Free Surface Lattice-Boltzmann fluid simulations with and without level sets

Nils Thürey, Ulrich Rüde

University of Erlangen-Nuremberg
System Simulation Group
Cauerstr. 6, 91054 Erlangen, Germany
Email: Nils.Thuerey@cs.fau.de

Abstract

We present two variants of free surface Lattice-Boltzmann fluid simulations for the animation of liquids in computer graphics. The Lattice-Boltzmann method is an attractive alternative to conventional fluid solvers, due to its simplicity and flexibility, especially for changing geometries and topologies. While our first method directly calculates the mass fluxes between the cells of the computational grid, another variant of the method is explained, that uses level sets to track the fluid surface. This has advantages for the smoothness of the fluid surface and improves the representation of details in the free surface, but makes the conservation of mass more difficult. Several examples will be shown to highlight the differences between the two methods.

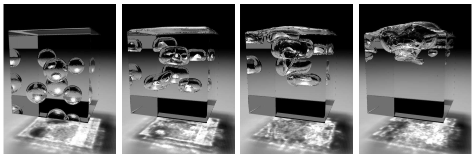

Figure 1: A simulation of six bubbles rising in a container with fluid.

1 Introduction

The simulation of fluids with free surfaces is important for a variety of technical applications like foaming or casting processes, and moreover, the correct animation of fluids is a challenge in the field of photo realistic visualization of physical phenomena for computer graphics. In both cases the correct treatment of the interface between gas and fluid determines the accuracy of both the results and the plausibility of the visual appearance. The motion of a fluid is governed by the *Navier-Stokes equations* (NSE) which describe the evolution of the velocity and pressure of the fluid. In contrast to conventional computational fluid dynamics solvers, which discretize the macroscopic differential equations, the method presented here will use the *Lattice-Boltzmann method* (LBM). It represents the fluid as a cellular automaton to solve the underlying microscopic transport equations. Level sets will be used to track the movement of the free surface, along which special boundary conditions will be applied to calculate the appropriate values for velocity and pressure at the interface.

Section 2 will present related work in the area of fluid animation and LBM simulations, the basic Lattice-Boltzmann algorithm will be described in more detail in Section 3, and an overview of level set methods for fluid simulations will be given in Section 4. Section 5 will describe the free surface LBM model, followed by a geometric curvature calculation algorithm, and the combined LBM level set free surface algorithm. First results of the level set algorithm will be presented in the results section, together with some comparisons of LBM simulations with and without level sets.

2 Related work

Several algorithms for the simulation of multi-phase systems with the LBM exist [3]. Drawbacks of these methods are, e.g. instabilities for greatly differing densities of the phases or smeared out regions for interface tracking [5]. The first restriction is a severe limitation for computer graphics applications, as the two phases are usually a gas and liquid. Smeared out interface representations, on the other hand, result in higher resolution requirements for the simulation, and thus require more computations than a simulation that could resolve the same fluid interface on a smaller grid. The LBM itself, however, is an interesting alternative to Navier-Stokes solvers, mainly due to its simplicity and the capability to handle complex geometrical and topological boundary conditions [8]. The free surface method described here, which is also used in e.g. [7], does not have these drawbacks, and can simulate a single fluid phase without the necessity of additionally computing the motion and pressure of the gas phase.

The relevance of fluid simulations has been shown in a variety of publications. Chen et. al [2] were among the first to use computational fluid dynamics to compute motions of fluids for computer graphics. In [4] methods for successfully dealing with complex fluid surfaces were shown. These algorithms commonly use discretizations of the Navier-Stokes equations, with differing forms of time stepping and stabilizations [14]. Although the LBM has become an established method in fluid dynamics research, it is up to now not commonly used for animations in computer graphics. The LBM has been used for effects like wind [18] and was implemented in graphics hardware [19], but in both cases without simulating a free surface flow.

3 The Lattice-Boltzmann method

The LBM usually works on an equidistant grid of cells, each of which stores a discrete number of velocities, the particle *distribution functions* (DFs). For two-dimensional simulations nine directions are commonly used (D2Q9 model), while for three dimensions, the D3Q19 model with nineteen velocities is the most common. The directions of the velocity vectors are shown in Figure 2. The following examples and equations will assume a D3Q19 model, but are directly transferable to the two dimensional case. Each DF represents a number of particles in the fluid that moves along the direction of its velocity vector. Thus, for each time step for a cell at position $\mathbf{x}$ a DF $f_i(\mathbf{x}, t)$ has to be stored as a floating-point value for each velocity direction $(i = 1..19)$, with the direction vectors $\mathbf{e}_i$ defined as

$$(0,0,0) \quad \text{for} \quad i=0$$

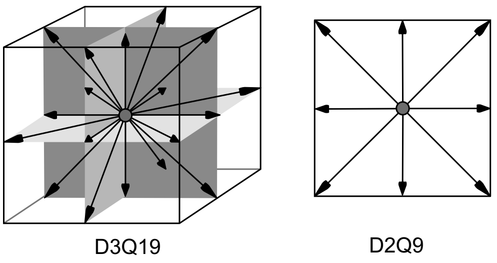

Figure 2: The velocities of the D3Q19 and D2Q9 LBM models.

$$\begin{aligned}
(\pm1,0,0) &\quad \text{for} \quad i=1,2\\
(0,\pm1,0) &\quad \text{for} \quad i=3,4\\
(0,0,\pm1) &\quad \text{for} \quad i=5,6\\
(\pm1,\pm1,0) &\quad \text{for} \quad i=7..10\\
(0,\pm1,\pm1) &\quad \text{for} \quad i=11..14\\
(\pm1,0,\pm1) &\quad \text{for} \quad i=15..18
\end{aligned}$$

Note that the first DF f_0 represents the particles in the cell which are at rest. From these values, the macroscopic values for density ρ and velocity $\mathbf{v}$ can be easily computed with the moments of f.

$$\rho = \sum_{i=1}^{19} f_i \ , \quad \mathbf{v} = \frac{1}{\rho}\sum_{i=1}^{19} f_i \mathbf{e}_i \qquad (1)$$

The algorithm proceeds in two steps – the *stream* and the *collide* steps. During streaming, the particles are moved with their corresponding velocities. As the time step is normalized to a length of 1, the particles directly move to the neighboring cell along their velocity direction, as shown in Figure 3. After each streaming step, all cells have a complete set of particle distribution functions again. The collide step subsequently accounts for the collisions between the particles that occur in a real fluid during the movement. This is done by relaxing the incoming DFs from streaming with a set of equilibrium distribution functions f^eq_i that can be calculated for each cell with its density and velocity from Eq. (1). The equilibrium distribution functions are defined as:

$$f_i^{eq} = w_i\rho\left[1 + \frac{3}{2}\mathbf{u}^2 + 3\mathbf{e_i}\cdot\mathbf{u} + \frac{9}{2}(\mathbf{e_i}\mathbf{u})^2\right] \quad (2)$$

where w_i are weights that depend on the length of the velocity vector:

$$w_i = \frac{1}{3} \quad \text{for} \quad i=0$$

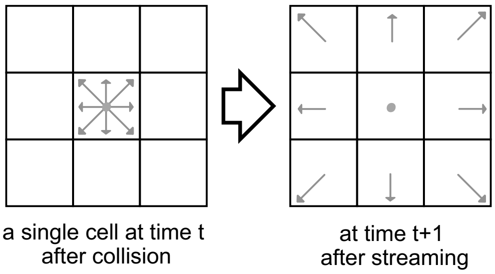

Figure 3: The distribution functions of a single cell at time before and after the streaming step.

$$w_i = \frac{1}{18} \quad \text{for} \quad i = 1..6$$
$$w_i = \frac{1}{36} \quad \text{for} \quad i = 7..18$$

The value of the distribution function for the next time step f_i' is calculated by a weight ω that is set according to the viscosity of the fluid.

$$f_i' = (1-\omega) f_i + \omega f_i^{eq} \tag{3}$$
$$\text{with } \omega = \frac{2}{6\nu + 1} \tag{4}$$

Here ν is the viscosity of the fluid in lattice units. For stability reasons, the value of omega has to be in the range of $0..2$. The combined equation for the stream and collide step is:

$$f_i(\mathbf{x} + \mathbf{e_i} \triangle t, t + \triangle t) - f_i(\mathbf{x}, t) = -\triangle t\, \omega \left(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t) \right) \tag{5}$$

and is known as the BGK equation due to its approximation of the particle collisions [1] with a single parameter. Additionally, external forces like gravity can be applied by accelerating the fluid in each cell before the calculation of the equilibrium distribution functions. No-slip boundary conditions for the LBM are implemented by reflecting the DFs at the boundary. Hence for each cell, instead of copying the neighboring DF from a boundary cell during streaming, its own opposing DF is taken. The results in a normal and tangential velocity of zero along the boundary.

4 Level sets

Level sets are a well-known method for front tracking – they have been applied to a variety of problems from computational vision to physical simulations [12]. For the class of level set algorithms the surface is represented as the level set of a continuous higher dimensional function $\phi(\mathbf{x})$. Usually the zero level set is used to represent the surface, which means $\phi(\mathbf{x}) = 0$ for all points that lie on the surface. To simplify the calculations, it is useful to define ϕ as being a signed distance function. This means for the surface Γ of a volume Ω:

$$\begin{aligned} \phi(\mathbf{x}) &= 0 \quad \forall \mathbf{x} \in \Gamma \\ \phi(\mathbf{x}) &= -d(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega \\ \phi(\mathbf{x}) &= d(\mathbf{x}) \quad \forall \mathbf{x} \notin \Omega\,, \quad \text{where} \\ d(\mathbf{x}) &= \min(\mid \mathbf{x} - \mathbf{x}' \mid) \quad \forall \mathbf{x}' \in \Gamma \end{aligned}$$

ϕ is then advected with the velocities $\mathbf{v}(\mathbf{x})$ that must be known where ϕ is defined, and in our implementation are calculated by the LBM fluid solver. The evolution of ϕ is given by the solution of the advection/convection equation:

$$\phi_t + \mathbf{v} \cdot \nabla \phi = 0 \tag{6}$$

where ϕ_t is the temporal derivative of ϕ and ∇ denotes the gradient operator. Thus, instead of a Lagrangian front representation, which would advect a certain number of particles on the surface in the velocity field, level set methods use an Eulerian approach, and advect a continuous function implicitly representing the surface. This has a number of advantages – the surface will not break up due to an insufficient number of particles (a common problem for Lagrangian surface tracking), and the normal and curvature can be easily calculated. For applications like fluid simulations, ϕ can be stored as an additional value in each cell, whose value is the distance from the cell center to the nearest point on the surface, see Figure 7. The easiest way to solve Eq. (6) is to use finite differences and upwinding. Depending upon the direction of the velocity, each spatial derivative of the gradient $\nabla\phi$ is calculated with one of its adjacent neighbors. Considering the spatial derivative in one direction, e.g. the x derivative at $\phi_i = \phi(x_i, y, z)$ can be calculated as:

$$\phi_x = \begin{cases} \frac{\phi_i - \phi_{i-1}}{\triangle x} & , \quad \mathbf{v}_i > 0 \\ \frac{\phi_{i+1} - \phi_i}{\triangle x} & , \quad \mathbf{v}_i < 0 \end{cases} \tag{7}$$

where $\triangle x$ is the size of a cell in x direction. Using this approximation of the components of the gradient, the advection of ϕ can be calculated with the following formula:

$$\phi(\mathbf{x}, t + \triangle t) = \phi(\mathbf{x}, t) - (\nabla\phi \cdot \mathbf{v}) \triangle t \tag{8}$$

For better accuracy, higher order methods are usually desirable. Directly extending this approach, a more accurate derivative of ϕ can be found by approximating ϕ as a polynomial of higher order, and using the derivative of this polynomial. Polynomials of third order are commonly used, the coefficients for these can be computed from divided differences of correspondingly higher order. To further increase the accuracy Osher et al. use weighting factors to compute the divided differences of third order using the values of ϕ where it is smooth [10]. With this method, the order of the level set advection can be increased to five. The accuracy of time stepping can also be improved: instead of performing a simple Euler step, other commonly used time stepping methods can be applied – for example modified Euler (midpoint rule) and higher order Runge-Kutta methods [13]. For the stability of the method it is usually required, that the velocities are smaller in each dimension than the resolution of the grid (CFL condition). Since the velocities from the LBM solver are used, this causes no problems, because for the stability of LBM the velocities of the fluid have to be significantly smaller than the grid size. This also means that higher order time stepping algorithms do not substantially improve the front tracking.

An inherent problem of the level set methods described so far, when using them for tracking the interface of a fluid, is the loss of mass. It can be shown with simple test cases, that under-resolved regions of the level set can vanish during the advection. E.g. thin layers of fluid, or small drops can incorrectly be removed from the simulation this way. To alleviate this problem Enright et. al propose a hybrid level set method in [4], that additionally advects two types of particles on both sides of the level set. These particles are used to reconstruct the level set in under-resolved regions, and significantly improves mass conservation this way.

Another class of algorithms that is closely linked to the level set methods are *fast marching methods*. They can be used to efficiently compute the evolution of a surface along its normal direction and represent an explicit form of upwinding. By computing the time the surface would need to reach a certain cell, fast marching methods can be used to compute signed distance functions or extrapolate velocities. The latter case is commonly needed for fluid simulations, as these compute the velocities inside the fluid, the level set method, however, requires values for the velocity on both sides of the surface to maintain the validity and smoothness of the signed distance function around the surface. Hence, the algorithm that will be explained in Section 7, also uses the fast marching method to extrapolate the fluid velocities into the gas region in each time step.

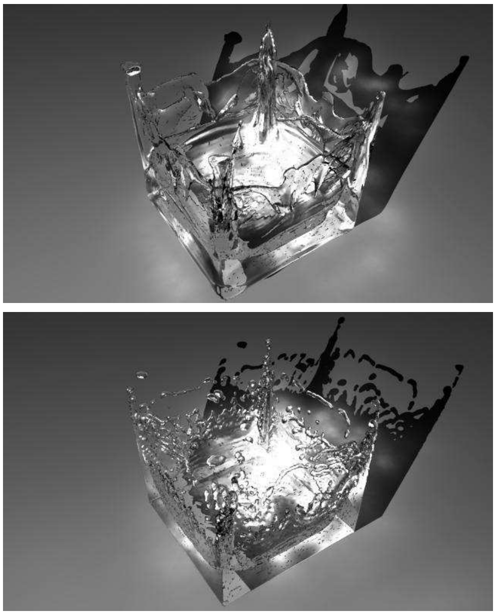
Figure 4: A square drop of fluid is falling down into a container. The upper picture is taken from a simulation using level sets, while the simulation of the lower picture uses the mass tracking algorithm.

5 Free surface tracking

The free surface algorithm was first demonstrated for a two-dimensional problem in [6] and is based on tracking the mass of fluid throughout the computational grid, coupled with boundary conditions for the free surface. With small extensions it can be applied to 3D problems.

The gas and fluid phase of a free surface simulation require a distinction between cells that contain no fluid (only gas), interface cells which are partially filled, and cells completely filled with fluid. An exemplary configuration is shown in Fig. 5. The latter type of cells can be treated as described in Section 3, and no computations are necessary for

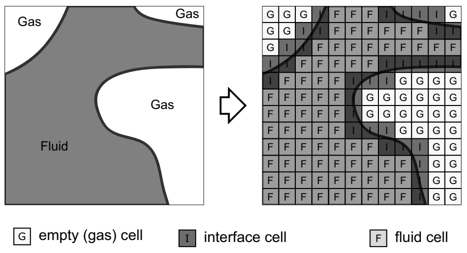

Figure 5: The LBM free surface algorithm distinguishes between fluid, interface and empty cells.

empty cells. Only interface cells need to track the amount of fluid they contain. As this amount usually changes during time, the mass exchange with all neighboring fluid and interface cells is computed by subtracting the outgoing DFs from the incoming ones during the stream step:

$$m(\mathbf{x}, t + \triangle t) = m(\mathbf{x}, t) + \sum_{l=1}^{19} \frac{\epsilon(\mathbf{x}_{e_i}, t) + \epsilon(\mathbf{x}, t)}{2} (f_I(\mathbf{x}_{e_i}, t) - f_i(\mathbf{x}, t)) \quad (9)$$

where $\mathbf{x}_{e_i} = \mathbf{x} + \mathbf{e_i}$, and $m(\mathbf{x}, t)$ denotes the mass of the cell at position $\mathbf{x}$ and time t. $\epsilon(\mathbf{x}, t)$ is the fraction of the cell that is filled with fluid. As empty cells are not included in the computation, all DFs which would be streamed from empty cells need to be reconstructed for the interface cells. This can be achieved by recalculating the equilibrium DFs, using the velocity of the interface cell (the gas is assumed to have the same velocity as the fluid close to the interface) and the gas density.

$$f_I(\mathbf{x}, t + 1) = (f_i^{eq}(\mathbf{u}, \rho) + f_I^{eq}(\mathbf{u}, \rho)) \cdot \sigma - f_i(\mathbf{x}, t) \quad (10)$$

Here σ is a factor to control the curvature force and gas pressure, and f_I is the distribution function opposite to f_i. $\mathbf{u}$ and ρ are the velocity and density of the interface cell. For the atmosphere, the gas density is simply the default density of the LBM simulation. For the treatment of bubbles, the gas density is tracked by storing an initial mass for a bubble, and calculating its volume for each time step. A high gas density ($\sigma > 1$) results in a pressure force at the fluid interface, pushing the fluid away, while a lower pressure ($\sigma < 1$) applies a force towards the gas phase. This reconstruction step ensures that all DFs are known for interface cells. Therefore, these cells can be treated in the usual way during collision as described above. The algorithm can deal with bubbles in the fluid by storing an id with each empty cell that identifies the bubble it belongs to. Coalescence of bubbles can be detected when an interface cell has empty neighbors with differing bubble id's. The two bubbles then can be combined by adding their masses and volumes, and reassigning the corresponding empty cell bubble id's. The atmosphere can also be treated as a bubble with constant pressure. Our current implementation does not recognize when new bubbles form due to gas regions that get enclosed by fluid in the course of the simulation. However, this could be done by adding an additional segmentation pass after the cell reinitialization.

Once interface cells become completely filled or empty – indicated by a mass of zero or equal to the density, respectively – their type has to be changed accordingly. This can also cause neighboring fluid or empty cells to be changed to interface cells, since the layer of interface cells has to be closed. Neighboring empty and fluid cells would result in a loss of mass, as a DF copied to the empty cell would not be included in the simulation anymore. Thus, a closed layer of interface cells is necessary to track the interface motion correctly, for details we refer to [15].

6 Geometric curvature calculation

During the reconstruction with Eq. (10), the surface tension can also be included by modifying the pressure coefficient σ according to the interface curvature. One possibility to compute the curvature of the fluid surface is to use the geometry generated with the marching cubes algorithm [9]. This algorithm is usually used to visualize isosurfaces from scalar fields, and generates a closed triangulated surface for a given isolevel. For the free surface LBM simulation, the mass values from each cell can be used to calculate the isosurface, using an isolevel of 0.5. The points generated for the triangulation can be used to determine the curvature along two coordinate axis planes. The two planes are determined according to the approximated surface normal from the scalar field. For each dimension, a circle through three chosen points is constructed. The two radii are then averaged to compute the mean

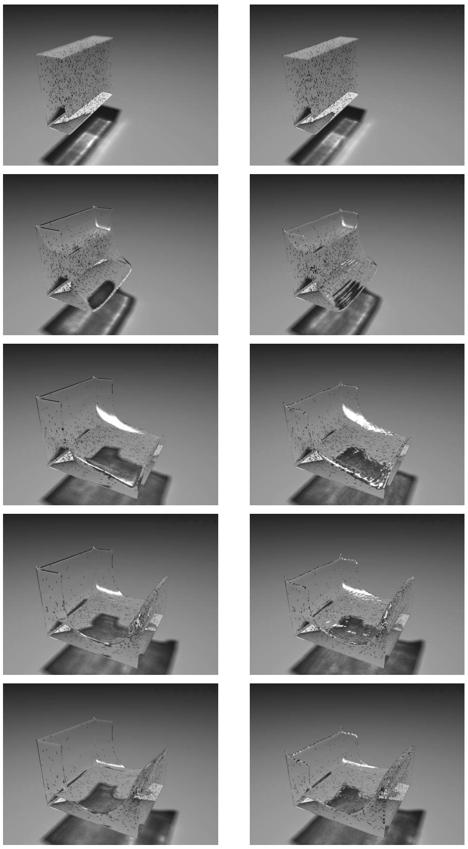

Figure 6: A comparison of two breaking dam simulations – the left one uses level sets, the right one the mass tracking algorithm.

curvature, and modify σ according to the surface tension of the fluid to be simulated. Depending on the side of the interface, the sign of the curvature may need to be inverted.

While this works well for large curvatures, especially for smooth fluid regions, errors in the computed curvature can occur, and lead to flickering of the fluid surface together with the generation of artificial velocities. In [16], these problems were reduced by using the triangulated surface itself for the computation of the curvature, instead of a selection of single points. Nevertheless, this also significantly increases the computational overhead, and the accuracy of the method varies depending on the underlying triangulation.

7 LBM free surfaces with with level sets

Due to these problems, an LBM method using level sets to track the fluid surface for the LBM simulation was implemented. As described in Section 4, the fluid surface given by the amount of fluid in each LBM cell can also be described using the zero level set of a signed distance function. From the initial fluid configuration, the level set can be initialized with two passes of the fast marching method. From the interface cells of the simulation, one pass constructs the distance values outside, the other one inside of the fluid. From then on, the values of the signed distance function can be used to determine whether a certain cell has to be treated as fluid, interface or empty cell. This algorithm, as well as the one described in Section 5, requires the reconstruction of incoming DFs at interface cells as free surface boundary condition. The calculation of the mass exchange with Eq. (9), however, is not necessary. The treatment of cell changes, the most complicated part, is also simplified, as this information is now given by the position of the level set. After performing stream and collide steps of the LBM, the level set is advected in each time step. Unfortunately, the LBM does not compute velocity values outside of the fluid region, hence, these have to be extrapolated using another fast marching pass during each time step.

The distance of extrapolation can be reduced by using a narrow band level set method, as described in [11]. In this case, the level set is only propagated in a region within a certain range around the surface, hence the velocities only have to be extrapolated within this region. This of course also reduces the number of level set advection steps, that are necessary for each time step. Still, for our current implementation, the velocity extrapolation is the computationally most expensive part.

8 Results

Our implementation includes a raytracing module for the visualization of the fluid simulations, which was also used to create the pictures shown here.

Figure 1 shows a simulation with six bubbles rising in a square container. This simulation was done without the use of level sets, and demonstrates the capabilities of the free surface LBM algorithm. As

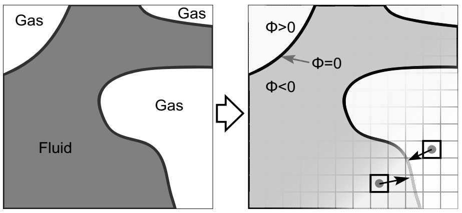

Figure 7: Discretization of the signed distance function for an exemplary surface - each cell stores the (positive or negative) distance from its center to the nearest point on the surface.

can be seen, the rising motion of the bubbles, and their deformation due to the downward streaming of the fluid around the surface, are properly captured. The algorithm furthermore correctly deals with the coalescence of bubbles.

For free surface simulations, the standard test setup is the breaking dam problem. In a brick-shaped domain an amount of resting fluid is initialized, and the simulation starts when an imaginative wall separating it from the rest of the domain is removed. The gravity then accelerates the fluid, and results in a swashing motion. For this problem, a simulation comparing the two LBM variants can be seen in Figure 6.

For the pictures to the left, the mass tracking algorithm was used, while for the right column of pictures the algorithm with level sets was used. For the latter case the fluid surface is considerably smoother than the other one, which exhibits steps at the moving front, and is not able to completely resolve the thin fluid regions at the walls. The smooth surface also results in improvements when calculating the caustics, as steps in the surface result in unwanted focussing of the light on the floor plane.

Another comparison of the two methods can be found in Figure 8. Here a spherical drop is falling down to the bottom of rectangular container. Here drawbacks of our current level set implementation can be seen, as the mass is not completely conserved by the simulation using level sets, which currently does not yet use the hybrid level set method described above. Figure 4 shows another setup that demonstrates the complexity of the structures that can develop from fluid simulations. Here the fluid is especially turbulent, resulting in strong splashes

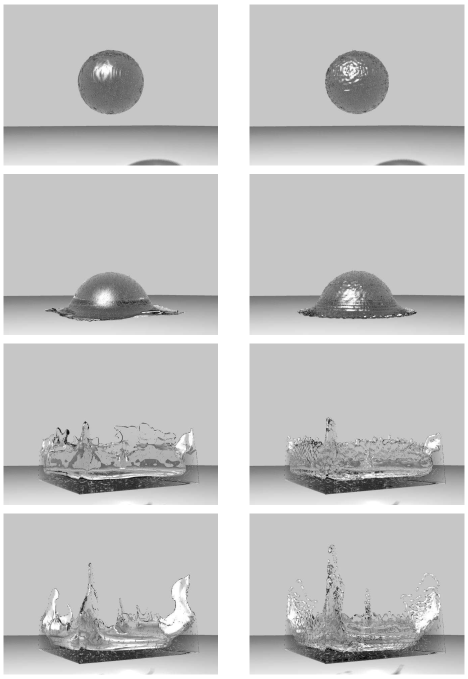

Figure 8: A drop of falling fluid. Again to the left with, to the right without level sets.

at the four corners of the domain boundary. Animations for these test cases can also be downloaded from [17].

9 Conclusion

We have presented two variants of free surface Lattice-Boltzmann fluid simulations. The LBM free surface model can correctly reproduce the behavior of a fluid gas interface, and can incorporate bubble coalescence and surface tension. The mass tracking method ties directly into the LBM, and despite its simplicity strictly conserves mass. Problems arise when the curvature of smooth fluid regions is calculated, or thin layers of fluid are moving through the grid. A new variant of this method uses level sets to track the fluid surface, thus increasing its ability to resolve thin fluid layers, and simplifying the calculation of surface normals and curvature. However, additional complexity is required to extrapolate the fluid velocities into the gas region, and to guarantee mass conservation.

We will continue to evaluate the level set method for LBM – especially the conservation of mass has to be improved, e.g. by the hybrid particle level set method. While the curvature calculation with level sets is inexpensive compared to the geometric approach, its accuracy needs to be tested.

10 Acknowledgements

This research is funded by the DFG Graduate College GRK-244 *3-D Image Analysis and Synthesis*. Furthermore, we want to thank Thomas Pohl and Carolin Körner, Michael Thies and Torsten Hofmann from *WTM Erlangen*.

References

[1] P. Bhatnagar, E. Gross and M. Krook, "*A model for collision process in gases I: small amplitude processes in charged and neutral one-component systems*", Phys. Rev. E, Vol. 50, pp. 511-525, 1954

[2] J. Chen and N. da Vitoria Lobo, "*Towards interactive-rate simulations of fluids with moving obstacles using Navier-Stokes equations.*", Graphical Models and Image Processing 57, pp. 107-116

[3] S. Chen and G. Doolen, "*Lattice Boltzmann Method for Fluid Flows*", Annual Review of Fluid Mechanics, 30, pp. 329-364, 1998

[4] D. Enright, R. Fedkiw, J. Ferziger and I. Mitchell, "*A Hybrid Particle Level Set Method for Improved Interface Capturing*", Journal of Comp. Phys. 183, pp. 83-116, 2002

[5] I. Ginzburg and K. Steiner, "*A free-surface lattice Boltzmann method for modelling the filling of expanding cavities by Bingham fluids*", Phil. Trans. R. Soc. Lond. A, 360, pp. 453-466, 2002

[6] C. Körner and R.F. Singer, "Numerical Simulation of Foam Formation and Evolution with Modified Cellular Automata", Metal Foams and Porous Metal Structures, MIT Publishing, pp.91-96, 1999.

[7] Carolin Körner and Michael Thies, "*Foaming of Light Metals, Mechanisms and Numerical Modeling*", to appear in Proceedings of SIMTEC 2004.

[8] M. Krafczyk, P. Lehmann, O. Philippova, D. Hänel and U. Lantermann, "*Lattice Boltzmann Simulations of complex Multi-Phase Flows*", Springer Verlag, pp. 50-57, 2000

[9] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm", Computer Graphics Vol. 21, No. 4, pp. 163-169, 1987.

[10] S. Osher and R. Fedkiw, "*Level Set Methods and Dynamic Implicit Surfaces*", Springer Verlag, 2003

[11] D. Adalsteinsson and J.A. Sethian, "A Fast Level Set Method for Propagating Interfaces", Academic Press, Inc., 1995

[12] J.A. Sethian, "Level Set Methods and Fast Marching Methods", Cambridge University Press, 1996

[13] C. W. Shu and S. Osher, "*Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes*", Journal of Comp. Phys. 83, pp. 32-78, 1988

[14] J. Stam, "Stable fluids", in Proceedings of SIGGRAPH 99, Annual Conference Series, pp. 111-120

[15] Nils Thürey, "A single-phase free-surface Lattice-Boltzmann Method", *Masters thesis*,IMMD10 University of Erlangen-Nuremberg, 2003.

[16] Nils Thüerey, Thomas Pohl, Ulrich Rüde, Markus Öchser, Torsten Hofmann and Carolin Körner, "*LBM Simulation and Visualization of Free Surface Flows in 3D*", to appear in Proceedings of ICMMES 2004

[17] Nils Thürey, "*Fluid Simulation with LBM*", www.ntoken.com/fluid, 2004.

[18] Leonid Velikovich, Amitabh Varshney "*Adapting the Lattice-Boltzmann Model for Efficient Airflow Modeling inside the View Frustum*", Report of University of Maryland, Computer Science Department, 2003.

[19] Wei Li, Zhe Fan, Xiaoming Wei, and Arie Kaufman "*GPU-Based Flow Simulation with Complex Boundaries*", Technical Report 031105, Computer Science Department, SUNY at Stony Brook, 2003

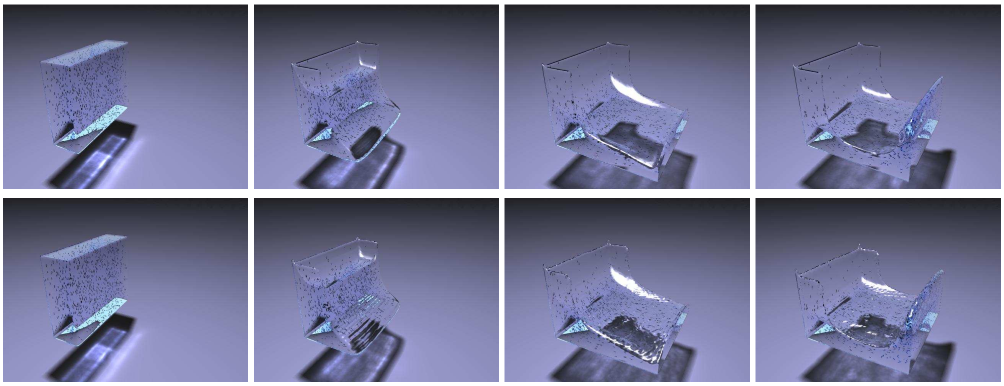

Figure 9: A simulation of the breaking dam problem using the LBM free surface algorithm. The pictures of the top row were generated using level sets for front tracking, while the lower row uses the mass tracking method.

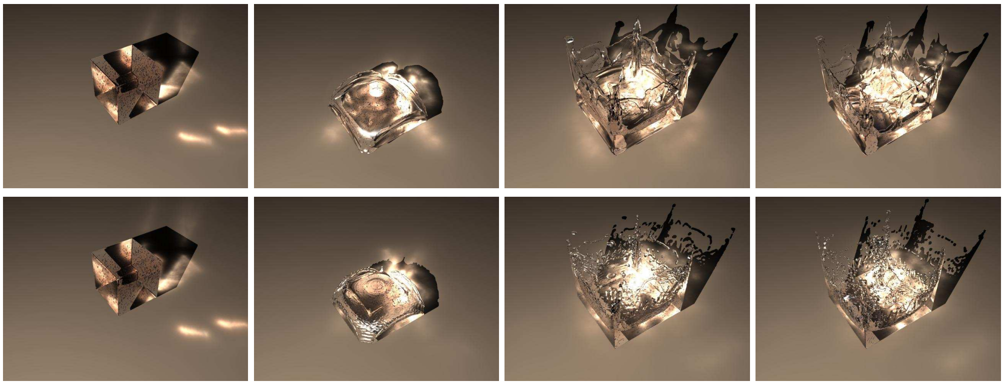

Figure 10: A square drop of fluid falls in a rectangular container. Again the upper row uses level sets, while the lower one was calculated without them.

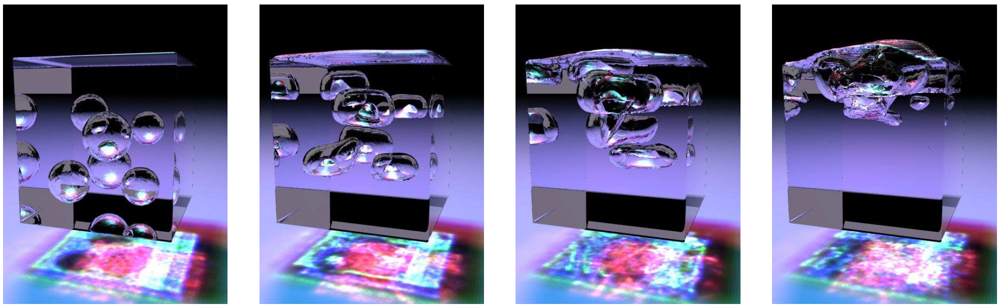

Figure 11: Pictures of a simulation of six rising bubbles that were simulated with the mass tracking method.