
     float **$\Theta$) {
  int i, out, in, neigh;
  float new_dense[19];

  // Inject new light density on the
  // borders of the lattice
  inject_light(src);

  // Distribute density locally
  // according to the collision rules
  for (i=0, i<# nodes; i++) {
    for (in=0; in<19; in++)
      new_dense[in] = 0;

    for (in=0; in<19; i++) {
      for (out=0; out<19; out++) {
        new_dense[out] +=
          src[i].dir[in]*$\Theta$[in][out];
      }
    }

    for (in=0; in<19; in++)
      src[i].dir[in] = new_dense[in];
  }

  // Now flow density to neighbors
  for (i=0, i<# nodes; i++) {
    for (in=0; in<19; in++) {
      // Compute the index of the
      // node at i in the in direction
      neigh = compute_neighbor (i, in);

      if (neigh is a valid index)
        dst[neigh].dir[in]
          = src[i].dir[in];
    }
  }
}
```