Efficient LBM Visual Simulation on Face-Centered Cubic Lattices

Kaloian Petkov, Feng Qiu, *Member*, *IEEE*, Zhe Fan, Arie E. Kaufman, *Fellow*, *IEEE*, and Klaus Mueller, *Senior Member*, *IEEE*

Abstract—The Lattice Boltzmann method (LBM) for visual simulation of fluid flow generally employs cubic Cartesian (CC) lattices such as the D3Q13 and D3Q19 lattices for the particle transport. However, the CC lattices lead to suboptimal representation of the simulation space. We introduce the face-centered cubic (FCC) lattice, fD3Q13, for LBM simulations. Compared to the CC lattices, the fD3Q13 lattice creates a more isotropic sampling of the simulation domain and its single lattice speed (i.e., link length) simplifies the computations and data storage. Furthermore, the fD3Q13 lattice can be decomposed into two independent interleaved lattices, one of which can be discarded, which doubles the simulation speed. The resulting LBM simulation can be efficiently mapped to the GPU, further increasing the computational performance. We show the numerical advantages of the FCC lattice on channeled flow in 2D and the flow-past-a-sphere benchmark in 3D. In both cases, the comparison is against the corresponding CC lattices using the analytical solutions for the systems as well as velocity field visualizations. We also demonstrate the performance advantages of the fD3Q13 lattice for interactive simulation and rendering of hot smoke in an urban environment using thermal LBM.

Index Terms—Lattice Boltzmann method, face-centered cubic, fD3Q13, D3Q13, D3Q19, GPU.

1 Introduction

When it comes to regular gridded sample arrangements in both space and time, the Cartesian lattice has been the lingua franca in science, medicine, and engineering. It is simple, straightforward to index, and easy to collect data on. However, recent years have seen an increasing move toward other regular lattices hailed to be more space efficient, and thus, more optimal in their distribution of samples. These proposals were based on the recognition that an optimal packing of these lattices in the frequency domain yields the sparsest sample arrangement in the spatial domain, assuming a near-spherical frequency spectrum. This finding emerges from the application of the Fourier sampling theorem and also the theory of lattice reciprocity. The one lattice that achieves this sparse sampling is the Body-Centered Cubic (BCC) lattice. It was elaborately shown that the BCC lattice affords a 30 percent reduction in the number of samples, or a 30 percent increase of resolution in the spatial domain, a number that grows to 50 percent when extended into 4D. The reciprocal lattice to the BCC is the Face-Centered Cubic (FCC) lattice, which is the lattice that exhibits the tightest packing. However, this tight packing has other implications as well. It also produces the most isotropic sample distribution, yet not the sparsest. This property, while often neglected, is very desirable in any type of communication scenarios and also for digital object representation. It affords the best optimal orientation independency in the coverage of space and time.

We claim that the FCC lattice is the perfect medium to conduct simulations that use communications over lattice links to propagate energy within the simulation medium. At the same time, we also claim that this lattice is ideal to model interactions of the flow phenomena with scene objects, since afforded by this isotropy, this lattice is most insensitive to how scene objects came to rest with respect to the orientation and organization of the lattice. Interestingly enough, it is this isotropy that subsequently affords a more efficient communication in these simulations, which leads to the possibility to dispense with some of the lattice links without harm, and thus save on costly communication computations. In this paper, we employ the FCC lattice in the context of flow simulation and visualization via the Lattice Boltzmann method (LBM) which has become a popular method for physically-based simulations.

In recent years, the importance of physically-based simulations has been growing in fields such as computer games, movie productions, and product design testing, where there is a need for computer-generated realism at interactive rates. Simulations based on the finite element method are expensive and unsuitable for interactive application while lattice-based simulations have emerged as an efficient and easy-to-implement alternative. In particular, the LBM is a powerful technique that can be used to model complex fluid flows [29]. It uses a simple set of collision and propagation rules that are applied independently at a mesoscopic level on particle distributions in the lattice. These local distributions can then be used to model the global behavior of the fluid flow, solving the Navier-Stokes equations in the incompressible limit.

We adopt the notation DmQn used by physicists to describe lattices, where m represents the number of spatial

• *The authors are with the Computer Science Department, Stony Brook University, Stony Brook, NY 11790.*
E-mail: {kpetkov, qfeng, fzhe, ari, mueller}@cs.sunysb.edu.

Manuscript received 8 Sept. 2008; revised 31 Dec. 2008; accepted 11 Feb. 2009; published online 20 Feb. 2009.
Recommended for acceptance by M. Chen, C. Hansen, and A. Pang.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2008-09-0143.
Digital Object Identifier no. 10.1109/TVCG.2009.32.

dimensions and n is the number of velocity links at each lattice site, including a zero-velocity link. Traditionally, LBM simulations have been performed on cubic Cartesian (CC) lattices, where the D3Q19 lattice provides a good balance between stability and computational/memory cost. Its properties and limitations are well known [29] and its cubic grid structure maps conveniently to memory layouts on both CPU and GPU architectures. At the same time, its high number of links and multiple link speeds introduce additional complexity in the LBM simulation, increase communication overhead across the links, and therefore, decrease performance. Based on the arguments of isotropy raised above, we propose the use of the face-centered cubic lattice (fD3Q13) for highly efficient LBM simulations.

The low number of single-speed links and the efficient node indexing on the fD3Q13 lattice greatly simplify and accelerate the simulation. The LBM is already amenable to GPU acceleration because of the highly local computations and the use of the fD3Q13 lattice further increases the speed of GPU-based implementations.

2 Related Work

2.1 The Lattice Boltzmann Method

The LBM is a grid-based model that is used to compute macroscopic fluid behavior by simulating mesoscopic particle interactions [29]. The approach is similar to the previously employed lattice-gas cellular Automata (LGCA), which as the name suggests models fluid flows in a manner similar to a cellular automaton. In the lattice-based approach, we describe the mesoscopic behavior by particle collisions at lattice sites and propagations along lattice links. The LGCA method models single particle Boolean dynamics, which leads to statistical noise. The LBM overcomes this limitation by tracing particle density distributions and the average effect of particle collisions. In the limit, as the lattice pitch approaches 0, the system solves the Navier-Stokes differential equation for incompressible fluids.

In the LBM, we have a lattice, which discretizes the spatial domain into a regular arrangement of lattice sites $\mathbf{x}$. Also, particle velocities $\mathbf{u}$ are represented by a finite set of lattice velocities denoted by $\mathbf{c_i}$. The continuous distribution function $f(\mathbf{x}, \mathbf{u}, t)$ is therefore represented by a set of distributions $f_i(\mathbf{x}, t)$, which assign a particle density value to each of the discrete lattice velocities for a site $\mathbf{x}$ at time t. For a given state of the particle densities, the next discrete state of the simulation is described by the discrete Lattice Boltzmann equation:

$$f_i(\mathbf{x} + \mathbf{c_i}\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}\left(f_i^{eq} - f_i\right). \tag{1}$$

In essence, this formula describes a simple collision and propagation rule. At each step of the simulation and for each lattice site, we examine the particle distributions, calculate the distribution contributions from particle collision, and synchronously propagate the new densities along the lattice links. Note that here, we employ the Bhatnagar-Gross-Krook (BGK) approximation to the collision integral, which describes how the particle densities are affected by the collision. The BGK approximation models collisions as a relaxation of momentum toward an equilibrium state defined by the distributions f_i^{eq} and conserves mass and momentum in the system. The constant τ controls the relaxation rate and its value is derived from the kinematic viscosity of the fluid ν:

$$\nu = \frac{1}{2}c_s^2(2\tau - 1) \Rightarrow \tau = \frac{\nu}{c_s^2} + \frac{1}{2}. \tag{2}$$

In this equation, c_s is the lattice speed of sound. This value determines the highest possible speed for information propagation and is specific to the selected lattice structure. Negative values for the distributions f_i quickly lead to numeric instability, and therefore, stable simulations can be achieved only for a limited range of τ values. At each step of the simulation, we can recover the macroscopic properties of the flow from the mesoscopic particle distributions at lattice sites:

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t), \tag{3}$$

$$j(\mathbf{x}, t) = \rho(\mathbf{x}, t)\nu(\mathbf{x}, t) = \sum_i \mathbf{c_i} f_i(\mathbf{x}, t). \tag{4}$$

In these equations, ρ is the mass density at a specific site location $\mathbf{x}$ and time t, and $\mathbf{j}$ is the momentum density. We can then calculate the particle velocity:

$$\mathbf{u}(\mathbf{x}, t) = \frac{\mathbf{j}(\mathbf{x}, t)}{\rho(\mathbf{x}, t)} = \frac{1}{\rho(\mathbf{x}, t)}\sum_i \mathbf{c_i} f_i(\mathbf{x}, t). \tag{5}$$

The equilibrium state used in the BGK collision model depends only on the conserved quantities, namely mass and momentum, and is described by the following:

$$f_i^{eq} = W_i\rho\left(1 + \frac{\mathbf{c_i}\cdot\mathbf{u}}{c_s^2} + \frac{(\mathbf{c_i}\cdot\mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}\cdot\mathbf{u}}{2c_s^2}\right). \tag{6}$$

Here, c_s is the speed of sound for the lattice, $\mathbf{u}$ is the particle velocity calculated from (5), and each lattice velocity $\mathbf{c_i}$ has an associated weight W_i. These weights and the lattice speed of sound are specific to the selected lattice geometry. Their values are obtained from the nonvanishing elements of the even velocity moments up to fourth order:

$$\begin{aligned}
\sum_i W_i &= \rho_0, \\
\sum_i W_i c_{i\alpha} c_{i\beta} &= \rho_0 c_s^2 \delta_{\alpha\beta}, \\
\sum_i W_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} &= \rho_0 c_s^4\left(\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}\right).
\end{aligned} \tag{7}$$

The calculated weights W_i ensure that: 1) the mass density is positive and 2) the lattice velocity moments up to fourth order are identical to the respective velocity moments over the Maxwell distribution [29]. Vanishing velocities have no influence on the isotropy of the lattice tensors. In these equations, ρ_0 is the constant mass density for incompressible fluids, which is usually set to 1, and δ_{ij} is the Kronecker delta symbol:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{8}$$

Note that the odd velocity moments vanish and do not provide any additional constraints. Each of the Greek letters α, β, γ, and δ corresponds to a Cartesian index of the lattice velocity, such as 1 or 2 in the case of 2D LBM and 1, 2, or 3 in the case of LBM in 3D. Detailed derivations and analysis of the LBM on frequently used lattices can be found in the work of Wolf-Gladrow [29] and Chen and Doolen [5].

While the LBM can model complex fluid systems with a set of simple collision and propagation rules, it suffers from a few limitations. It cannot easily handle the high-velocity flows used in aerodynamic simulations, for example. The physical system under consideration is made dimensionless by the selection of a length scale and a time scale, and is further discretized using an appropriate lattice step and a time step. The lattice viscosity depends on the discretization parameters and restricts the range of flow velocities that lead to stable simulations. Decreasing the lattice spacing and the time step increases the viscosity used in the LBM and allows for stable simulations with larger physical velocities (see [18] for additional information). However, simulating high-Reynolds-number flows can be prohibitively expensive in terms of both memory requirements and computational time. Hou et al. [13] proposed a subgrid model to deal with high Reynolds Number flows. Also, the single-relaxation-time constraint imposed by the BGK collision model leads to numerical instabilities when the simulation is coupled with heat transfer or body forces. The multiple-relaxation-time LBM (MRT-LBM) performs the collisions in moment space as opposed to the single-relaxation-time LBM (SRT-LBM) collision in the distributions space, and each moment is given an independent relaxation parameter [8]. After the relaxation, the inverse transform is used and the propagation rules are applied in phase space, or the space of the distributions. This approach exhibits improved numerical stability and has been coupled with heat transfer in the hybrid thermal Lattice Boltzmann Model (HTLBM) [17].

2.2 Boundary Conditions

In many simulations, the fluid interacts with objects whose boundaries may not follow the links in the discrete lattice structure, for example, when the object is defined procedurally or through a finite mesh. Different schemes for handling boundary conditions can be implemented by simply manipulating the particle densities during the collision step. This simplicity is offset by the fact that any error in the scheme will propagate throughout the simulation domain. The bounceback scheme is often used to implement no-slip boundary conditions [29]. The boundary is discretized over the lattice nodes and is assumed to intersect lattice links at the halfway point. Densities propagating along those links are simply moved to links in the opposite direction. This approach is very efficient in handling boundaries aligned with the lattice links, but leads to inaccuracies with curved boundaries. Different boundary conditions exist for LBM [19], [20], [25] including schemes that exhibit second-order accuracy for the boundary representation of curved surfaces.

In the 2D case, we employ the scheme proposed by Bouzidi et al. [3] for the treatment of curved boundary conditions. The curved boundary is illustrated in Fig. 1. The scheme combines the bounceback approach with linear interpolation along lattice links to achieve second-order accuracy of the boundary condition:

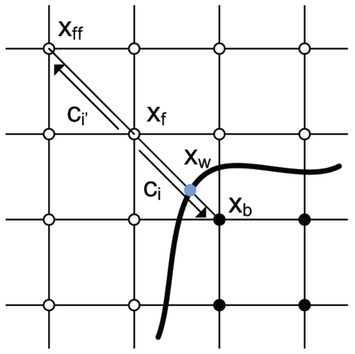


Fig. 1. Curved boundary treatment. The filled circles represent the discretized geometric object. The clear circles represent fluid cells. The blue circle at position x_w is the intersection point between the surface of the geometric object and a lattice link.

$$f_{i'}(\mathbf{x_f}, t+1) = \begin{cases} 2\Delta f_i^c(\mathbf{x_f}, t) + (1-2\Delta) f_i^c(\mathbf{x_f} - \mathbf{c_i}, t), & \text{if } \Delta < \frac{1}{2}, \\ \frac{1}{2\Delta} f_i^c(\mathbf{x_f}, t) + \frac{2\Delta - 1}{2\Delta} f_{i'}^c(\mathbf{x_f}, t), & \text{if } \Delta \geq \frac{1}{2}. \end{cases} \tag{9}$$

Here, f_i^c denotes the particle distribution along link i after the collision step, but before propagation, and $\Delta = \frac{\|\mathbf{x_f} - \mathbf{x_w}\|}{\|\mathbf{x_f} - \mathbf{x_b}\|}$. Our 3D simulations implement a different, computationally more expensive scheme based on linear extrapolation [20] with the following streaming rule:

$$\begin{aligned} f_{i'}(\mathbf{x_f}, t+1) &= (1-\chi) f_i^c(\mathbf{x_f}, t) + \chi f_i^{feq}(\mathbf{x_b}, t) \\ &\quad + 2W_i \rho \frac{\mathbf{c_{i'}} \cdot \mathbf{u_w}}{c_s^2}, \\ f_i^{feq}(\mathbf{x_b}, t) &= W_i \rho(\mathbf{x_f}, t) \left[1 + \frac{\mathbf{c_i} \cdot \mathbf{u_{bf}}}{c_s^2} + \frac{(\mathbf{c_i} \cdot \mathbf{u_f})^2}{2c_s^4} - \frac{\mathbf{u_f} \cdot \mathbf{u_f}}{2c_s^2} \right], \end{aligned} \tag{10}$$

where $f_i^{feq}(\mathbf{x_b}, t)$ is the equilibrium distribution for the fictitious fluid node $\mathbf{x_b}$ within the boundary of the object, weighted by the extrapolation (or interpolation) factor χ. The fluid velocity at the wall position $\mathbf{x_w}$ is denoted as $\mathbf{u_w}$ and equals the velocity of the boundary object. The extrapolated fluid velocity $\mathbf{u_{bf}}$ along lattice link $\mathbf{c_i}$ is not uniquely defined. Mei et al. propose the following choices [20], which retain the second-order accuracy of the boundary condition treatment and improve the computational stability:

$$\begin{aligned} \mathbf{u_{bf}} &= \begin{cases} \frac{\Delta - 1}{\Delta} \mathbf{u_f} + \frac{1}{\Delta} \mathbf{u_w}, & \text{if } \Delta \geq \frac{1}{2}, \\ \mathbf{u_{ff}}, & \text{if } \Delta < \frac{1}{2}, \end{cases} \\ \chi &= \begin{cases} \frac{2\Delta - 1}{\tau}, & \text{if } \Delta \geq \frac{1}{2}, \\ \frac{2\Delta - 1}{\tau - 2}, & \text{if } \Delta < \frac{1}{2}. \end{cases} \end{aligned} \tag{11}$$

The value of Δ is again defined as $\Delta = \frac{\|\mathbf{x_f} - \mathbf{x_w}\|}{\|\mathbf{x_f} - \mathbf{x_b}\|}$, which is illustrated in Fig. 1 for the 2D case.

3 The fD3Q13 Lattice

In LBM, both space and velocity are discretized [29]. It is expected that the spatial discretization will be smoothed on

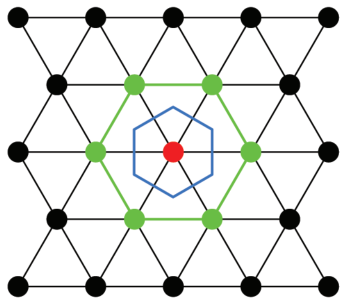


Fig. 2. Hexagonal 2D lattice, D2Q7. The six nearest neighboring sites (green circles) of a site (red circle) form a regular hexagon. The Voronoi cell (blue) is also a regular hexagon.

large enough scales compared to the grid size. However, it is not obvious whether the coarse angular discretization of the velocity space on the lattice links correctly models the macroscopic fluid behavior. It has been proven that the LBM recovers the Navier-Stokes equations at the incompressible limits through Chapman-Enskog expansion [29]. In this multiscale analysis, sufficient lattice symmetry is necessary to guarantee the isotropy of second and fourth-rank lattice tensors. The generalized lattice tensor is defined by

$$G_{\alpha_1\alpha_2\cdots\alpha_n} = \sum_i W_i c_{i\alpha_1} c_{i\alpha_2} \cdots c_{i\alpha_n}. \qquad (12)$$

Here, $\mathbf{c}_{i\alpha_j}$ is the α_jth Cartesian component of the lattice velocity $\mathbf{c}_i$. From this formulation, it follows directly that velocities with the same vector length will have the same weights. This can also be proven by solving (7). In single speed models such as D2Q7 on the hexagonal lattice and D3Q13 on the Cartesian cubic (CC) lattice, all the lattice links have the same length (1 and $\sqrt{2}$, respectively), and therefore, all nonzero velocities have the same weight.

The D2Q7 LBM is built on the hexagonal lattice, also called the triangular lattice. As shown in Fig. 2, every lattice site in the hexagonal lattice is linked to six nearest neighboring sites. The six neighbors form a regular hexagon and the Voronoi cell is also a regular hexagon. It has been proven that the hexagonal lattice is the optimal sphere packing in 2D [6]. Moreover, the hexagonal lattice has the maximum kissing number (or the number of nearest neighbors) in 2D, which means that it has the maximum number of velocity vectors of all possible single-speed 2D LBM. It was used in the first LGCA model that has been proven to recover the Navier-Stokes equation [12]. In comparison, the D2Q9 LBM is built on the Cartesian lattice, where every lattice site is connected to four nearest neighbors and four secondary neighbors. Because of the different link lengths (1 for nearest neighbors and $\sqrt{2}$ for secondary neighbors), the weights W_i used for the calculation of equilibrium distributions in (6) are also different.

In 3D, many lattices can be considered for LBM, such as CC, BCC, FCC, and hexagonal close packing (HCP) lattices. Fig. 3 shows the construction of the BCC, FCC, and HCP lattices. Traditionally, the D3Q13, D3Q15, and D3Q19 LBMs are the most frequently used models, all based on the CC lattice. The difference is the choice of lattice links. In the D3Q13 lattice, links connect only secondary neighbors. In the D3Q15 lattice, sites are connected to primary and tertiary

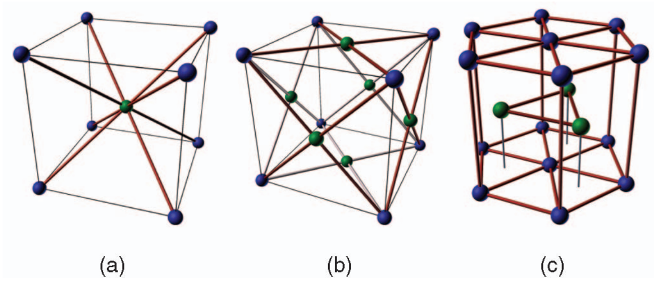


Fig. 3. (a) A 3D BCC lattice can be constructed by adding one lattice site at the center of every cell of the CC lattice. (b) A 3D FCC lattice can be constructed by placing sites at the centers of the square surfaces of every cell of the CC lattice. (c) A 3D HCP lattice can be constructed by layering 2D HCP lattices. In (c), the blue sites form the 2D HCP lattice, which is the first layer of the 3D HCP lattice. The green sites represent the second layer. The entire 3D HCP lattice contains alternating blue and green layers.

neighbors, and in the D3Q19 lattice, they are connected to primary and secondary neighbors. The CC lattice is prevailing because of its simplicity and because researchers have extensive knowledge of its properties. The FCC and BCC lattices have not been fully explored as the underlying lattice structures for LBM simulations.

In the D3Q13 LBM, it is observed that the lattice sites can be divided into two distinct half-sets [7]. One set contains all the sites where the sum of index components is even, which we call even sites. The complementary set contains all the odd sites and there does not exist any link between an odd site and an even site. The partitioning is illustrated in Fig. 4. Based on this observation, it is clear that an LBM simulation over a D3Q13 lattice can be decomposed into two independent simulations, each executed on an FCC lattice. In other words, for a given simulation configuration (same parameters, initial and boundary conditions), a simulation on a half-set of D3Q13 will produce identical results on the corresponding nodes of the full D3Q13 simulation. The reason is that during the streaming step, the physical properties of one site are calculated only from linked neighboring sites, and the collision step is a local computation at each lattice site. This feature is very important in the sense that we do not need to prove the validity of the LBM simulation on an FCC lattice. The D3Q13 LBM has been proven to recover the Navier-Stokes equation in the incompressible limit, which automatically leads to the conclusion that the LBM on an FCC

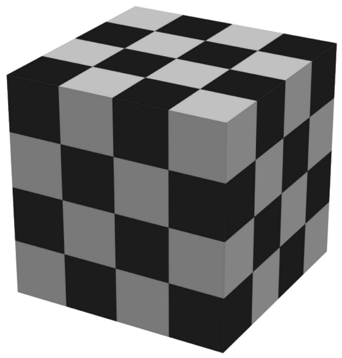


Fig. 4. A 4^3 CC lattice divided into two interleaving FCC lattices denoted by light and dark cells.

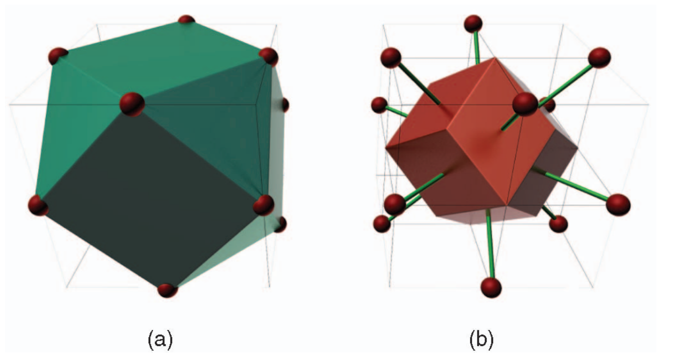


Fig. 5. (a) Unit cell and (b) Voronoi structures for the FCC lattice. The structures are the cuboctahedron and the rhombic dodecahedron, respectively.

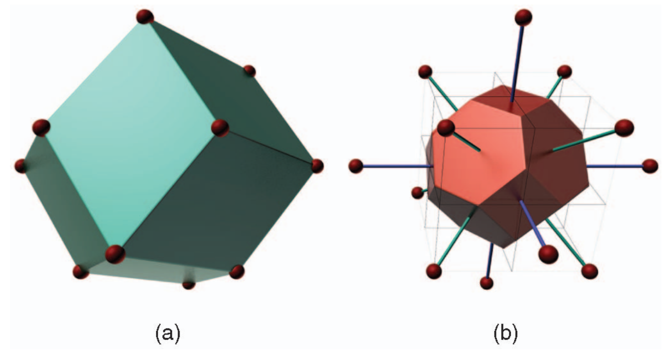


Fig. 6. (a) Unit cell and (b) Voronoi structures for the BCC lattice. The structures are the rhombic dodecahedron and the truncated octahedron, respectively.

lattice also recovers the Navier-Stokes equation. Furthermore, LBM is a mesoscopic method and the macroscopic physical values of the fluid field are statistically calculated from the particle distribution. Therefore, the LBM on the FCC lattice (the fD3Q13 LBM) can recover the macroscopic flow field with half the number of sites of the D3Q13 LBM, which means that the simulation speed can be doubled with little to no overhead. The possibility of using half the nodes of D3Q13 for LBM simulation has been presented by D'Humières et al. [7], and in this paper, we propose an efficient implementation that maps naturally to GPU architectures.

An FCC lattice achieves the optimal sphere packing in 3D (another optimal sphere packing in 3D is the asymmetric HCP) [6]. It is also the lattice with maximum kissing number, similarly to the hexagonal lattice in 2D. Among all possible single-speed LBMs in 3D, it is also the one with best angular discretization granularity of the velocity space. As shown in Fig. 5, the 12 neighbors of an FCC lattice site form a cuboctahedron and the Voronoi cell is a rhombic dodecahedron. Every lattice link (velocity vector) of the site intersects one rhombus on the Voronoi cell at the rhombus center.

Both the FCC and HCP structures can be constructed by layering 2D hexagonal lattices. Given a hexagonal lattice M on the plane $z=0$, the HCP lattice can be constructed as $\{M + 2i(0,0,\frac{\sqrt{6}}{3})|i \in Z\} \cup \{M + (2i+1)(\frac{1}{2},\frac{\sqrt{3}}{6},\frac{\sqrt{6}}{3})|i \in Z\}$ while the FCC is defined as $\{M + i(\frac{1}{2},\frac{\sqrt{3}}{6},\frac{\sqrt{6}}{3})|i \in Z\}$. In HCP, each lattice site also has 12 nearest neighbors, which is the maximum possible value in 3D and the lattice is an optimal sphere packing. However, for any link with direction $d = (d_x, d_y, d_z)$ such that $d_z \neq 0$, the link with direction $-d$ does not exist. Consequently, the HCP lattice does not satisfy the isotropy constraints and cannot be used in LBM simulations.

The BCC lattice can be constructed by adding an extra site at each cell of the CC lattice. BCC is the optimal lattice for sampling in 3D, but is not necessarily the best choice for LBM simulation. In BCC, each site has eight nearest neighbors and six secondary neighbors, and the unit cell is a rhombic dodecahedron as shown in Fig. 6. Thus, BCC and FCC are duals of each other and the Voronoi cell in BCC is a truncated octahedron. The faces on the truncated octahedron are eight regular hexagons (corresponding to eight nearest neighbors) and six squares (corresponding to six secondary neighbors). The distance from the site at the Voronoi cell center to the faces is not uniform, while in FCC, this distance is uniform. As a result, the BCC lattice is less isotropic than the FCC. Furthermore, the FCC lattice is capable of simulating the flow field with half the number of CC lattice sites, while the BCC reduces the site count by about 30 percent [1]. Therefore, the FCC is our lattice of choice for LBM and we call the resulting simulation model fD3Q13 LBM.

In any simulation of a practical fluid field problem, the geometric objects of boundary conditions must be discretized on the underlying lattice, which inevitably introduces some discretization error. Both the hexagonal lattice in 2D and the FCC lattice in 3D are better than CC lattices in representing geometric objects because of their better isotropy. Consider the vertices of the Voronoi cell in which the distance to sites is a local maximum: these are called *holes* [6]. If the distance from a point to the nearest site is the absolute maximum, it is called a deep hole, otherwise it is shallow. The maximum distance is also called the covering radius. Suppose the distance between two neighboring sites is 2, the covering radius is $\frac{2\sqrt{3}}{3} \approx 1.15$ for hexagonal lattice and $\sqrt{2} \approx 1.41$ for the FCC lattice, while the covering radius for the CC lattice is $\sqrt{2} \approx 1.41$ in 2D and $\sqrt{3} \approx 1.73$ in 3D. Intuitively, the smaller covering radius leads to less noisy and more isotropic representation of the discrete geometric object's surface. In Section 2.2, we have discussed the curved boundary treatment used in our simulation which is second order accurate. The precision of the extrapolation depends on the lattice link length. In hexagonal and FCC lattices, the link length is uniform, thus, the extrapolation precision does not vary with the link direction.

A hexagonal lattice in 2D is compactly stored in a 2D array of dimension (n_x, n_y) in row major order. For the ith lattice site on row j, its lattice coordinates are defined as (i, j) and it is the $j \times n_x + i$th element in the array. The coordinates in Cartesian space are computed by the following:

$$(x, y) = \left(i + \frac{1}{2}(j \bmod 2), \frac{\sqrt{3}}{2} j\right). \qquad (13)$$

Here, the mod 2 operation can be implemented with a bitwise AND instruction. The 6 neighboring lattice sites are enumerated in counterclockwise order:

$$(i+1, j), (i + j \bmod 2, j+1), (i + 1 + j \bmod 2, j+1),$$
$$(i-1, j), (i + 1 + j \bmod 2, j-1), (i + j \bmod 2, j-1).$$

There are many ways to generate the lattice sites for the FCC lattice. We adopt a checkerboard generation pattern illustrated in Fig. 4:

$$D_3 = \{(i, j, k) \in Z^n : (i + j + k) \bmod 2 = 0\}. \quad (14)$$

The pattern can be viewed as removing half of the lattice sites with odd index sums from the D3Q13 lattice. Similarly to the 2D hexagonal lattice, the FCC lattice is stored in a 3D array of dimension (n_x, n_y, n_z) in row-major format. Compared to the D3Q13 LBM of size (n'_x, n'_y, n'_z), each row of the lattice contains $n_x = n'_x/2$ lattice sites and the other two dimensions do not change: $n_y = n'_y$, $n_z = n'_z$. The index of a lattice site in the fD3Q13 LBM is defined to be the same as in the D3Q13 LBM. The lattice site (i, j, k) is the $\big(i/2 + j * n_x + k * n_x * n_y\big)$th element in the array. The index of the tth lattice site in the array is described by the following:

$$\begin{aligned} k &= t/(n_x \cdot n_y), \\ j &= (t - k \cdot n_x \cdot n_y)/n_x, \\ i &= (t - k \cdot n_x \cdot n_y - j \cdot n_x) \cdot 2 + (j + k) \bmod 2. \end{aligned} \quad (15)$$

With this scheme, the 12 neighbors of a lattice site (i, j, k) can be enumerated as $(i \pm 1, j \pm 1, k), (i, j \pm 1, k \pm 1)$, and $(i \pm 1, j, k \pm 1)$.

4 Validation

As described in Section 3, the D3Q13 LBM simulation can be decoupled into two independent LBM simulations on fD3Q13 lattices. The two fD3Q13 lattices are defined over the nodes with even and odd sums of index components, respectively. Therefore, the validity of the fD3Q13 LBM is proven by the validity of the D3Q13 LBM. We further compare the performance on the Cartesian and hexagonal lattices in 2D and the CC, FCC, and BCC lattices in 3D. The results of interest include stability of the simulation, convergence speed, and accuracy with respect to the known steady-state analytical solution or approximation.

4.1 Poiseuille Flow

We investigate the performance of the hexagonal D2Q7 and Cartesian D2Q9 lattices in an LBM simulation of the steady Poiseuille channel flow. For a channel with width $2L$ that supports a steady flow driven by a constant force F along the x-axis, the velocity is governed by the simplified Navier-Stokes equation [29]:

$$\nu \frac{d^2\mathbf{u}}{dy^2} + F = 0, \quad (16)$$

where ν is the fluid viscosity, and the boundaries along the x-axis at $y = -L$ and $y = L$ enforce a no-slip boundary condition. Then, the steady-state solution for the velocity is

$$\mathbf{u}(y) = \frac{F}{2\nu}\big(L^2 - y^2\big). \quad (17)$$

Our first experiment employs LBM to simulate the Poiseuille flow using both the D2Q9 and the hexagonal D2Q7 lattices. The simulation setup consists of solid walls across the top and bottom row of lattice sites and a periodic boundary condition along the x-axis. The boundary condition on the walls reduces to the simple bounce-back scheme since the boundary is aligned with the lattice links for both the D2Q7 and D2Q9 lattices. The flow is initially at rest and is accelerated by a constant force parallel to the x-axis, using the microscopic forcing method [29]. With this method, a force contribution is added to the velocity moment of the equilibrium distribution of each lattice node at each time step. Both simulations use approximately the same number of lattice sites and our hypothesis is that the hexagonal lattice will perform no worse than the cubic lattice in terms of convergence behavior and ability to reach the steady-state velocity profile given by the analytical solution.

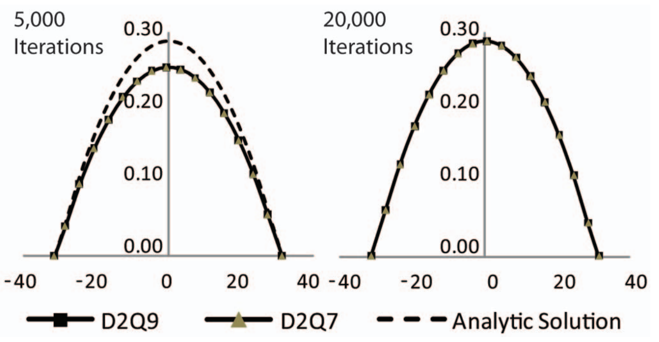


Fig. 7. Comparison of the velocity profiles for the Poiseuille Flow. The Cartesian and hexagonal lattices perform identically and both achieve the analytical result after 20,000 iterations.

In our experiments of Poiseuille flow, the D2Q9 domain contains 63×63 nodes with a channel width $2L = 62$. The viscosity is $\nu = 0.1667$ and the driving force is set to $F = 0.0001$. The equivalent D2Q7 domain contains 59×68 nodes and the LBM parameters are scaled appropriately. The initial fluid density is set to $\rho = 1$ and link densities are in equilibrium. The force is applied at each simulation step, slowly accelerating the fluid. As can be seen in Fig. 7, after 20,000 iterations, the simulations have reached a steady state and the velocities agree with the analytical velocity profile described by (17). Fig. 8a shows the velocity field rendered with the Line Integral Convolution (LIC) method [4].

In addition to the simple Poiseuille Flow, we have also implemented a simulation of a flow around a circular boundary. The boundary condition, in this case, uses a combination of bounce-back rules with first-order linear interpolation, which is able to handle curved boundaries [3]. We describe this technique in Section 2.2. Instead of a constant driving force, we now employ inlet and outlet boundary conditions along the left and right walls of the simulation domain, respectively. The inlet and outlet are implemented as Dirichlet boundary conditions, where the velocity moment along the boundary is reset to the initial value at each simulation step. Our investigation focuses on the ability of both lattices to resolve the vortex structures that form near the circular boundary.

For the simulation with circular boundary, we place a disc of diameter $d = 0.2$ at position $(0.3, 0.5)$. The disc parameters are normalized to the dimensions of the simulation domain. The viscosity is changed to $\nu = 0.04$ and the inlet velocity is set to $u = 0.1$. Fig. 8 shows the resulting steady-state flows using a GPU-accelerated version of the LIC method with additional streamline rendering to enhance the presentation of the vortices. The

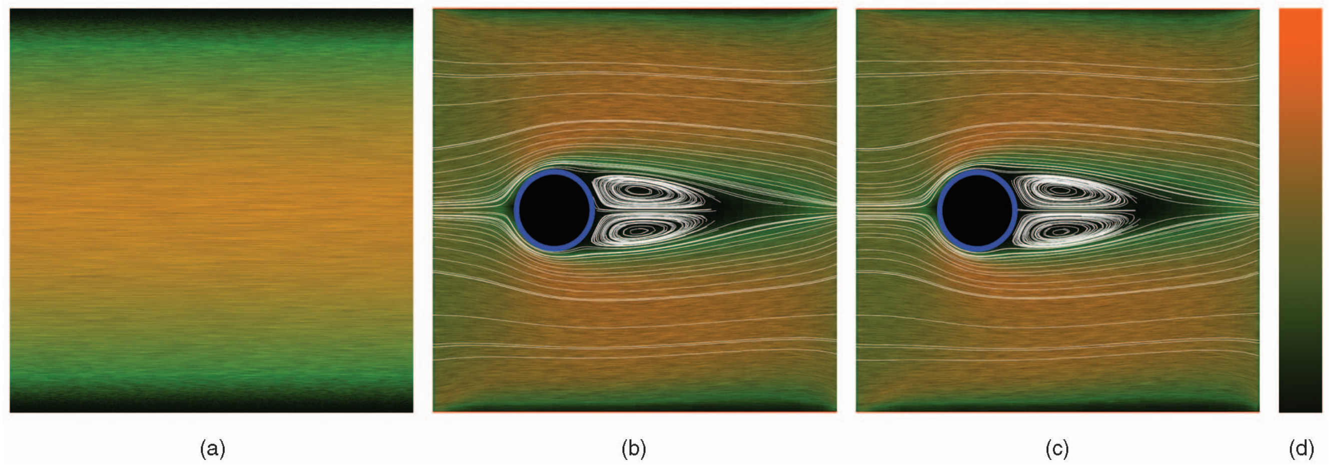


Fig. 8. (a) Illustration of the Poiseuille Flow after 20,000 iterations. The simulations based on D2Q7 and D2Q9 lattices produce identical results. (b) and (c) Simulation results for flow past a circular boundary in 2D for the D2Q9 and D2Q7 lattices, respectively. (d) Relative scale used for visualization of the speed of the flows.

hexagonal lattice produces flow results comparable to the Cartesian lattice while improving the speed of the simulation, as shown in Table 1, due to the decreased use of memory and computational resources.

4.2 Flow Past a Sphere

We investigate the performance of fD3Q13 LBM in simulating a flow moving past a sphere, which is a well-studied problem [11], [14], [15], [27]. It is one of the few setups in 3D for which there is a know-analytical approximation to the dimensionless drag coefficient if the simulation domain is an infinitely long circular pipe with a finite diameter D [11].

For this simulation, the Reynolds number is defined as $Re = \frac{U_0 d}{\nu}$, where U_0 is the steady speed of the sphere moving against a fluid with viscosity ν. The diameter of the sphere is d. $c'_d = \frac{24}{Re}$ is the drag coefficient for Stokes's Law region ($Re \leq 1$). We use the analytical approximation developed by Wham et al. [28], which accounts for the effect of the wall on a moving sphere for flows with $Re \in [1, 100]$. The drag coefficient c_d is then defined by

$$\frac{c_d}{c'_d} = \left(1 + 0.03708 Re^{1.514 - 0.1016 \ln Re}\right) \times \frac{1 - 0.75857\lambda^5}{1 - K\lambda + 2.0865\lambda^3 - 1.7068\lambda^5 + 0.72603\lambda^6}, \tag{18}$$

where $\lambda = \frac{d}{D}$ is the ratio of diameters for the sphere and the cylindrical domain and $K = 0.6628 + 1.458e^{-0.05635 Re}$. The drag force acting on the sphere is

TABLE 1
Comparison of LBM Performance on the D2Q7 and D2Q9 Lattices in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS)

Lattice Type	Nodes	SPTS (s)	LUPS (u/s)
D2Q7	4012	0.002172	1,847,145
D2Q9	4096	0.003376	1,213,270

Lattice sizes are 64×64 for the Cartesian lattices and 59×68 for the hexagonal lattice. Simulations are run on an Intel Core Duo 1.86 GHz processor.

$$F_d = c_d \frac{\rho U_0^2}{2} \pi \frac{d^2}{4}. \tag{19}$$

In our simulation, we compute the force that a density of particle exerts on the boundary by considering the change of momentum along a link that intersects that boundary. Suppose we have a link m that originates from a lattice site at position $\mathbf{x_f}$ toward a position $\mathbf{x_b}$ inside the boundary. The link has index i in the lattice structure, velocity $\mathbf{c_i}$, and link i' is the opposite in the lattice. Then, the force F_m exerted on the boundary by momentum exchange on link m is

$$F_m = -c_i\left(f_i(x_f, t + \Delta t) + f_{i'}(x_f, t)\right). \tag{20}$$

The total force acting on the boundary can be computed by integrating the force over the surface, which, in our discrete case, corresponds to finding the sum of the individual force contributions from each link m that intersects the boundary:

$$F = V \sum_m F_m. \tag{21}$$

Note that in (20) and (21), we assume that each site represents a unit of the simulation domain. Therefore, to make the results consistent across different simulations, the force must be scaled by the volume V of the unit cell in the lattice.

We have performed LBM simulations based on the Cartesian lattices (D3Q13, D3Q15, D3Q19, D3Q17), the BCC lattice (D3bQ15) [1], and the FCC lattice (fD3Q13). The cubic and BCC lattices use the single-relaxation-time model, while the FCC lattice also employs the multiple-relaxation-time model (MRT-LBM) [8] to improve numerical stability. The additional computational overhead of MRT-LBM is offset by the performance advantage of using only half of the simulation nodes compared to the Cartesian lattices. This allows us to achieve numerical stability comparable to D3Q19 at reduced computational cost.

We model a channel of finite length $L = 64$ enclosed by a solid cylindrical boundary with a cross section diameter $D = 60$. Inlet and outlet boundary conditions are applied along the left and right simulation boundaries, respectively.

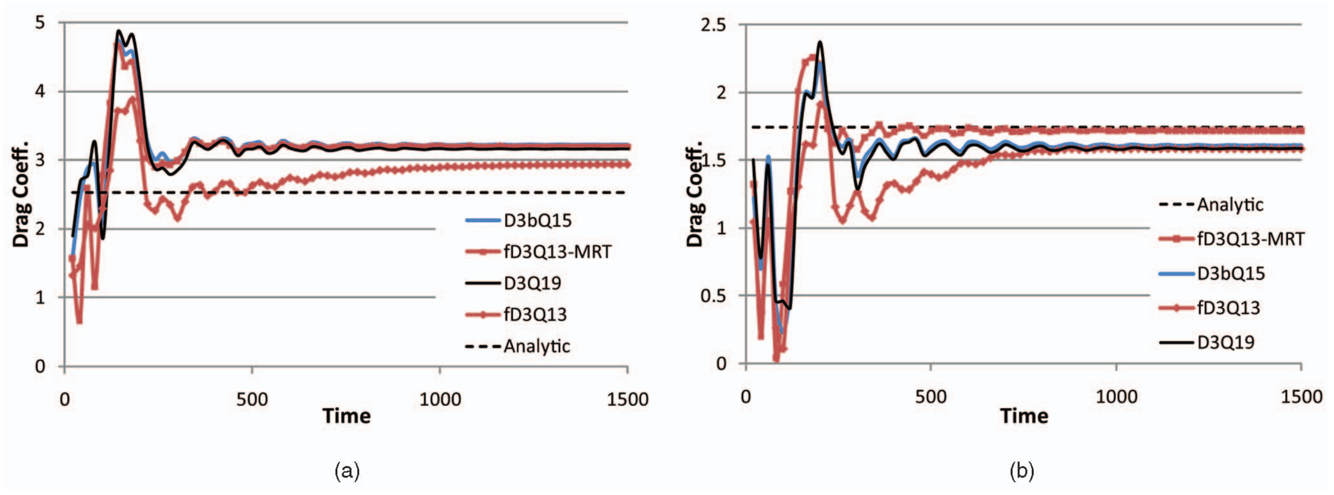


Fig. 9. Comparison with the steady-state analytical approximation for the drag coefficient of a sphere moving through a viscous media over time. (a) $Re = 40$ (b) $Re = 80$.

Additional velocity boundary conditions are enforced at the fluid interface on the pipe boundary and the sphere boundary through the u_w term in (10). The velocity of the sphere with diameter $d = 15$ is 0 and the velocity at the pipe boundary is U_0. The viscosity is set to $\nu = 0.0375$. This setup effectively models the steady movement of a sphere through a viscous media in an infinitely long pipe. We use (21) to calculate the total drag force acting on the sphere for varying Reynolds numbers. The dimensionless drag coefficient computed from (19) is then compared to the analytical approximation for the system, given by (18).

Fig. 9 shows the convergence toward the steady-state analytical approximation to the dimensionless drag coefficient. The $D3Q15$ and $D3Q27$ simulations yield results virtually equivalent to the $D3Q19$ simulation and are omitted from the graph. The simulation based on the BCC lattice also exhibits similar convergence characteristics with 30 percent less samples. The FCC lattice further reduces the number of samples and the fD3Q13-MRT simulation shows marginally faster convergence rates. It can also be expected that the MRT model will allow for greater numerical stability when $Re > 80$. The performance results for our LBM implementations with the different lattice types are shown in Table 2.

Fig. 10 uses a dense set of streamlines to visualize the vortices formed behind the sphere. The visualization component creates streamlines in a single slice parallel to the xz-plane. The slice is slightly away from the centerline of the cylindrical domain in order to capture the three-dimensional nature of the velocity field around the sphere. Additional streamline seeds are placed behind the sphere to enhance the rendering of the vortices.

5 Smoke Simulation in Urban Environment

We demonstrate the high performance of LBM based on the fD3Q13 lattice in a simulation of smoke propagation in an urban environment. A GPU-accelerated LBM simulation coupled with an OpenGL renderer allows for interactive exploration of a city environment encoded by a hierarchical definition and a texture library.

5.1 Urban Modeling

Our model captures intrinsic features of an urban environment using a hierarchical, grammar-based representation. At the highest level, there are three basic primitives: buildings, sidewalks, and roads. We take advantage of the regular structure found in metropolitan areas by modeling buildings as rectangular boxes and the roads and sidewalks as layers of polygons. This simplicity of representation allows for efficient storage of large data sets and rapid rendering. Additional details are inserted into the scene by rendering their respective subhierarchies to textures, which are then mapped onto the primitives defined at the higher levels. For example, our facade grammar defines features such as brick type, storefront, window and balcony structure, etc. A library of textures is then used to create a novel appearance or model the facade based on real-world images.

We use a prototype OpenGL-based renderer with render-to-texture capabilities and support for shaders written with NVIDIA's CgFX language. At runtime, we traverse the grammar file and render the elements of each facade to a texture. All textures are kept locally on the GPU and texture resolution is based on available memory. Certain elements in the facade grammar have associated CgFX shaders, which are executed during the rendering

TABLE 2
Comparison of LBM Performance on Different Lattice Types in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS)

Lattice Type	Nodes	SPTS (s)	LUPS (u/s)
fD3Q13	131,072	0.20892	627,379
D3Q13	262,144	0.41114	637,603
D3Q15	262,144	0.44122	594,134
D3bQ15	184,275	0.38581	477,631
D3Q19	262,144	0.54297	482,796
D3Q27	262,144	0.70180	373,531
fD3Q13-MRT	131,072	0.43914	298,474
D3Q15-MRT	262,144	1.04203	251,570
D3Q19-MRT	262,144	1.44079	181,945

Lattice sizes are $64 \times 64 \times 64$ for the Cartesian lattices, $32 \times 64 \times 64$ for the FCC lattice, and $45 \times 45 \times 91$ for the BCC lattice. Simulations are run on an Intel Core Duo 1.86 GHz processor.

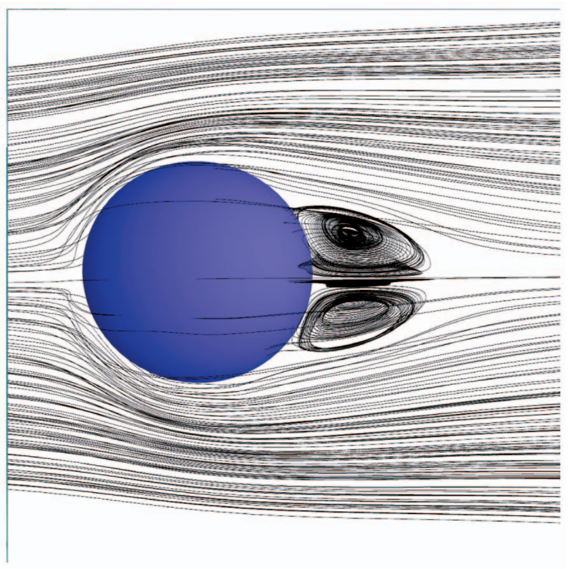


Fig. 10. Visualization of vortices in the flow behind the sphere at $Re = 80$. $\lambda = \frac{d}{D} = \frac{15}{60}$, where d is the diameter of the sphere and D is the diameter of the cylindrical boundary.

stages. For this project, we have implemented a simple reflection and refraction effect, which is used to model the appearance of glass. It is associated with windows and glass sections on doors and storefronts.

5.2 Modeling of Smoke in Urban Environment

We use the MRT-LBM [8] to simulate the airflow in the urban environments which achieves better numerical stability compared with the commonly used SRT-LBM [26], [24]. The idea of the MRT-LBM is to transform the particle distributions from phase space (i.e., the space of the distributions f_i) to the space of hydrodynamic moments (i.e., density, momentum, energy, etc.) and to perform the collision step in the moment space. As in the SRT-LBM model, the effect of collisions is approximated by a relaxation toward an equilibrium state, but in the moment space, each moment is allowed to relax individually. The transformations between the phase space and the moment space are given by

$$\begin{aligned} |f\rangle &= (f_0, f_1, \dots, f_n)^T, \\ |m\rangle &= (m_0, m_1, \dots, m_n)^T, \\ |m\rangle &= M|f\rangle, f\rangle = M^{-1}|m\rangle. \end{aligned} \tag{22}$$

Here, the Dirac notation $|.\rangle$ is used to denote column vectors, T denotes the transpose, n is the number of the distributions ($n = 13$ for the fD3Q13 LBM), and M is an $n \times n$ transformation matrix. In MRT-LBM, the collision step becomes

$$|f(\mathbf{x}, t + \Delta t)\rangle - |f(\mathbf{x}, t)\rangle = -M^{-1}S[|m(\mathbf{x}, t)\rangle - |m^{eq}(\mathbf{r}, t)\rangle], \tag{23}$$

where the components of the vector $|m^{eq}\rangle$ are the local equilibrium values of the moments. The matrix S in the collision equation is a diagonal matrix $S \equiv diag(s_0, s_1, \dots, s_n)$ whose elements are the relaxation rates. Their values are directly related to the kinematic shear and bulk viscosities [8].

The improved stability of the MRT-LBM allows thermal effects to be coupled in, resulting in the hybrid thermal LBM (HTLBM) [17]. Zhao et al. [30] have used the HTLBM to model the weakly compressible thermal flow for shimmering effects. We adopt it for smoke simulation in an urban environments. To capture thermal effects, temperature is coupled to the MRT-LBM through the energy moment, meaning that the energy equilibrium is modified during the calculation of equilibrium distributions at each time step.

The row of the transformation matrix M relevant to the computation of the energy moment is constructed as follows:

$$M_4 = \langle \frac{13}{2}\mathbf{c_i} \cdot \mathbf{c_i} - 12| = (-12, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), \tag{24}$$

where $\mathbf{c_i}$ is a lattice velocity and $\langle .|$ denotes a row vector. The energy moment is computed through (23) and the equilibrium energy moment for the HTLBM model is defined by:

$$m_4^{eq} = e^{eq} n_1 (c_{s0}^2 - n_2)\rho + n_3(n_4 - \gamma)\mathbf{u} \cdot \mathbf{u} + q_1 T. \tag{25}$$

The new variables in (25) are the temperature T, its constant coupling coefficient q_1, and the specific heat γ. The parameters n_1 to n_4 are constants and their values are determined by the linear stability analysis. The constant c_{s0} is the isothermal speed of sound, which is specific to the chosen lattice velocity set. Heat transfer is modeled separately with a standard advection-diffusion equation:

$$\partial_t T + \mathbf{u} \cdot \nabla T = \kappa \Delta T + q_2(\gamma - 1)c_{s0}^2 \nabla \cdot \mathbf{u}. \tag{26}$$

The parameter κ is the thermal diffusivity of the fluid and q_2 is another constant coupling coefficient. The advection computation executes after the streaming step in LBM and modifies the temperature used in the calculation of the energy moment in the following simulation step. Finally, the evolution of the smoke density volumetric data set is also modeled by an advection-diffusion equation, which is computed by a backtracing algorithm [23]. We use the fluid velocity computed at each lattice site to determine a sampling position for the backtracing density value. The monotonic cubic interpolation [10] is used to compute densities at off-grid sampling positions and we have also implemented the lower order trilinear interpolation for cases that require higher performance at the expense of visual fidelity. The advection-diffusion process for the smoke density ρ_s is described by the following:

$$\partial_t \rho_s + \mathbf{u} \cdot \nabla \rho_s = 0. \tag{27}$$

Here, the flow velocity $\mathbf{u}$ is obtained from the HTLBM. Based on the smoke density, a gravity force is applied to the HTLBM:

$$f_{body} = -\alpha \rho_s y + \beta(T - T_0)\hat{y}. \tag{28}$$

In this equation, $\hat{y}$ is the unit upward vector, α and β are the coefficients of gravity and buoyancy, and T_0 is the temperature of the surrounding air. The force is applied through the velocity moment of the equilibrium distribution, calculated in (23). More details are available in the paper on HTLBM [17] and in the work of Zhao et al. [30].

Unlike Zhao et al. [30] who have implemented the D3Q13 HTLBM on the CC grid on the GPU, we implemented the FCC HTLBM on the GPU. Half of storage consumption and almost half of the computations are therefore saved. As described in Section 3, the FCC lattice can be represented as a 3D array and the indices of lattice sites can be calculated with (15). Therefore, a natural choice is to represent the lattice data on the GPU as a set of 3D textures. The 13 particle distributions f_i are stored in the color channels of four 3D textures, the elements of which have 4, 4, 4, and 1 components, respectively. The post-collision particle distributions $f_i(\vec{\mathbf{x}}, t^+)$, moments m_i, and equilibrium moments m_i^{eq} are also stored in this way. The flow velocity $\mathbf{u}$ and density ρ, and the temperature and smoke density are stored in two additional 3D textures.

Each simulation step includes a series of computation substeps, such as computing the macroscopic flow velocity and density, computing the equilibrium moments, performing particle collision and streaming, applying buoyancy and gravity body forces, and advecting the density and the temperature fields with the LBM flow field. The substeps involve only data from the local neighbors of a site in the FCC lattice and the operations are explicitly parallel. They are implemented as computation kernels in OpenGL fragment programs, each of which operates on the lattice in SIMD fashion and updates the corresponding lattice data. The fragment program samples the lattice data from the 3D textures, computes the new lattice data, and renders slices of the output back into the 3D textures.

To handle the boundary conditions of objects in the flow, we calculate the intersection points at which the lattice links are cut by the object surfaces and use this information to locally modify the particle distributions. In preprocessing, the intersection points of the object surface and lattice links are computed and the related boundary information is stored as a vertex buffer object. During simulation, we trigger the boundary condition computation kernel on the intersected links by rendering this vertex buffer object to the frame buffer. This operation performs the calculations described in (10) on the particle distributions stored in textures on the GPU.

5.3 Rendering

The fD3Q13 LBM simulation produces the smoke density volume on an FCC lattice. Qiu et al. [22] have proposed a framework for volumetric global illumination on the FCC lattice volume, which is capable of capturing multiple scattering. However, although this method is much faster than conventional methods such as radiosity and photon mapping, it does not achieve real-time performance. Because the simulation is performed on the GPU at interactive speed with results residing on the local video memory, we would like to also use the GPU for rendering the smoke in the urban environment. A major difficulty of rendering FCC volumes is that a computationally intensive filter such as Gaussian filter is necessary for reconstruction [22], which is slow even on the GPU. Entezari has proposed box splines for reconstruction of data sampled on the FCC lattice [9], however, we are not aware of any GPU

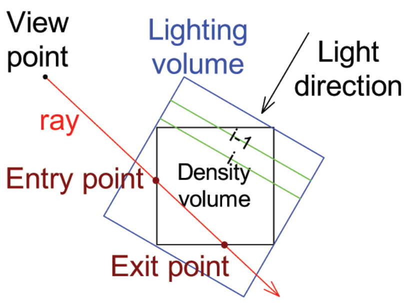


Fig. 11. Every slice (green lines) of the lighting volume (blue box) is perpendicular to the light direction. The lighting volume stores light intensity and density sampled from the density volume (black box). The light intensity of slice i is calculated by attenuating slice $i-1$ with the current opacity values. The ray is traced through the lighting volume. The entry point and exit point of the ray are on the bounding box of the density volume.

implementations. Therefore, in this application, we first convert the FCC lattice volume into a rectilinear volume. This allows us to take advantage of the texturing unit in current GPUs for efficient trilinear interpolation. Note that the index of a lattice site in the fD3Q13 LBM is defined to be the same as in the D3Q13 LBM. For a sampling point in the regular volume with index (i,j,k), if $(i+j+k) \bmod 2 = 0$, the density value is copied from the fD3Q13 LBM lattice. Otherwise, the density value is interpolated from neighboring even points:

$$d(i,j,k) = \frac{1}{6}(d(i \pm 1, j, k) + d(i, j \pm 1, k) + d(i, j, k \pm 1)). \tag{29}$$

We use ray tracing with single scattering for volume rendering. The illumination $I(P, \omega')$ arriving at point P for a directional light source in the direction $-\omega'$ is calculated and stored in a lighting volume. As shown in Fig. 11, the lighting volume is the minimal enclosing box of the density volume such that slices of the lighting volume are perpendicular to the light direction. The light intensity $L(p_i)$ at point p_i on slice i is computed by attenuating $L(p_{i-1})$ on slice $i-1$ with the dispersion opacity $\tau(p_{i-1})$. In our application, the objects cast shadows on the volume. Because the objects are opaque, $L(p)$ is 0 if p is occluded by some objects from the light source. This can be accomplished by calculating the shadow volume of the geometric mesh for the light source and using it in the fragment program to modulate the light intensity.

One approach to calculating the lighting volume involves a multipass algorithm using the OpenGL pipeline. In a preprocessing pass, the polygonal mesh of the buildings is rendered and only the surface depth is stored in the depth buffer. This pass can be executed very efficiently since modern GPUs double their effective pixel fill rate when writing only to a depth buffer. In each of the following passes, a single slice of the lighting volume is calculated and stored in a 3D texture. For each fragment, we compare the depth with the surface depth so that occluded fragments have a zero lighting value. Due to hardware and OpenGL limitations, the fragment program cannot simultaneously read and write the 3D texture of the lighting volume.

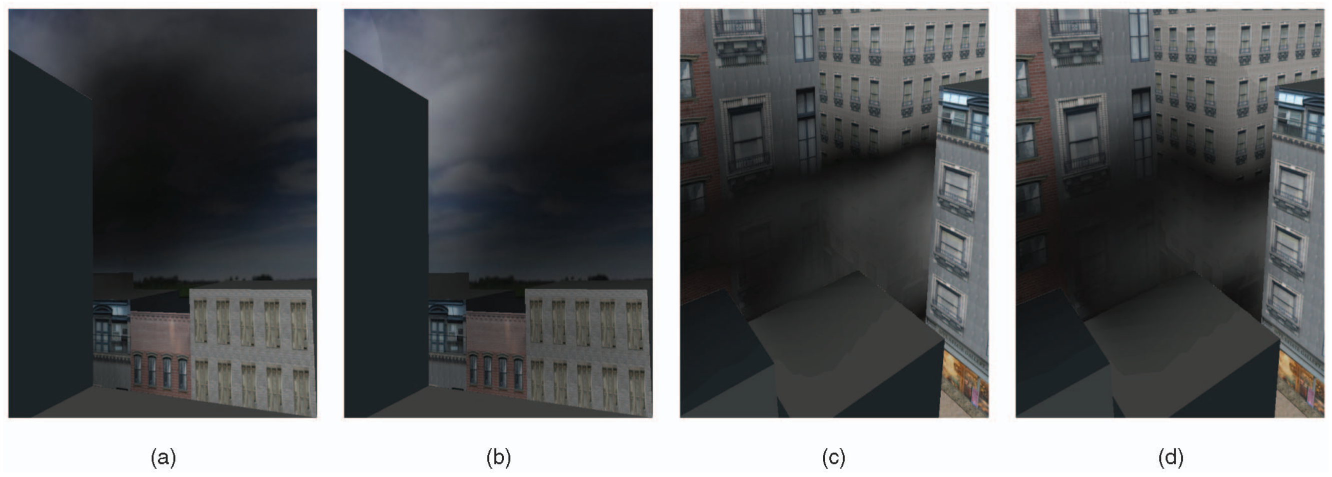


Fig. 12. Snapshots of smoke simulation in the urban environment.

Therefore, in each pass, we sample the light intensity of the previous slice from the 3D texture and the result is used to compute the intensities of the current slice. After this operation, the current slice is copied from the frame buffer to the 3D texture. This algorithm can be implemented using only the fixed-function OpenGL pipeline [2], and can therefore run on 3D hardware without shading capabilities. Our initial implementation uses fragment shaders, but suffers from some inherent limitations of the algorithm nevertheless. During a single pass, every slice of the light volume is rendered to the frame buffer, copied to the 3D texture and sampled in the fragment program. A major performance limitation is that the OpenGL pipeline is stalled at the end of each rendering pass while waiting for the texture copy. This memory transfer also consumes the limited memory bandwidth of the GPU.

To address the performance issues, we have implemented a new method for calculating the lighting volume with NVIDIA's CUDA toolkit. CUDA is a C language environment for writing applications that execute in parallel on the processing units of supported NVIDIA graphics hardware [21]. The main advantage is that it alleviates some of the limitations of the graphics pipeline in respect to general purpose computations. In our application, it allows for memory scatter operations or the ability of a computation kernel to write to multiple memory addresses in a single thread. This is the exact feature needed to compute the lighting volume efficiently. For a volume of resolution $x \times y \times z$, we call a CUDA kernel with $x \times y$ threads with each thread processing one lighting ray. The kernel samples z voxels on the path of each ray using a loop and stores the resulting values in the lighting volume.

With a single kernel call, we obtain the entire lighting volume, which is much more efficient than the multipass OpenGL approach and relaxes the memory bandwidth requirements. Currently, CUDA cannot share buffers with OpenGL as they live in different contexts, and therefore, we need to copy the resulting volume data from the CUDA memory to an OpenGL pixel buffer object and then to a 3D texture. This process is much faster than the memory copy needed in the multipass algorithm since it only involves high-bandwidth GPU memory and does not use the comparatively slower PCI-Express or AGP buses for the data transfer.

Previous approaches trace rays in both the density volume and lighting volume. Each ray needs to maintain the current sampling position and ray direction in both volumes, and at each sampling point, the program performs two trilinear interpolations, which is inefficient. We propose a method of tracing rays in a single volume only. During the computation of the lighting volume, the CUDA kernel samples the density volume with trilinear interpolation. For each point p in the lighting volume, in addition to the light intensity $L(p)$, we also store the interpolated density $s(p)$. This reduces the number of texture sampling operation needed during the ray-casting pass.

Krüger and Westermann [16] have implemented ray-casting for volume rendering on the GPU with empty space skipping and early ray termination. However, empty space skipping requires an additional data structure (such as min-max octree) to be calculated and stored on the GPU. Our simulation generates different density volumes at each time step and recalculating the data structure for empty space skipping has a high cost compared to the rendering of a single frame. Also, because of the phenomena we model, the dispersion volume is highly transparent and the opacity of most rays is not saturated. As a result, early ray termination is not efficient in this case. Therefore, our implementation is a single-pass ray-casting algorithm without support for empty space skipping or early ray termination.

5.4 Results

Fig. 12 shows the result of our smoke simulation with a region in New York City, from 6th to 7th Avenue and from 57th to 59th Street. This urban model has six blocks and tens of buildings. An east wind blows through the urban environment and the hot smoke is released from an upwind position. Its motion is affected by the wind, buoyancy force due to the temperature, gravity due to the smoke density, and the geometry of the buildings.

The platform we use is a PC with a Geforce 8800 GTX graphics card with 768 MB memory and two Xeon 3.6 GHz CPUs (our CPU code is single-threaded). In Table 3, we compare three different implementations on the GPU: D3Q19 SRT LBM, D3Q13 MRT LBM, and our fD3Q13 MRT LBM. The simulation resolution for D3Q19 and D3Q13 is $128 \times 64 \times 64$. The simulation resolution for fD3Q13 is $64 \times 64 \times 64$, as only half of the lattice sites are needed for

TABLE 3
Comparison of Runtime Performance for Simulations Using D3Q19 SRT LBM, D3Q13 MRT LBM, and fD3Q13 MRT LBM

Simulation Type	Performance (Frames/second)
fD3Q13 MRT LBM	187
D3Q13 MRT LBM	108
D3Q19 SRT LBM	72

the FCC lattice. Performance does not increase linearly with the number on lattice sites due to fixed overhead associated with texture and data management on the GPU. Nevertheless, the fD3Q13 LBM is shown to be much more efficient than simulations using the two other lattice types. The major reason for this performance improvement is that fD3Q13 computation requires less memory access operations, while on the GPU, the memory bandwidth is the dominant factor for computational performance. With the fD3Q13 MRT LBM, our simulation runs at 187 FPS. With the GPU rendering incorporated into the system, the simulation runs at 66 FPS. This allows for real-time navigation in the urban environment.

6 Conclusion

We have presented the fD3Q13 lattice for highly efficient LBM simulations. The main advantage of the method is that LBM on the fD3Q13 lattice can recover macroscopic fluid properties with half the lattice sites used with D3Q13. With a low number of lattice links, an efficient indexing scheme and a demonstrated GPU implementation, the fD3Q13 lattice is a preferred choice for use in a GPU-based real-time simulations and rendering pipeline.

The high performance also allows the implementation of MRT-LBM, which improves the stability of the simulation and allows the coupling of thermal effects. The added computational load has a minimal effect when implemented on a GPU architecture whose performance is mainly restricted by memory bandwidth. Simulations on FCC lattices are faster than on D3Q19 while offering similar stability characteristics. Savings on memory bandwidth consumption because of the reduced number of lattice nodes and links can dramatically improve performance on the GPU even with increased computational load.

In addition to more advanced simulation techniques, the efficient base simulation frees computational resources for improved advection-diffusion and rendering techniques. We can use a more advanced advection scheme than the semi-Lagrangian method used in our current implementation, which will improve the smoke detail resolution in our renderings. Qiu et al. [22] have proposed a technique for volumetric lighting based on tracing photons on an FCC lattice. We can implement this technique with CUDA to achieve interactive photon mapping with multiple scattering on a unified lattice type. Also, a CUDA-based renderer will remove the need to copy the lighting volume in video memory, further increasing the performance of our framework.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) CCF-0702699 grant.

References

[1] U.R. Alim, A. Entezari, and T. Möller, “The Lattice-Boltzmann Method on Optimal Sampling Lattices,” *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 4, pp. 630-641, July/Aug. 2009.

[2] U. Behrens and R. Ratering, “Adding Shadows to a Texture-Based Volume Renderer,” *Proc. IEEE Symp. Volume Visualization,* pp. 39-46, 1998.

[3] M. Bouzidi, M. Firdaouss, and P. Lallemand, “Momentum Transfer of a Boltzmann-Lattice Fluid with Boundaries,” *Physics of Fluids,* vol. 13, no. 11, pp. 3452-3459, 2001.

[4] B. Cabral and L.C. Leedom, “Imaging Vector Fields Using Line Integral Convolution,” *Proc. ACM SIGGRAPH '93,* pp. 263-270, 1993.

[5] S. Chen and G.D. Doolen, “Lattice Boltzmann Method for Fluid Flows,” *Ann. Rev. Fluid Mechanics,* vol. 30, pp. 329-364, 1998.

[6] J.H. Conway, N.J.A. Sloane, and E. Bannai, *Sphere-Packings, Lattices, and Groups.* Springer-Verlag, 1987.

[7] D. D’Humières, M. Bouzidi, and P. Lallemand, “Thirteen-Velocity Three-Dimensional Lattice Boltzmann Model,” *Physical Rev. E,* vol. 63, p. 066702, 2001.

[8] D. D’Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo, “Multiple-Relaxation-Time Lattice Boltzmann Models in Three Dimensions,” *Royal Soc. London Philosophical Trans. Series A 360,* vol. 1792, pp. 437-451, 2002.

[9] A. Entezari, “Optimal Sampling Lattices and Trivariate Box Splines,” PhD thesis, Simon Fraser Univ., 2007.

[10] R. Fedkiw, J. Stam, and H.W. Jensen, “Visual Simulation of Smoke,” *Proc. ACM SIGGRAPH '01,* pp. 15-22, 2001.

[11] Z.-G. Feng and E.E. Michaelides, “Hydrodynamic Force on Spheres in Cylindrical and Prismatic Enclosures,” *Int'l J. Multiphase Flow,* vol. 28, no. 3, pp. 479-496, 2002.

[12] U. Frisch, B. Hasslacher, and Y. Pomeau, “Lattice-Gas Automata for the Navier-Stokes Equation,” *Physical Rev. Letters,* vol. 56, no. 14, pp. 1505-1508, 1986.

[13] S. Hou, J. Sterling, S. Chen, and G.D. Doolen, “A Lattice Boltzmann Subgrid Model for High Reynolds Number Flows,” *Computer,* pp. 1004-1022, 1994.

[14] T.A. Johnson and V.C. Patel, “Flow Past a Sphere up to a Reynolds Number of 300,” *J. Fluid Mechanics,* vol. 378, pp. 19-70, 1999.

[15] D. Kim, H. Choi, and H. Choi, “Characteristics of Laminar Flow Past a Sphere in Uniform Shear,” *Physics of Fluids,* vol. 17, no. 10, p. 103602, 2005.

[16] J. Krüger and R. Westermann, “Acceleration Techniques for GPU-Based Volume Rendering,” *Proc. IEEE Visualization Conf.,* pp. 38-45, 2003.

[17] P. Lallemand and L.-S. Luo, “Theory of the Lattice Boltzmann Method: Acoustic and Thermal Properties in Two and Three Dimensions,” *Physical Rev. E,* vol. 68, no. 3, pp. 036706-+, 2003.

[18] J. Latt, “Choice of Units in Lattice Boltzmann Simulations,” technical report, LBMethod.org, 2008.

[19] R.S. Maier, R.S. Bernard, and D.W. Grunau, “Boundary Conditions for the Lattice Boltzmann Method,” *Physics of Fluids,* vol. 8, no. 7, pp. 1788-1801, 1996.

[20] R. Mei, W. Shyy, D. Yu, and L.-S. Luo, “Lattice Boltzmann Method for 3-d Flows with Curved Boundary,” *J. Computational Physics,* vol. 161, no. 2, pp. 680-699, 2000. 350006.

[21] NVIDIA, *Compute Unified Device Architecture Programming Guide,* 2008. Version 2.0, http://www.nvidia.com/object/cuda_develop.html, 2009.

[22] F. Qiu, F. Xu, Z. Fan, N. Neophytos, A. Kaufman, and K. Mueller, “Lattice-Based Volumetric Global Illumination,” *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1576-1583, Nov./Dec. 2007.

[23] J. Stam, “Stable Fluids,” *Proc. ACM SIGGRAPH '99,* pp. 121-128, 1999.

[24] N. Thuerey and U. Ruede, “Free Surface Lattice-Boltzmann Fluid Simulations with and without Level Sets,” *Proc. Vision, Modeling, and Visualization Conf.,* pp. 199-207, 2004.

[25] R. Verberg and A.J. C. Ladd, “Lattice-Boltzmann Model with Sub-Grid-Scale Boundary Conditions,” *Physical Rev. Letters,* vol. 84, no. 10, pp. 2148-2151, 2000.

[26] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and A. Kaufman, “Lattice-Based Flow Field Modeling,” *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 6, pp. 719-729, Nov./Dec. 2004.

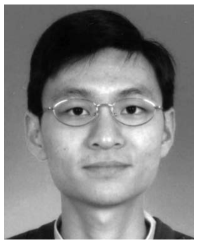
[27] S.-B. Wen and C.-L. Lai, "Theoretical Analysis of Flow Passing a Single Sphere Moving in a Micro-Tube," *Proc. Royal Soc. Math. Physical and Eng. Sciences,* vol. 459, no. 2030, pp. 495-526, 2003.

[28] R.M. Wham, O.A. Basaran, and C.H. Byers, "Wall Effects on Flow Past Solid Spheres at Finite Reynolds Number," *Industrial & Eng. Chemistry Research,* vol. 35, no. 3, pp. 864-874, 1996.

[29] D.A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction,* first ed. Springer-Verlag, 2000.

[30] Y. Zhao, Y. Han, Z. Fan, F. Qiu, Y.-C. Kuo, A. Kaufman, and K. Mueller, "Visual Simulation of Heat Shimmering and Mirage," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 1, pp. 179-189, Jan./Feb. 2007.

Kaloian Petkov received the BA degree in computer science and mathematics from Lake Forest College in 2006. He is currently working toward the PhD degree in computer science at Stony Brook University. His research focuses on computer graphics, volume rendering, natural phenomena modeling and visualization, and general-purpose computing on graphics hardware. For more information, see http://www.cs.sunysb.edu/~kpetkov.



Feng Qiu received the PhD degree in computer science at the Department of Computer Science at Stony Brook University in 2008 and is currently a research scientist at Siemens Corporate Research. His research interests are focused on volume graphics, medical visualization, and general-purpose computation on GPUs and GPU clusters. He has authored and coauthored more than 20 journal and conference papers. He is a member of the IEEE and the IEEE Computer Society.

Zhe Fan received the BS degree in computer science from the University of Sciences and Technology of China in 1998, the MS degree in computer science from Chinese Academy of Sciences in 2001, and the PhD degree in computer science from Stony Brook University in 2008. He is currently working with Google Inc. His research is focused on flow simulation and visualization on GPUs and GPU clusters.

Arie E. Kaufman received the BS degree (1969) in mathematics and physics from the Hebrew University of Jerusalem, Israel, the MS degree (1973) in computer science from the Weizmann Institute of Science, Rehovot, Israel, and the PhD degree (1977) in computer science from the Ben-Gurion University, Israel. He is a distinguished professor and chair of the Computer Science Department, chief scientist of the Center of Excellence in Wireless and Information Technology (CEWIT), and the director of the Center for Visual Computing (CVC) at the State University of New York at Stony Brook (Stony Brook University). He is fellow of the IEEE and the IEEE Comuputer Society and the recipient of IEEE visualization career award (2005). He further received the IEEE outstanding contribution award (1995), ACM service award (1998), IEEE CS meritorious service award (1999), member of the European Academy of Sciences (since 2002), State of New York entrepreneur award (2002), IEEE Harold Wheeler award (2004), and State of New York innovative research award (2005). He was the founding editor-in-chief of *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 1995-1998. He has been the cofounder, papers/program cochair, and member of the steering committee of *IEEE Visualization Conferences*; cofounder/chair of *Volume Graphics Workshops*; cochair for *Eurographics/SIGGRAPH Graphics Hardware Workshops*, the papers/program cochair for *ACM Volume Visualization Symposia*. He previously chaired and is currently the director of IEEE CS technical committee on visualization and graphics. He has conducted research and consulted for over 35 years specializing in volume visualization, graphics architectures, algorithms, and languages, virtual reality, user interfaces, multimedia, medical imaging and their applications. For more information, see http://www.cs.sunysb.edu/~ari.

Klaus Mueller received the MS degree in biomedical engineering in 1991 and the PhD degree in computer science in 1998, both from the Ohio State University. He is currently an associate professor in the Computer Science Department at Stony Brook University, where he also holds coappointments in the Biomedical Engineering and Radiology Departments. His current research interests are computer and volume graphics, visualization, visual analytics, medical imaging, and computer vision. He won the US National Science Foundation CAREER award in 2001 and has served as a cochair at various conferences, such as IEEE Visualization, Volume Graphics Symposium, and the Fully 3D Workshop on High-Performance Image Reconstruction. He has authored and coauthored more than 100 journal and conference papers, and has participated in 15 tutorials at international conferences on various topics in visualization and medical imaging. He is a senior member of the IEEE and the IEEE Computer Society. For more information, see http://www.cs.sunysb.edu/~mueller.