

Adapted Unstructured LBM for Flow Simulation on Curved Surfaces

Z. Fan, Y. Zhao, A. Kaufman, Y. He

{fzhe, yezhao, ari, yhe}@cs.sunysb.edu
Stony Brook University

Abstract

Flow motion on curved surfaces of arbitrary topology is an interesting visual effect but a complex dynamics to simulate. In this paper, we introduce a novel and effective way to model such dynamics. We propose a technique that adapts a recently emerged computational fluid dynamics (CFD) model, unstructured lattice Boltzmann model (Unstructured LBM), from the 2D unstructured meshes to the 3D surface meshes. Unlike previous methods in modeling flows on surfaces, which start from the macroscopic point of view and modify the Navier Stokes solvers for the curved surfaces, our method is based on the microscopic kinetic equations for discrete particle distribution functions. All computations on the surface mesh only involve the information within local neighborhoods. This model lends itself the following advantages: (i) simplicity and explicit parallelism in computation, (ii) great capability in handling complex interactions, such as the interactions between flow and boundaries and the interactions of multiple-component fluids; (iii) no need of global surface parameterization which may cause strong distortions; (iv) capability of being applied to meshes with arbitrary connectivity.

Categories and Subject Descriptors (according to ACM CCS): 1.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation

1. Introduction

In 2003, Stam [Sta03] brought an interesting question to the graphics community: how would a fluid flow look like on arbitrary surfaces? Although such a phenomenon would only exist in an imaginary world, its visual simulation can create interesting special effects on the 3D surface models and is surely desirable for computer graphics applications. Furthermore, shallow (thin) fluid flows are actually often seen in the real-world, such as the swirling pattern on soap bubbles, the atmosphere on the earth, and the lava drifting from the peak of the mountain. Reducing their visual simulations from 3D flows to surface flows can simplify the model and make the computation affordable.

Modeling the flow on arbitrary curved surfaces is a challenging problem, and most previous flow models only solve the regular domains, such as 3D volumes, 2D grids or the sphere. Stam [Sta03] has firstly proposed a solution that realistically simulated fluid flows on curved surfaces of arbitrary topologies. He has extended his 2D fluid solver to the curvilinear coordinates over Catmull-Clark surfaces. How-

ever, this technique requires the global surface parameterization that may cause visible distortions in the flow. Shi and Yu [SY04] have presented a method that performs the in-viscid and incompressible flow simulation directly on surface meshes with only local parameterizations. Their model works for the triangular mesh, which is more general in terms of the mesh connectivity. Note that both above methods have been started from a macroscopic point of view to globally solve the Navier Stokes over 3D surface meshes.

In this paper, we demonstrate the direct simulation method based on the microscopic view as an excellent candidate to solve the original problem. In our microscopic method, which is based on the lattice Boltzmann model (LBM), computing the flow properties at each mesh point only involves the information within its nearest local neighborhood on the surface mesh. This makes the complex problem much simpler. The LBM is a relative new and increasingly popular CFD method [Suc01]. It follows the prescribed, statistic Boltzmann equations for particle distribution functions and has been proved to recover the Navier-

Stokes equations at the macroscopic level. Although the traditional LBM is devised on regular grids, the unstructured LBM method on 2D unstructured grids has been developed by computational physicists recently [UBS03, US04]. Our idea is to adapt this unstructured LBM from 2D meshes to 3D surfaces meshes. As we will show in Section 4, such an adaptation is not trivial and we propose an elegant technique to fulfill this goal. Our method results in a model that is simple in implementation and explicitly parallel in computation. Inheriting the advantages from the LBM, it can handle microscopic physical interactions with ease, such as the interaction of the flow with static or moving boundaries (see Figure 8 and 9) and the mixture of multiple-component fluids (see Figure 10). Our method does not require global surface parameterization. In addition, it has no special requirement for the connectivity of the surface meshes. It is derived for the triangular meshes and can be extended to other kinds of meshes as well.

The remainder of this paper is organized as follows: Section 2 briefly reviews the related work in physically-based flow modeling. Section 3 introduces the background of LBM and unstructured LBM. In Section 4, we present the details of our technique for adapting the unstructured LBM from the 2D grids to the surface meshes. In Section 5, we describe how to enhance our basic model with body forces, vorticity confinement (for unstructured grids), boundary conditions, and multi-component fluids. The implementation is presented in Section 6. Next, we show the results in Section 7 and have a discussion in Section 8. Finally, in Section 9 we conclude the paper and present ideas for future research.

2. Related Work

Physically-based flow models have been used in computer graphics and have realistically simulated the smoke, gas, fire, wind, and liquids. Those methods can be roughly categorized into the following two classes.

A prevalent class of method is to solve the Navier-Stokes, the finite difference equations, to obtain the velocity field that globally satisfies the fluid behavior. Foster and Metaxas [FM96, FM97] have employed the explicit finite difference solver to model liquids and turbulent gas. Stam [Sta99] has devised an unconditionally stable fluid solver using semi-Lagrangian advection schemes and implicit solvers. The large time-step can be used in this model to achieve fast simulation without sacrificing visual plausibility. Fedkiw et al. [FSJ01] have greatly improved the capturing of small-scale rolling characteristic of smoke by introducing the vorticity confinement to the visual simulation. Combining the fluid solver with the free surface tracking, Foster and Fedkiw [FF01] and Enright et al. [EMF02] have simulated the motion of liquids and Nguyen et al. [NFJ02] have simulated fire dynamics. Losasso et al. [LGF04] have extended the Navier-Stokes solver from the regular grids to the more efficient octree structures. Carlson et al. [CMT04] have pro-

posed a rigid-fluid method for the two-way coupling of the fluid with the rigid solid objects.

Another class of methods locally describes the statistical microscopic dynamics of the fluid particles or the particle distribution functions. These microscopic dynamics aggregate to yield the flow behavior. The examples are the particle-based method [MCG03, PTB*03] and LBM-based method [WZF*03, WZF*04, TR04]. The former is grid-less, while the latter defines the particle distribution functions of the given discrete velocities on a lattice grid. Although currently these methods may still be immature in terms of the stability or the achieved finest visual details compared to Navier-Stokes solvers mentioned above, they have already shown their own advantages. The local nature of operations decreases the complexity in computation and lends ease in handling microscopic interactivity due to the boundaries or the multiple fluids.

Unlike the above methods derived for 3D or 2D domains, the models that are most related to our work are those solving the flow dynamics for such special domains as over the curved 3D surfaces of arbitrary topologies. They are Stam's model [Sta03] and Shi and Yu's [SY04] model, having been mentioned in Section 1. Stam's model has extended his stable fluids solver [Sta99] for Navier-Stokes to the global parameterization space of the surfaces. He has further proposed a technique to alleviate distortions caused by the global parameterization. The distortions are still apparent but can disappear as the grid is refined. Shi and Yu's model avoids the global parameterization and directly applies the advection and the pressure solver on triangular surface meshes. However, a sparse linear system derived from the Poisson equation for pressure still needs to be solved globally. Like Shi and Yu's model, our approach directly simulates the flow on surface meshes avoiding global surface parameterization. Unlike both of above methods, our model is an extension of the microscopic method, the LBM. As a result, it is simple and intrinsically parallel. We also demonstrate in this paper its great capability of simulating the interesting motions of the flow with different kinds of boundary objects inside it and the mixture of multiple components of fluids.

3. Background

3.1. LBM

In the Boltzmann equation, the particle distribution function $f(\vec{r}, \vec{e}, t)$ is the probability of finding a particle at time t positioned at \vec{r} which has the velocity of \vec{e} . The LBM, a discretized version of the Boltzmann equation, discretizes the space to a regular lattice grid and the particle velocities to a set of velocity vectors $\{\vec{e}_i\}$. These velocity vectors should be symmetrical to satisfy the isotropic requirement of fluid properties. Figure 1 shows a 2-dimensional 9-velocity lattice model, called D2Q9. The 9 velocity vectors include the zero velocity vector, \vec{e}_0 , and 8 velocity vectors pointing to neighboring grid points, \vec{e}_1 through \vec{e}_8 . The lengths of \vec{e}_1 through

\vec{e}_4 are 1 while the lengths of \vec{e}_5 through \vec{e}_8 are $\sqrt{2}$. Associated with each \vec{e}_i , ($i = 0, 1, \dots, 8$) is a scalar value, the particle distribution function f_i . The macroscopic fluid density ρ and fluid velocity \vec{u} are given by

$$\rho(\vec{r}, t) = \sum_i f_i(\vec{r}, t), \quad \vec{u}(\vec{r}, t) = \frac{1}{\rho(\vec{r}, t)} \sum_i f_i(\vec{r}, t) \vec{e}_i. \quad (1)$$

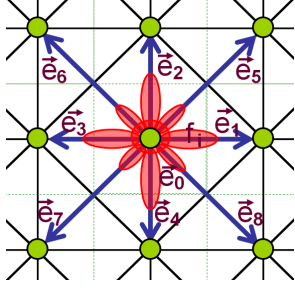


Figure 1: The D2Q9 lattice grid. Vector \vec{e}_0 stands for the zero velocity, while \vec{e}_1 through \vec{e}_8 are the velocity vectors pointing to neighboring points. Associated with each \vec{e}_i is a particle distribution function f_i (shown as a red ellipse).

Using the Bhatnagar, Gross, Krook (BGK) single-time-relaxation model [BGK54], the LBM explicitly evolves in a two-step process of collision and ballistic streaming,

$$\text{Collision} \quad f_i(\vec{r}, t^+) = f_i(\vec{r}, t) - \frac{1}{\tau} (f_i(\vec{r}, t) - f_i^{eq}(\vec{r}, t)), \quad (2)$$

$$\text{Streaming} \quad f_i(\vec{r} + \vec{e}_i, t + 1) = f_i(\vec{r}, t^+), \quad (3)$$

where the constant τ represents the relaxation time determined by the kinematic viscosity ν of the fluid ($\tau = 3\nu + \frac{1}{2}$), the notation $f_i(\vec{r}, t^+)$ denotes the post-collision particle distribution function, and f_i^{eq} represents the local equilibrium particle distribution function which is given by

$$f_i^{eq}(\rho, \vec{u}) = \rho(A + B(\vec{e}_i \cdot \vec{u}) + C(\vec{e}_i \cdot \vec{u})^2 + D\vec{u} \cdot \vec{u}). \quad (4)$$

Here, A through D are constant coefficients specific to the chosen lattice geometry. For further read of the LBM, interested readers are referred to [Suc01].

3.2. Unstructured LBM

An unstructured grid is an array of points with their connectivity relationship explicitly stated. Unlike the structured grid, its points have no particular logical order. This allows itself more geometrical flexibility. For example, in computer graphics, the triangular mesh has become a dominant method for the presentation of 3D surfaces. In modern CFD techniques, the unstructured grid is often used in finite-element or finite-volume computations. These computations have also appeared in visual simulations, such as the modeling of muscle dynamics [TBHF03].

The LBM on the unstructured grid was firstly proposed

by Peng et al. [PXDC98]. By importing finite-volume techniques within the LBM framework, their model applies to meshes without requiring any special kind of connectivity. Later, Ubertini et al. [UBS03, US04] have improved the 2D unstructured LBM and further incorporated it with a set of boundary conditions. Because our further development is based on Ubertini et al.'s 2D method, we briefly review their model below.

Figure 2 shows a 2D triangular LBM grid. Every grid point has nine velocity vectors, each of which is associated with a particle distribution function. Figure 3 shows a closer view to the geometrical 1-ring neighborhood around a grid point P . The neighboring points of P on the grid are denoted as P_k , $k = 1, 2, \dots, K$. The green regions in Figure 3 are a set of finite volumes defined around P . They are denoted as Ω_k , $k = 1, 2, \dots, K$. Each Ω_k is the union of two triangles $\Omega_k^- = [P, E_k, O_k]$ and $\Omega_k^+ = [P, O_k, E_{k+1}]$, where O_k is the center of the triangle $[P, P_k, P_{k+1}]$ and E_k, E_{k+1} are the midpoints of the edges PP_k, PP_{k+1} , respectively.

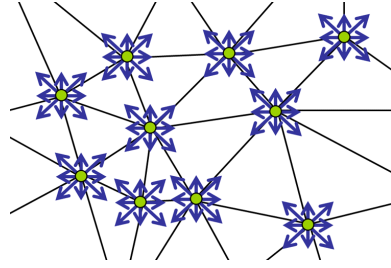


Figure 2: The 2D unstructured LBM grid. Every grid point has nine symmetrical velocity vectors (including the zero velocity), each associated with a particle distribution function.

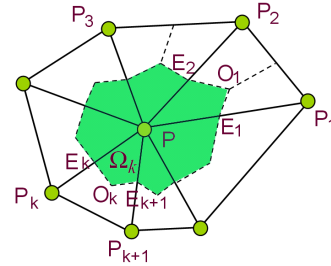


Figure 3: The geometrical layout of the 1-ring neighborhood around a grid point P . Points P_k are the neighboring points of P . The green regions stand for the finite-volumes which are defined around P .

For the unstructured grid, the Boltzmann equation is written as the following finite-difference equation:

$$f_i(P, t + dt) = f_i(P, t) + dt \sum_{k=1}^K (\Phi_{ik} - \Xi_{ik}), \quad (5)$$

where Φ_{ik} and Ξ_{ik} denote the streaming and collisional

fluxes of the i th particle distribution function f_i coming from the k th finite volume Ω_k . Applying the linear interpolation rules, the calculation of streaming fluxes is straightforward. The contribution of collisions arise from the integration of the linear interpolated value of the collision term $(f_i - f_i^{eq})/\tau$ over each finite volume Ω_k . The resulting finite-volume equation takes the following general form:

$$f_i(P, t + dt) = f_i(P, t) + dt \sum_{k=0}^K S_{ik} f_i(P_k, t) - \frac{dt}{\tau} \sum_{k=0}^K C_{ik} [f_i(P_k, t) - f_i^{eq}(P_k, t)], \quad (6)$$

where index $k = 0$ denotes the pivotal point P itself. The detailed expressions of the streaming and collision matrices S_{ik} and $C_{ik} = C_k \delta_{ik}$ ($\delta_{ik} = 1$, if $i = k$; and $\delta_{ik} = 0$ if $i \neq k$) are obtained by

$$S_{i0} = 0, \quad S_{ik} = \vec{e}_i \cdot \vec{N}_k / V_P, \quad k = 1, 2, \dots, K, \quad (7)$$

and

$$C_0 = 1/3, \quad C_k = \frac{V_{k-1} + V_k}{3V_P}, \quad k = 1, 2, \dots, K. \quad (8)$$

In the above, V_k is the area of Ω_k , while V_P is the area of $\Omega = \bigcup_k \Omega_k$. \vec{N}_k is defined as

$$\vec{N}_k = \left[\frac{5}{12} (\vec{A}_{k-1}^+ + \vec{A}_k^-) + \frac{2}{12} (\vec{A}_{k-1}^- + \vec{A}_k^+) \right], \quad k = 1, 2, \dots, K, \quad (9)$$

where \vec{A}_k^\mp are the vectors normal to the lines $E_k O_k$, $O_k E_{k+1}$, with magnitude equal to the length of these lines. Similarly, \vec{A}_{k-1}^\mp associate with lines $E_{k-1} O_{k-1}$ and $O_{k-1} E_k$, respectively.

4. Unstructured LBM on Curved Surfaces

In this section, we present the basic algorithm of our model. This model is devised by adapting Ubertini et al.'s unstructured LBM from 2D meshes to manifold surface meshes. To successfully realize this adaption, the following problems should to be solved: (1) for each mesh point, we need to define its nine velocity vectors; and (2) in order to apply Equation 6 in the computation, for each mesh point P , we need to locally flatten P 's 1-ring neighborhood.

To solve the first problem, for each mesh point, we firstly define the local frame $\langle \vec{s}, \vec{t} \rangle$, in its tangent space, then define the nine velocity vectors based on this local frame. Note that for most surface meshes (specifically close meshes whose genus is not one), it's impossible to define on them the local frames that are globally continuous. This means that there are unavoidable differences in the orientations of velocity vectors between neighboring mesh points. Fortunately, in Section 4.3 we introduce a technique to rotate and align velocity vectors and recompute the corresponding particle distribution functions. With this technique, there is no need for local frames to be globally continuous.

For the second problem, we flatten the 1-ring neighborhoods to the tangent planes in pre-processing. We use the *ghost points*, G_k , on behalf of the neighboring points P_k (see Figure 4). In the simulation, for each time-step we first update the states of all ghost points based on corresponding neighboring points, and then for each mesh point we execute the streaming and collision computation with information from ghost points, a procedure similar to that in 2D unstructured LBM. Section 4.1 through Section 4.3 give more details.

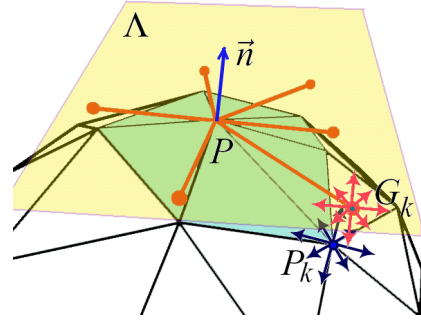


Figure 4: Flatten the 1-ring neighborhood of P to its tangent plane Λ . Ghost point G_k is on behalf of neighboring point P_k . The velocity vectors (dark blue) at P_k are transformed into vectors (pink) that lie in Λ . Such transformations cause no or only negligible distortions.

4.1. Define Velocity Vectors for Mesh Points

To define the nine velocity vectors for each mesh point P , we first define the local frame $\langle \vec{s}, \vec{t} \rangle$, where \vec{s} and \vec{t} are two orthogonal unit vectors defining the local tangent space at P . Any definition of local frames can be used for our model and we choose the following simple method. Assuming \vec{n} is the normal vector at mesh point P , we let

$$\vec{s}' = \begin{cases} \vec{n} \times \vec{x} & , \quad \text{if } |\vec{n} \cdot \vec{x}| \leq \frac{\sqrt{2}}{2} \\ \vec{n} \times \vec{y} & , \quad \text{otherwise,} \end{cases} \quad (10)$$

$\vec{s} = \vec{s}' / |\vec{s}'|$ and $\vec{t} = \vec{n} \times \vec{s}$. In the above, \vec{x} , \vec{y} , and \vec{z} are three mutually orthogonal unit vectors, “ \times ” denotes the vector cross product, and “ \cdot ” denotes the vector dot product. It can be seen that in above definition the local frame is uniquely determined by \vec{n} .

Similarly to the D2Q9 model described in Section 3, for each mesh point, we define its velocity vectors \vec{e}_i as follows in the 3D world space:

$$\vec{e}_i = \begin{cases} 0 & , \quad i = 0 \\ \cos(\theta_i) \vec{s} + \sin(\theta_i) \vec{t} & , \quad i = 1, \dots, 4 \\ \sqrt{2} \cos(\theta_i) \vec{s} + \sqrt{2} \sin(\theta_i) \vec{t} & , \quad i = 5, \dots, 9, \end{cases} \quad (11)$$

where

$$\theta_i = \begin{cases} (i-1)\pi/2 & , i=1, \dots, 4 \\ \pi/4 + (i-5)\pi/2 & , i=5, \dots, 9. \end{cases} \quad (12)$$

4.2. Flatten the 1-Ring Neighborhoods

In flattening (see Figure 4), the positions of the ghost points G_k are computed as follows. First, we find such neighboring point P_1 that the angle between the edge $\overline{PP_1}$ and tangent plane Λ is the smallest. Second, we project edge $\overline{PP_1}$ onto Λ and scale the length of the projected edge to $|\overline{PP_1}|$. The resulting edge is $\overline{PG_1}$ and hence the position of G_1 is determined. Third, similar to [LTD05], the positions of other ghost points are calculated in a way that all the edge lengths are exactly preserved and the angles between two consecutive edges are preserved up to a common factor. We denote the transformation matrix as M'_k which rotates $\overline{PP_k}$ to $\overline{PG_k}$ around point P with the vector $\overline{PP_k} \times \overline{PG_k}$ to be the rotation axis.

We then transform the velocity vectors at each neighboring point P_k into vectors that lie in P 's tangent plane. The resulting vectors are defined as G_k 's velocity vectors (see Figure 4). Such transformations should cause no or only negligible distortions. This is achieved by the matrix $M_k = M''_k M'_k$, where M''_k rotates the vector \vec{n}_k into \vec{n} around point P with the vector $\vec{n}_k \times \vec{n}$ to be the rotation axis, where \vec{n}_k is the normal vector at neighboring point P_k and $\vec{n}_k = M'_k \vec{n}_k$.

4.3. Rotate and Align the Velocity Vectors

The local LBM computation can not yet be directly applied as on the 2D unstructured grid, because the velocity vectors of the ghost points have different orientations from those of point P 's velocity vectors (see Figure 5).

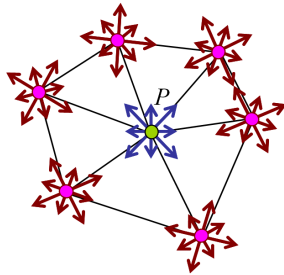


Figure 5: The velocity vectors of the ghost points have different orientations from those of the point P 's velocity vectors.

Our solution to this is to rotate the velocity vectors of the ghost points and align them with P 's velocity vectors. Accordingly, the particle distribution functions need to be recomputed in order to preserve the flow properties, such as the fluid density ρ and the fluid velocity \vec{u} (see Figure 6). For a given ghost point G_k , we denote the rotation angle as

Θ , which is in the range of $[0, 2\pi]$. We denote the original velocity vectors and their associated particle distributions as e'_i and f'_i respectively. The target velocity vectors and the corresponding new particle distribution functions are denoted as e_i and f_i respectively.

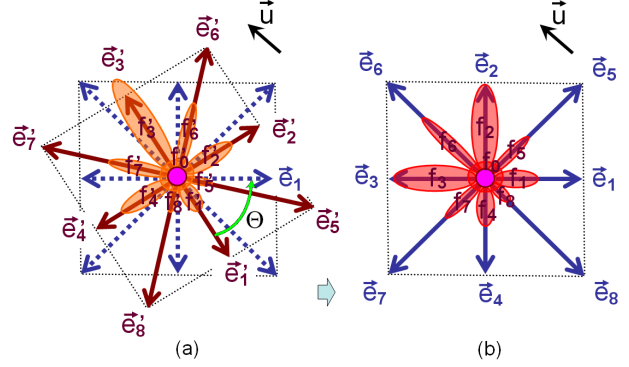


Figure 6: (a) We rotate the velocity vectors e'_i of a ghost point and align them with the velocity vectors e_i of point P . Θ is the rotation angle. The original particle distributions are denoted as f'_i . (b) We recompute the new particle distributions f_i for the rotated velocity vectors, which preserve the fluid density ρ and the fluid velocity \vec{u} .

The following equations preserve the fluid density and the fluid velocity:

$$\sum_i f_i = \rho = \sum_i f'_i, \quad (13)$$

$$\sum_i f_i \vec{e}_i = \vec{u} = \sum_i f'_i \vec{e}'_i, \quad (14)$$

They can be further supplemented with equations for preserving the energy, the stress tensor, and other flow properties. However, they are enough for our visual simulation and we have chosen the following way to satisfy them.

Without loss of generality, let's assume the rotation angle Θ is in the range of $[0, \pi/2]$, meaning that e_1 is between e'_1 and e'_2 . We let f_0 equal f'_0 . The values f_1, \dots, f_4 are computed based on f'_1, \dots, f'_4 . In this computation, we first let

$$\bar{f} = (f'_1 + f'_2 + f'_3 + f'_4)/4. \quad (15)$$

Then we subtract \bar{f} from each f'_i ($i = 1, \dots, 4$) and get

$$\lambda'_i = f'_i - \bar{f}, \quad i = 1, \dots, 4. \quad (16)$$

After that, we project the vectors $\lambda'_i e'_i$ and $\lambda'_{i+1} e'_{i+1}$ on the direction of e_i . The resulting vector length is

$$\lambda_i = \lambda'_i \cos(\Theta) + \lambda'_{i+1} \sin(\Theta), \quad i = 1, \dots, 4. \quad (17)$$

Finally, we add \bar{f} back and get

$$f_i = \bar{f} + \lambda_i, \quad i = 1, \dots, 4. \quad (18)$$

It is easy to prove that $\sum_{i=1}^4 f_i = \sum_{i=1}^4 f'_i = 4\bar{f}$ and

$\sum_{i=1}^4 f_i e_i = \sum_{i=1}^4 f'_i e'_i$. Similarly, we compute the values f_5, \dots, f_8 based on f'_5, \dots, f'_8 . Thus Equation 13 and 14 are satisfied.

5. Enhancements to Our Model

In this section, we introduce other elements that enhance our flow model over curved surfaces. They are boundary conditions, body forces, vorticity confinement on the unstructured grid, and multi-component fluids. Because these computations also only involve local operations and can be executed in flattened 1-ring neighborhoods, adapting them from previous 2D LBM models to our model is straightforward. From this part on, unless otherwise specially stated, all values and operations are presented in the tangent spaces $\langle \vec{s}, \vec{t} \rangle$ at mesh points.

5.1. Boundary Conditions

Ubertini et al. [UBS03] have introduced three ways to handle static and moving boundary conditions in the 2D unstructured LBM. They are, listed ascending in terms of implementation difficulty as well as physical accuracy, equilibrium method, mirror method, and covolume method. We have implemented the first method, which takes to set the particle distribution functions f_i as the equilibrium particle distribution functions f_i^{eq} for every boundary point. These f_i^{eq} are calculated using Equation 4, in which \vec{u} is set as the boundary velocity.

5.2. Body Forces

Body forces can be user applied force, gravity, vorticity confinement force, and etc. For each mesh point P , if the force vector is not in its tangent plane, we need to project it onto the tangent plane. We denote the resulting vector as \vec{F} . Then, this force affects the local particle distribution functions as follows, according to the previous work of LBM [BG00].

$$f_i \longleftarrow f_i + \frac{(2\tau - 1)}{2\tau} B \vec{F} \cdot e_i, \quad (19)$$

where B is the constant which has appeared in Equation 4.

5.3. Vorticity Confinement on Unstructured Grid

The vorticity confinement force adds small scale rolling features that are usually absent on coarse grid simulations. Fedkiw et al. [FSJ01] have firstly introduced to computer graphics the vorticity confinement method on the regular grid for the visual simulation of smoke.

Recently, a vorticity confinement formulation for the unstructured grid has been derived using dimensional analysis by Löhner et al. [LYR03]. We have incorporated this force into our model. The vorticity confinement force F_{vc} is expressed as a function of the local vorticity-based Reynolds-number $Re_{\omega,h}$, the local element size h , the vorticity ω , and

the gradient of the absolute value of the vorticity.

$$F_{vc} = g(Re_{\omega,h}) c_v \rho h^2 \nabla |\omega| \times \omega, \quad (20)$$

$$g(Re_{\omega,h}) = \max \left[0, \min \left[1, \frac{Re_{\omega,h} - Re_{\omega,h}^0}{Re_{\omega,h}^1 - Re_{\omega,h}^0} \right] \right], \quad (21)$$

$$Re_{\omega,h} = \frac{\rho |\omega| h^2}{\nu}, \quad (22)$$

$$\omega = \nabla \times \vec{u}, \quad (23)$$

where c_v is a constant regardless of the grid, and $Re_{\omega,h}^0$ and $Re_{\omega,h}^1$ are two parameters defining the effective range of $Re_{\omega,h}$. Note that calculations of Equation 20 and Equation 23 involve the finite difference operations ∇ and $\nabla \times$ respectively. The method to calculate these finite difference operations on the unstructured grid is introduced in the Appendix.

5.4. Multi-Component Fluids

The interaction of multiple-component fluids is a specially interesting and complex phenomenon, which has not been adequately addressed in computer graphics. In computational physics, there is a large literature for using the LBM to model this phenomenon, by taking the advantages of LBM in handling microscopic interactions. The fluids can be either immiscible or miscible. In our work, we focus on the immiscible two-component fluid, in which the interface between two fluids is always maintained. Adapting miscible fluids models into our model should be feasible as well.

Our method is based on the classic 2D LBM model for immiscible binary fluids [GRZZ91]. Red and blue particle distribution functions f_i^r and f_i^b are introduced to represent two different components of the fluid. From them, the density and velocity of the two fluids can be computed by Equation 1. The total (or the color-blind) particle distribution function is defined as $f_i = f_i^r + f_i^b$. The collision is applied on f_i as usual. After this, a special two-step two-component collision rule maintains the interfaces that separate the different components.

The first step is to add a perturbation to the particle distribution near the interface which creates the correct surface-tension dynamics. The interface is located by computing the local color gradient \vec{g} , which is perpendicular to the interface. The perturbation to each color-blind particle distribution is given by:

$$\hat{f}_i = f_i + A |\vec{g}| \cos 2(\theta_i - \theta_g), \quad (24)$$

where θ_i is the angle of velocity vector e_i and θ_g is the angle of the local color gradient \vec{g} . This operation redistributes mass near the interface, depletes mass along lattice links parallel to the interface and adds mass to the links perpendicular to the interface, while the total mass and momentum are conserved.

The second step is to recolor the mass after perturbation in

order to separate the two different components and maintain a clear interface. This is achieved by solving a maximization problem :

$$W(\hat{f}_i^r, \hat{f}_i^b) = \max \left[\left(\sum_i (\hat{f}_i^r - \hat{f}_i^b) \vec{e}_i \right) \cdot \vec{g} \right]. \quad (25)$$

To conserve the total red mass and blue mass, $\sum_i \hat{f}_i^r$ must equal to the total amount of red mass before collision. To conserve the mass in each lattice direction, $\hat{f}_i^r + \hat{f}_i^b = \hat{f}_i$ should be satisfied.

After the above collisions, the streaming is applied on \hat{f}_i^r and \hat{f}_i^b , and then the color-blind particle distribution is re-computed for the next time-step.

6. Implementation

6.1. Preprocessing

In preprocessing, we first scale up/down the mesh size, making the average edge length to be one. The reason for doing this is to make uniform parameters regardless of the original mesh dimension. Then the following values at every mesh point are calculated: the local frame, the positions of ghost points and their rotation angles Θ (mentioned in Section 4.3), and the coefficients S_{ik} and C_{ik} used in Equation 6.

6.2. Computational Procedure

The computational procedure of the flow simulation is listed as follows:

- 1: Initialize the values u , ρ , f_i^{eq} , and f_i for all mesh points
- 2: Update f_i for all ghost points
- 3: Compute \vec{u} , ρ and f_i^{eq} for all ghost points
- 4: **for** every mesh point **do**
- 5: Apply streaming and collision using Equation 6
- 6: Compute and apply body forces
- 7: **if** a boundary point **then**
- 8: Apply boundary condition
- 9: **end if**
- 10: Compute \vec{u} , ρ and f_i^{eq}
- 11: **end for**
- 12: Jump to line 2

This procedure is only for the single-component fluid. It can be just slightly modified as described in Section 5.4 to simulate the immiscible two-component fluid.

7. Results

We list here the results of several examples simulated with our model (please see the accompanying video which can also be downloaded at <http://www.cs.sunysb.edu/~vislab/projects/amorphous/animations/ULBMS/>).

Figure 7(a) and 7(b) show the flows over a dog surface and a two-hole torus surface respectively. We periodically deposit a material on the surfaces. Because the density of this material is larger than that of the fluid, gravity force drives the fluid to move, which causes the advection of the material. This advection is modeled by applying the semi-Lagrangian backtracing scheme with the flow velocity field in the flattened neighborhoods. Note that in some region, the fluid moves in the opposite direction of gravity due to incompressibility.

Figure 8 and Figure 9 show interactions of the flow over surfaces with boundary objects. In Figure 8, static boundaries over the dog model are the text shape *SCA 2005* and the star shape. Figure 9 shows animated boundaries, the white objects moving inside and activating the flow on the sphere. We visualize the velocity field using the existing image-based flow visualization method for curved surfaces [vW03].

Figure 10(a) and Figure 10(b) shows the immiscible two-component fluids over surfaces. For the simulation of Figure 10(a), at the beginning, the left part of the sphere are full of blue fluid while the right part full of pink fluid. The blue fluid is denser than the pink one. Gravity causes a turbulent flow motion, resulting very complex interfaces between two parts. For the simulation of Figure 10(b), the blue and pink fluids have the same density. Two parts continuously inosculate with each other, resulting in interesting dynamics.

Table 1: Frame rates of our simulations.

	Model	Vertices	Faces	Frames/Second
Figure 7(a)	Dog	37,502	75,000	1.7
Figure 7(b)	Two-Hole Torus	49,998	100,000	1.3
Figure 8	Dog	37,502	75,000	1.7
Figure 9	Sphere	4,098	8,192	10.0
Figure 10(a)	Sphere	4,098	8,192	7.8
Figure 10(b)	Skull	3,752	7,500	8.1

We report the performance of our simulations in Table 1. All the frame rates are measured on a Intel Xeon 2.40GHz processor. The per-step simulation time is nearly linear to the number of surface vertices for the single-component fluid. The preprocessing time for each of the examples, unlisted in the table, is in the range of 0.5 to 3.0 seconds.

8. Discussions

In previous sections, we have shown that our microscopic based computations simplify the modeling of fluids on curved surfaces. All operations are local and explicitly parallel. This allows the model the potential to be accelerated on model SIMD processors, such as graphics cards (GPUs). The model also lends flexibility in handling interaction of the fluid with static/moving boundary objects and interaction of multiple component fluids. Furthermore, it is worth to point out that the unstructured LBM and our extended version are

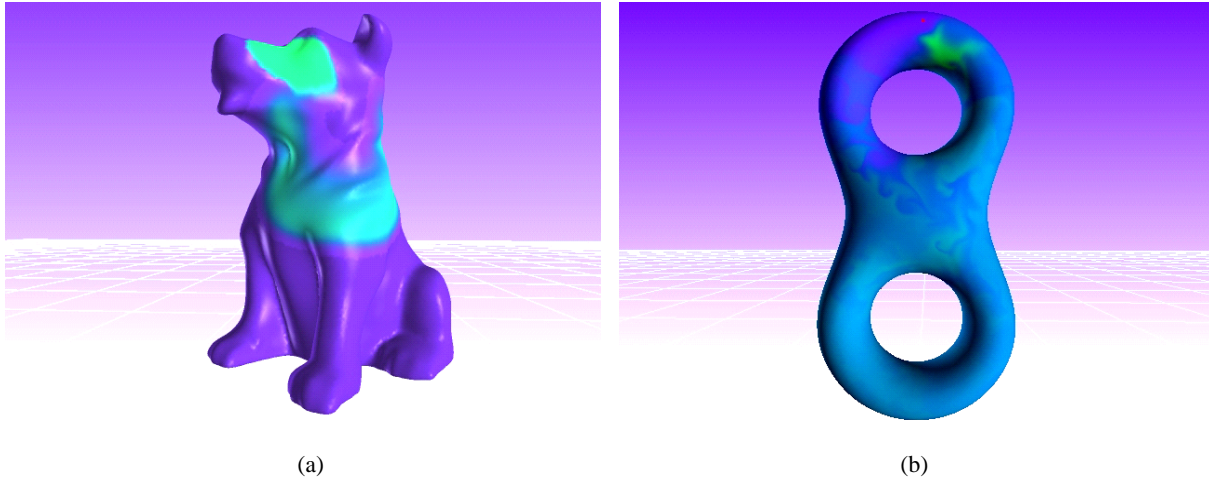


Figure 7: Flow motion due to gravity on (a) the dog model and (b) the two-hole torus model.

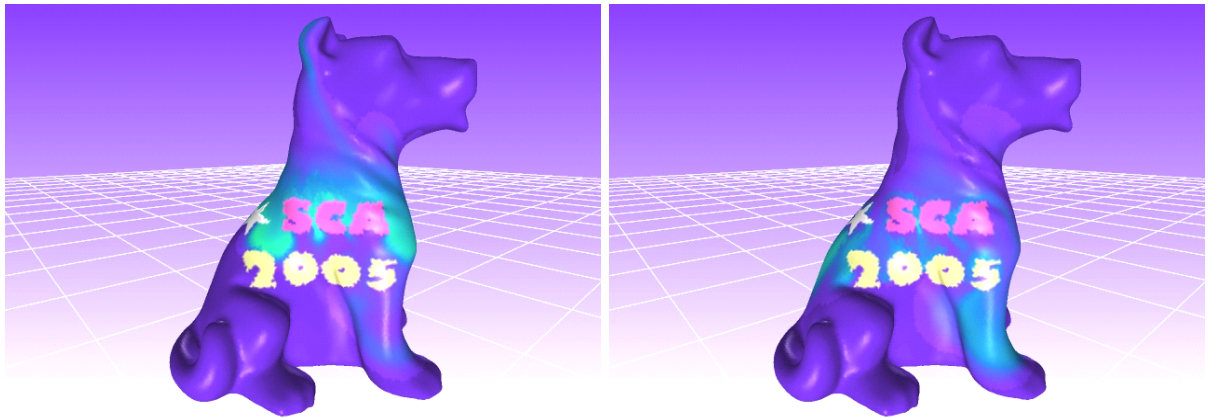


Figure 8: Flow motion due to gravity on the dog surface, with static boundaries.

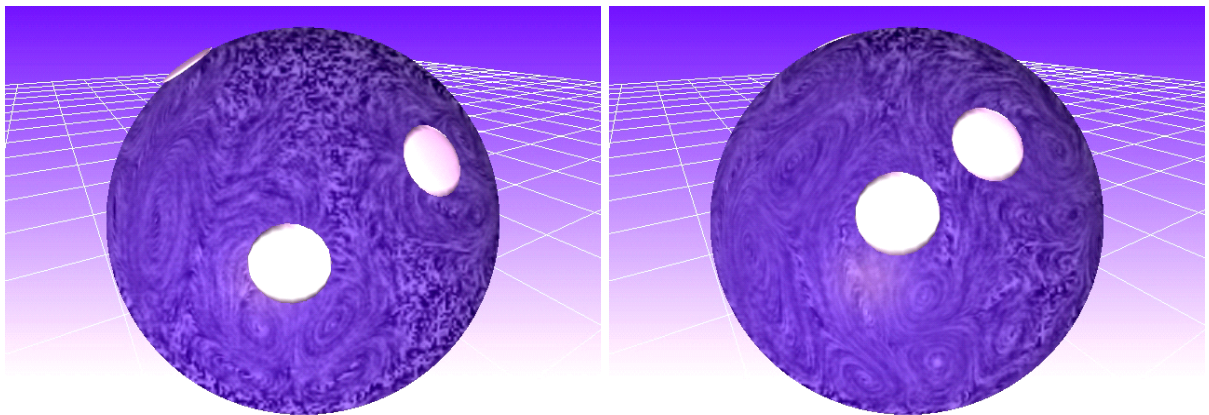


Figure 9: Flow motion caused by the animated boundary objects on the sphere.

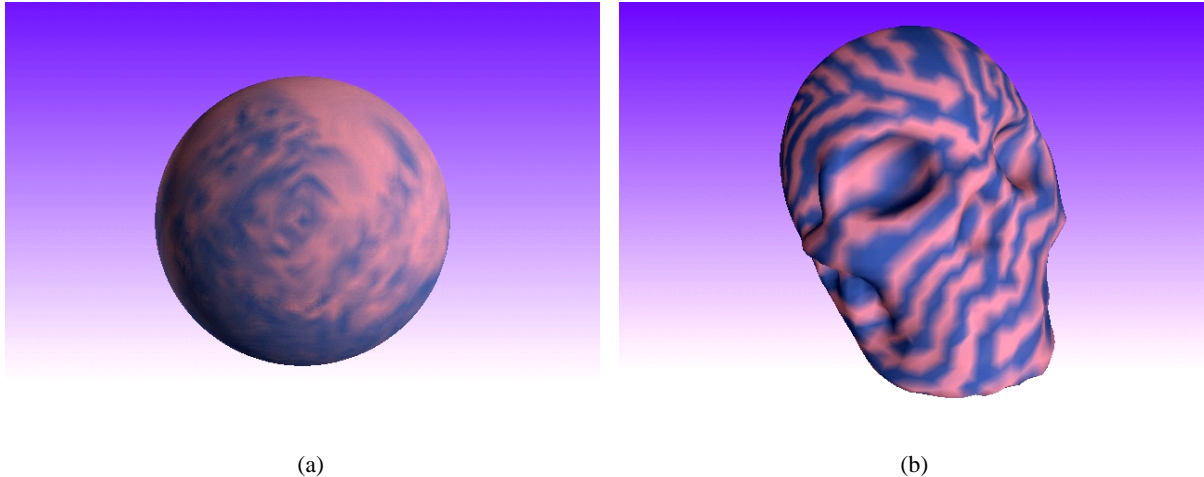


Figure 10: Immiscible two-component fluids, colored in blue and pink. (a) a turbulent mixture of two components on the sphere. (b) a peaceful inosculation on the skull.

not limited to triangular meshes. Using the finite volume technique described in Section 3.2, they can actually be derived for meshes with arbitrary connectivity [PXDC98].

As an explicit method, our model has the limitations in stability. The unstable states could be reached when the particle distribution functions f_i computed by Equation 6 became minus values. This could be caused by three factors: large time-step dt , long and narrow triangles or sharp features on the surface mesh. For dt , we have found that 0.25 is stable for all of our simulations. Long and narrow triangles cause large value of the coefficients S_{ik} , which may result in minus values f_i . Fortunately, we can address this problem by preprocessing the mesh models using the existing edge flip algorithm [dBvKOS00] to replace these triangles with triangles whose smallest internal angles are bigger. If the mesh has sharp features, we may need to smoothen them, for example, using subdivision methods.

9. Conclusions and Future Work

The contributions of this paper are: (1) We have extended the 2D unstructured LBM model to 3D curved surface meshes of arbitrary topology. (2) Compared with previous models solving the complex flow motion on curved surfaces, our microscopic-based model has advantages in simplicity and explicit parallelism of its computations. It can be easily extended to meshes with arbitrary connectivity as well. (3) We have demonstrated that the features such as static/animated boundary conditions, body forces, vorticity confinement for unstructured grid, and multi-component fluids can be easily incorporated in our model, making it a powerful solution.

In the future work, we plan to accelerate our model on the GPUs, taking advantages of their SIMD data processing capability to achieve real-time performance. Another direction

for further research is to design the unstructured LBM over curved surface based on the multiple-relaxation-time LBM scheme [DGK*02], which is more stable in computation.

Appendix: Finite Difference Operators on Unstructured Grid

Equation 20 and 23 involve the calculation of finite difference operators ∇ and $\nabla \times$ on the unstructured grid. In our case, these operators are executed in the flattened 1-ring neighborhoods on the tangent planes. This is done as follows. Four *virtual* points are defined with local coordinates $(-b, 0)$, $(b, 0)$, $(0, -b)$ and $(0, b)$ in the tangent space (we set b as $0.3h$). The flow properties at each virtual points are obtained by using bilinear interpolation in the appropriate triangle. Finally, the finite difference operator is executed upon these flow properties, similar to on a regular grid. Note that for all virtual points, the computation for determining which triangles they reside in and the corresponding bilinear interpolation coefficients can be done in the preprocessing time.

Acknowledgements

This work has been supported by an NSF grant CCR-0306438. Datasets are courtesy Headus Inc. and New York University. Special thanks to Feng Qiu, Haitao Zhang, Xianfeng Gu, and Hong Qin for insightful discussions on geometry issues.

References

- [BG00] BUICK J. M., GREATED C. A.: Gravity in a lattice Boltzmann model. *Physical Review E* 61, 5 (2000), 5307–5320.

- [BGK54] BHATNAGAR P. L., GROSS E. P., KROOK M.: A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. *Physical Review* 94, 3 (1954), 511–525.
- [CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *Proceedings of SIGGRAPH* (2004), 377–384.
- [dBvKOS00] DE BERG M., VAN KREVELD M., OVERMARS M., SCHWARZKOPF O.: *Computational Geometry: algorithms and applications*. Springer, 2000.
- [DGK*02] D’HUMIÉRES D., GINZBURG I., KRAFCHYK M., LALLEMAND P., LUO L.: Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions of Royal Society of London* 360, 1792 (2002), 437–451.
- [EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *Proceedings of SIGGRAPH* (2002), 736–744.
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. *Proceedings of SIGGRAPH* (2001), 15–22.
- [FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing* 58, 5 (1996), 471–483.
- [FM97] FOSTER N., METAXAS D.: Modeling the motion of hot, turbulent gas. *Proceedings of SIGGRAPH* (1997), 181–188.
- [FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. *Proceedings of SIGGRAPH* (2001), 15–22.
- [GRZZ91] GUNSTENSEN A., ROTHMAN D., ZALESKI S., ZANETTI G.: Lattice Boltzmann model of immiscible fluids. *Physical Review A* 43, 8 (1991), 4320–4327.
- [LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *Proceedings of SIGGRAPH* (2004), 457–462.
- [LTD05] LEE H., TONG Y., DESBRUN M.: Geodesics-based one-to-one parameterization of 3D triangle meshes. *IEEE Multimedia* 12, 1 (2005), 27–33.
- [LYR03] LÖHNER R., YANG C., ROGER R.: Tracking vortices over large distances using vorticity confinement. *Twenty-Fourth Symposium on Naval Hydrodynamics* (2003).
- [MCG03] MUELLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. *ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation* (2003), 154–159.
- [NFJ02] NGUYEN D., FEDKIW R., JENSEN H.: Physically based modeling and animation of fire. *Proceedings of SIGGRAPH* (2002), 721–728.
- [PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFJOHN A., WHITAKER R.: Particle-based simulation of fluids. *Eurographics* (2003), 401–410.
- [PXDC98] PENG G., XI H., DUNCAN G., CHOU S. H.: Lattice Boltzmann method on irregular meshes. *Physical Review E* 58, 4 (1998), 4124–4127.
- [Sta99] STAM J.: Stable fluids. *Proceedings of SIGGRAPH* (1999), 121–128.
- [Sta03] STAM J.: Flows on surfaces of arbitrary topology. *Proceedings of SIGGRAPH* (2003), 724–731.
- [Suc01] SUCCI S.: *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2001.
- [SY04] SHI L., YU Y.: Inviscid and incompressible fluid simulation on triangle meshes. *Journal of Computer Animation and Virtual Worlds* 15, 3-4 (2004), 173–181.
- [TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R. P.: Finite volume methods for the simulation of skeletal muscle. *Eurographics/SIGGRAPH Symposium on Computer Animation* (2003), 68–74.
- [TR04] THUREY N., RUDE U.: Free surface lattice-Boltzmann fluid simulations with and without level sets. *Workshop on Vision, Modeling, and Visualization (VMV Stanford)* (2004), 199–208.
- [UBS03] UBERTINI S., BELLA G., SUCCI S.: Lattice Boltzmann method on unstructured grids: Further developments. *Physical Review E* 68, 016701 (2003).
- [US04] UBERTINI S., SUCCI S.: Recent advances of lattice Boltzmann techniques on unstructured grids. *in press Progress in Computational Fluid Dynamics* (2004).
- [vW03] VAN WIJK J. J.: Image based flow visualization for curved surfaces. *Proceedings of IEEE Visualization* (2003), 123–130.
- [WZF*03] WEI X., ZHAO Y., FAN Z., LI W., YOAKUM-STOVER S., KAUFMAN A.: Blowing in the wind. *ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation* (2003), 75–85.
- [WZF*04] WEI X., ZHAO Y., FAN Z., LI W., QIU F., YOAKUM-STOVER S., KAUFMAN A.: Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics* 10, 6 (November 2004), 719–729.