Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Mass-Conserving Eulerian Liquid Simulation

Nuttapong Chentanez[1,2] and Matthias Müller[1]

[1]NVIDIA Research
[2]Chulalongkorn University

Abstract

We present a GPU friendly, Eulerian, free surface fluid simulation method that conserves mass locally and globally without the use of Lagrangian components. Local mass conservation prevents small scale details of the free surface from disappearing, a problem that plagues many previous approaches, while global mass conservation ensures that the total volume of the liquid does not decrease over time. Our method handles moving solid boundaries as well as cells that are partially filled with solids. Due to its stability, it allows the use of large time steps which makes it suitable for both off-line and real-time applications. We achieve this by using density based surface tracking with a novel, unconditionally stable, conservative advection scheme and a novel interface sharpening method. While our approach conserves mass, volume loss is still possible but only temporarily. With constant mass, local volume loss causes a local increase of the density used for surface tracking which we detect and correct over time. We also propose a density post-processing method to reveal sub-grid details of the liquid surface. We show the effectiveness of the proposed method in several practical examples all running either at interactive rates or in real-time.

1. Introduction

Tracking the free surface of a liquid is an important and challenging problem. For an overview of existing methods we recommend the class notes of Wojtan et al. [WMFB11]. The most popular approach is to advect a scalar field with the fluid and define the liquid surface to be one of the iso-surfaces. The main advantage of this class of methods is that they handle topological changes implicitly in contrast to mesh-based tracking methods. Until recently, the level set method was the method of choice in graphics. Here, the signed distance field is used as the scalar field with the zero-iso-surface as the liquid surface.

A well known drawback of the level set method is that volume is lost both globally and locally. With global volume loss the water level decreases over time while local volume loss causes small detail such as thin sheets and droplets to disappear. A way to alleviate this problem is to introduce Lagrangian components such as particles [FF01], [EMF02] or triangle meshes [BGOS05]. Even though these methods reduce volume loss, they cannot guarantee complete volume conservation. Moreover, Lagrangian components add significant run-time cost and complicate the implementation significantly, especially for GPUs.

As an alternative to the signed distance field, [MMTD07] introduced the idea of using a density field as the scalar field for surface tracking with the liquid surface being the 0.5 iso-surface. This density field is not to be mistaken for the density field of the liquid. In incompressible fluid simulations, the fluid-density is 1 everywhere and therefore not stored. So in what follows, we use the symbol ρ for the *surface density*.

We chose to use surface density instead of the signed distance field because there are advection methods that strictly conserve quantities like density such as the one proposed by [LAF11]. Their advection method is unconditionally stable and fully conservative.

With this approach, the overall mass defined by the surface density is conserved. Since the surface density can deviate from 1 temporarily, the overall volume may vary over time though. However, in contrast to the level set method where such variations go unnoticed, volume deviations are reflected in the density field. In this paper we propose several methods to preserve volume both globally and locally using the information stored in the density field.

Ideally, the surface density has the form of a step function at the liquid-air interface. Over time, however, the initially sharp boundary blurs out due to numerical diffusion. Therefore, [MMTD07] apply a sharpening filter at each time step which conserves mass globally but not locally. We propose a

DOI: 10.2312/SCA/SCA12/245-254

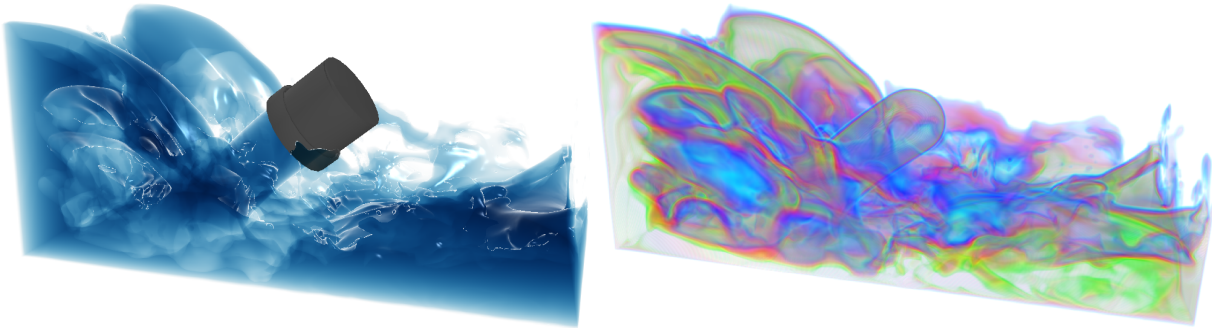

Figure 1: A liquid jet with large flow rate inside a rectangular tank simulated at a resolution of 256x128x128 cells. The simulation time step is $1/30$ seconds (CFL 25) which is much larger than what is typically used in other grid based liquid simulation approach. The liquid moves across many grid cells in a single time step, a situation our method handles without difficulty. Left) Surface rendering. Right) Volumetric rendering showing intricate detail of the density field.

new sharpening method which conserves mass both locally and globally.

Our main contributions are:

- A GPU friendly, purely Eulerian liquid simulator that conserves mass locally and globally without any need for Lagrangian components.
- A new GPU friendly sharpening method which conserves mass locally and globally.
- Modifications to reduce the computational cost of the conservative advection method of [LAF11] and to make it more GPU friendly.
- Additional novel steps to handle non-axis aligned and moving solid boundaries.
- A density post processing technique to bring out sub-grid detail.

2. Related Work

3D Eulerian liquid simulation was introduced to computer graphics by Foster and Metaxas [FM96] who used a finite difference approach to solve the governing equations. Later Foster and Fedkiw [FF01] employed the semi-Lagrangian method introduced by Stam [Sta99] to solve the advection term and the level set method and particles to track liquid surface. Enright et al. [EMF02] devised the Particle Level Set (PLS) method which uses particles on both sides of liquid-air interface to reduce volume loss. Since then, many methods have been proposed to further improve the accuracy of Eulerian surface tracking.

Various approaches have been proposed to track the liquid domain more faithfully. [BGOS05] used a triangle mesh representation in connection with a level set grid, [HK10] augmented the level set grid with quadrature points. Grid-less methods work with Lagrangian elements only such as particles [ZB05], [APKG07] or [YT10], triangles meshes [MÖ9], [BB09] and [WTGT10].

In this paper we focus on fluid mass and volume conservation. A popular way to compensate volume gain or loss is to modify the divergence of the velocity field as proposed in [FOA03]. This technique was extended and used for conserving volume of bubbles [KLL*07], highly deformable objects [ISF07] and liquids [MMTD07].

The problem of loss of liquid mass and momentum has also been addressed by proposing elaborate advection methods such as BFECC [KLLR05], modified MacCormack [SFK*08], derivatives advection [KSK08] and conservative semi-Lagrangian advection [LGF11], [LAF11].

As an alternative to level-set, the fluid domain can be tracked with a Volume-of-Fluid (VOF) approach [HN81] where the volume fraction of fluid in each cell is evolved over time. With proper care, VOF can be made mass conserving. However, despite several improvements in subsequent works such [PP04], [AGDJ08], reconstructing surface normal and curvature from VOF is still difficult. Sussman and Puckett [SP00] proposes coupled Level Set and Volume-of-Fluid (CLVOF) which track the fluid interface with both representations, where VOF is used for re-initializing the Level Set. The surface can then be extracted from the Level Set. CLVOF is extended to handle multiple interfaces in [KPyNS10]. The downside of CLVOF is the need to use two representations which can be quite computationally intensive. An alternative to VOF is to track a smeared-out surface density and keep it relatively sharp with a sharpening operation. This method was introduced to computer graphics by Mullen et al. [MMTD07]. Our fluid domain tracking approach builds upon this work and make it conserve mass both locally and globally.

Apart from the Eulerian formulation we use, there are many alternative models to simulate 3D liquids such as the Lattice-Boltzmann method [TR04] and [TR09], approaches based on the discrete sin-cosine transform [LR09] or particle based methods such as [MCG03], [PTB*03], [APKG07], [SP09], and [SG11].

3. Methods

We simulate the liquid by solving the inviscid Euler Equations,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{\mathbf{f}}{d} - \frac{\nabla p}{d}, \quad (1)$$

subject to the incompressibility constraint

$$\nabla\cdot\mathbf{u} = 0, \quad (2)$$

where $\mathbf{u} = [u,v,w]^{\mathrm{T}}$ is the fluid velocity field, p is the pressure, t is time, d the fluid density and $\mathbf{f}$ is a field of external forces.

The equations are solved in the domain specified by a *surface density field* ρ [MMTD07], in the region where $\rho > 0.5$. The surface density itself is advected with the fluid via

$$\frac{\partial \rho}{\partial t} = -\mathbf{u}\cdot\nabla\rho \quad (3)$$

and periodically sharpened to prevent the 0.5 iso-contour from being blurred by numerical damping. The interaction of the liquid with the environment is handled by considering the appropriate Dirichlet and Neumann boundary conditions.

3.1. Discretization

We discretize the simulation domain using a regular staggered grid [HW65]. The x, y and z components of fluid velocity $\mathbf{u} = (u,v,w)$ are stored at the center of the faces perpendicular to the x, y and z axis, respectively. The scalar pressure p and the density ρ are stored at cell centers following [MMTD07].

3.2. Time integration

Our time integration scheme is summarized in Algorithm 1. The overall structure is the same as the one proposed in [MMTD07] with our novel modifications to the advection, sharpening and pressure incompressibility enforcement steps.

Algorithm 1 Time step

1: Velocity extrapolation
2: Density advection and density sharpening
3: Velocity advection and external force addition
4: Incompressibility enforcement

First, we extrapolate the velocity field into the air region. Then, we advect the surface density field and sharpen it. After this we advect the velocity field and take external forces into account. Finally, we enforce incompressibility by making the velocity field divergence free.

3.3. Velocity Extrapolation

To extrapolate the velocities from inside the liquid into the surrounding air we use the scheme described in [CM11b], i.e. we apply the method of [JRW07] to derive the velocities a few grid cells away from the interface and then extrapolate based on a hierarchy of grids to obtain velocities far away from the surface.

3.4. Density Advection

We advect ρ using our unconditionally stable conservative advection method which we derived from the method proposed by Lentine et al. in [LGF11] and [LAF11] and improved in terms of computational cost.

Lentine et al. [LGF11] modified the semi-Lagrangian advection scheme to conserve mass by ensuring that each cell distributes all its mass of the current time step among some cells at the next time step. Let A be the matrix of the discretized advection operator such that $\rho^{n+1} = A\rho^n$, where ρ^n and ρ^{n+1} are the density in the current and the next time step respectively. Let w_{ij}^- (w_{ij}^+) represent the fraction of value that cell i gives to cell j which is found by backward (forward) tracing and computing the tri-linear interpolation weight. The entries of A in the standard semi-Lagrangian advection is hence $A_{ij} = w_{ji}^-$. Then, $\beta_j = \sum_i A_{ij}$ is the fraction of mass from cell j that gets advected. To ensure that mass is conserved, A needs to be modified such that all the β_j are 1.

Lentine et al. [LGF11] achieve this by first iterating through all cells j with $\beta_j > 1$ and re-scaling all A_{ij} to A_{ij}/β_j. In a second step they iterate through all cells j with $\beta_j < 1$ and forward trace the velocity field to adding the weights $(1-\beta_j)$ by distributing them among the A_{kj}, where k are the cells reached by forward tracing and tri-linear interpolation. At this point, all the β_j are 1, i.e. A is mass conserving. This method works well for compressible flow on fine grids. However, as discussed in [LAF11], the scheme produces artifacts when used for incompressible flow on coarser grids.

The problem is due to the clamping of the β_j by re-scaling which limits the amount of density that reaches certain cells. An indicator of this amount are the $\gamma_i = \sum_j A_{ij}$. The traditional semi-Lagrangian method ensures that all the γ_i are 1 while the β_j are arbitrary. In contrast, the scheme described above ensures that all the β_j are 1, while the γ_i are arbitrary.

Lentine et al. [LAF11] propose a method to ensure the β_j are all 1 while the γ_i stay close to 1. To this end they keep track of the cumulative γ_i over time as separate variables. The matrix A is computed by performing multiple forward and backward traces as follows:

1. Advect γ_i using the backward semi-Lagrangian method (set to 1 in the first time step).
2. Compute A by performing a backward tracing step as before, i.e. $A_{ij} = w_{ji}^-$.
3. Scale A by the γ_i, i.e. $A_{ij} \leftarrow A_{ij}/\gamma_i$.

4. Compute the β_j from A.
5. Forward trace the velocity field to add the weights $(1-\beta_j)$ to A for all cells j where $\beta_j < 1$ by distributing them among the A_{kj}, where k are the cells reached by forward tracing and tri-linear interpolation as before, i.e. $A_{kj} \mathrel{+}= (1-\beta_j)w^+_{jk}$.
6. Compute the new γ_i from the updated matrix A.
7. Scale A by the γ_i, i.e. $A_{ij} \leftarrow A_{ij}/\gamma_i$.
8. Re-compute the β_j from the updated matrix A.
9. Clamp the β_j to 1 by re-scaling $A_{ij} \leftarrow A_{ij}/\beta_j$.
10. Re-compute the γ_i from the updated A.
11. Evaluate $\rho^{n+1} = A\rho^n$.

At this point, all the β_j are 1 but the γ_i might still deviate from 1. To bring them even closer to 1 Lentine et al. apply a diffusion step on ρ^{n+1} and the γ_i. They iterate through all the cells dimension-by-dimension. If, for two neighboring cells i and j, $\gamma_j > \gamma_i$, they move $\rho_j(\gamma_j - \gamma_i)/2\gamma_j$ from cell j to cell i and set both γ_j and γ_i to $\frac{\gamma_j+\gamma_i}{2}$. If $\gamma_j < \gamma_i$, the flow happens in the opposite direction. This process is repeated 1 to 7 times per time step. Note that these diffusion iterations do not affect the β_j, so they remain 1.

Implementing the method described above on a GPU would require 5 scatter passes per iteration in steps 4, 6, 8, 10, and 11. Scattering is expensive on GPU's because it either requires atomic operations or a prefix-scan.

We propose a modification of this method. The basic idea is to reorder the forward tracing and the re-scaling steps to simplify the calculations. The resulting discrete conservative advection operator is not the same as the one computed with the original scheme. However, both are just approximations to the doubly-stochastic matrix (all row- and column sums are one) closest to the original discrete advection operator.

While the visual results are of similar quality as shown in Figure 2 and the accompanying video, our simplification reduces the number of scatter passes from 5 to 3. Another advantage of our new scheme is that A does not need to be stored explicitly because the order of the operations allow for updating ρ^{n+1}, β and γ directly. Not storing A explicitly is possible in the original scheme as well but it would complicate the process considerably and would require even more scatter operations.

Here is our modified scheme:

1. Advect γ_i using the semi-Lagrangian method (set to 1 in the first time step).
2. Initialize $\beta \leftarrow 0$.
3. Add the weights γ_i to β by distributing them among the β_l, where l are the cells reached by backward tracing and tri-linear interpolation, i.e. $\beta_l \mathrel{+}= w^-_{li}\gamma_i$.

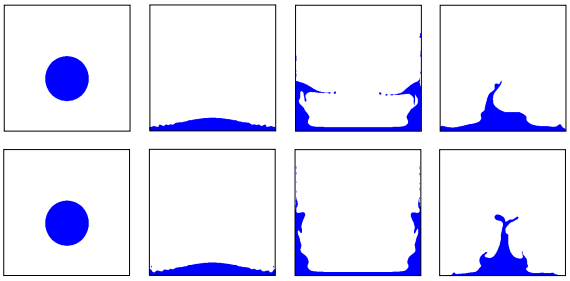

Figure 2: Snapshots from a simulation of a 2D ball of liquid dropping into an empty box at a resolution of 128^2 cells. Top) Using conservative advection method of Lentine et al. 2011. Bottom) Using our conservative advection method. The result are of similar visual quality.

4. Evaluate ρ^{n+1} from ρ^n and γ' from γ by backward tracing and tri-linear interpolation from cells l but this time scale the weights by $\frac{\gamma_i}{\max(1,\beta_l)}$, i.e. $\rho^{n+1}_i \mathrel{+}= \sum_l \frac{\gamma_i}{\max(1,\beta_l)} w^-_{li}\rho^n_l)$
5. $\gamma \leftarrow \gamma'$. (This can be done in-place during the previous step).
6. For each cell j whose $\beta_j < 1$, add $\rho^n_j(1-\beta_j)$ to ρ^{n+1} by distributing the value among the ρ^{n+1}_k, where k are the cells reached by the forward tracing and tri-linear interpolation, i.e. $\rho^{n+1}_k \mathrel{+}= \rho^n_j(1-\beta_j)w^+_{jk}$.
7. Similarly, for each cell j whose $\beta_j < 1$, add $(1-\beta_j)$ to γ by distributing the value among the γ_k, where k are the cells reached by the forward tracing and tri-linear interpolation, i.e. $\gamma^{n+1}_k \mathrel{+}= \gamma^n_j(1-\beta_j)w^+_{jk}$. These two steps can be done concurrently.
8. Apply diffusion as in the original approach.

This modified method only requires 3 scatter passes in the steps 3, 6, and 7. As demonstrated in Table 1, our method keeps γ in a similar range to that of [LAF11], while [LGF11] has a much larger range, resulting in visible compressibility artifacts.

Method	Minimum γ	Maximum γ
Our Method	0.627	2.403
[LAF11]	0.627	2.502
[LGF11]	0.271	9.793

Table 1: Minimum and maximum γ of our method, LAF11 and LGF11 for the situation of Figure 2. Our method and LAF11 have similar range, while LGF11 has a much larger range which explains the incompressibility artifacts.

3.5. Density sharpening

The technique above guarantees that mass is conserved. However, the density field smooths out over time blurring the 0.5 iso-contour with the effect that we can no longer track the movement of the liquid surface accurately. We solve this

problem by manipulating ρ to sharpen the interface. Following [MMTD07], we first compute the mass change of each cell due to unit velocity along the x axis as

$$\delta_i^{x^+} = \int_{C_i} (\nabla \cdot (\rho[1,0,0]^T)\Delta T)dV, \quad (4)$$

$$\delta_i^{x^-} = \int_{C_i} (\nabla \cdot (\rho[-1,0,0]^T)\Delta T)dV, \quad (5)$$

where ΔT is the fictitious time step, which we set to 3 times that of the simulation time step in all of our examples. We discretize $\delta_i^{x^+}$ and $\delta_i^{x^-}$ using an upwind scheme to get

$$\delta_i^{x^+} \approx -(\rho_i - \rho_{i-(1,0,0)})\Delta x \Delta T, \quad (6)$$

$$\delta_i^{x^-} \approx -(\rho_{i+(1,0,0)} - \rho_i)\Delta x \Delta T. \quad (7)$$

The mass change due to unit velocity along y and z axes, $\delta_i^{y^+}$, $\delta_i^{y^-}$, $\delta_i^{z^+}$, and $\delta_i^{z^-}$ are computed similarly.

The maximum mass increase and mass decrease due to *any* unit velocity in each cell is:

$$\Delta T |\nabla \rho|_i^+ = \frac{1}{\Delta x^2}(\max(\max(\delta_i^{x^+},0)^2, \min(\delta_i^{x^-},0)^2) + \quad (8)$$

$$\max(\max(\delta_i^{y^+},0)^2, \min(\delta_i^{y^-},0)^2) + \quad (9)$$

$$\max(\max(\delta_i^{z^+},0)^2, \min(\delta_i^{z^-},0)^2))^{\frac{1}{2}} \quad (10)$$

and

$$\Delta T |\nabla \rho|_i^- = \frac{1}{\Delta x^2}(\max(\min(\delta_i^{x^+},0)^2, \max(\delta_i^{x^-},0)^2) + \quad (11)$$

$$\max(\min(\delta_i^{y^+},0)^2, \max(\delta_i^{y^-},0)^2) + \quad (12)$$

$$\max(\min(\delta_i^{z^+},0)^2, \max(\delta_i^{z^-},0)^2))^{\frac{1}{2}}. \quad (13)$$

We then compute

$$w_i(\rho) = (\rho_i - 0.5)^3 (1 - \min(1, \frac{\max_{j \in \aleph(C_i)}(|\rho_i - \rho_j|)}{\tau})), \quad (14)$$

where $\aleph(C_i)$ is the set of cells adjacent to C_i. The parameter τ controls the maximum difference in density between two adjacent cells, which we set to 0.4 as in [MMTD07]. This yields the following density correction:

$$\Delta \rho_i = w_i(\rho) \begin{cases} \Delta T |\nabla \rho|_i^+ & \text{if } w_i(\rho) \geq 0 \\ \Delta T |\nabla \rho|_i^- & \text{if } w_i(\rho) < 0 \end{cases}. \quad (15)$$

ρ can then be sharpened by updating the density of each cell using

$$\rho_i \leftarrow \rho_i + \Delta \rho_i. \quad (16)$$

This update sharpens the interface. However, it does not conserve mass. Mullen et al. [MMTD07] modify it to conserve mass by summing up the mass change due to this update across all cells. Then they distribute a fraction of the total mass change back to each cell based on a local area measure. This successfully conserves mass globally. One artifact of this approach is that mass moves far, potentially across the entire simulation domain. This problem can be reduced by re-distributing mass only within connected regions as proposed by [KLL*07]. However, even with this technique, local mass loss can still occur due to moving mass away from small features resulting in the disappearance of small surface details. Figure 3 top shows a scene where liquid balls are thrown into a pool of water. The mass conserving sharpening method of [MMTD07] transfers the mass from the liquid balls to the pool causing them to disappear mid-air. The left side of Figure 4 illustrates the situation in greater detail.

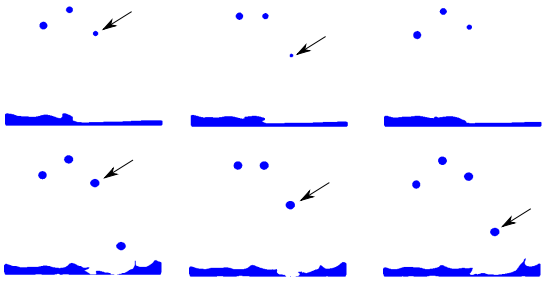

Figure 3: Top) The density sharpening method used by Mullen et al. [2007] conserves mass globally but not locally, causing the mass from a liquid ball (marked with the arrow) to disappear in mid air. Bottom) Our density sharpening method conserves mass both globally and locally preserving the mass of the liquid ball.

We propose a novel method to conserve mass during the sharpening phase that conserves mass both locally and globally. After evaluating $\Delta \rho_i$ using Equation 15, we modify it as follows:

$$\Delta \rho_i \leftarrow \begin{cases} -\rho_i & \text{if } \rho_i + \Delta \rho_i < 0 \text{ or } \rho_i < \varepsilon \\ 0 & \text{if } \rho_i > 0.5 \\ \Delta \rho_i & \text{otherwise,} \end{cases} \quad (17)$$

where we use $\varepsilon = 10^{-5}$ in all examples. In the first line we make sure that $\rho \geq 0$ at the next time step. We also clamp small positive densities to zero so that we do not have to apply the sharpening operator to this cell at the next time step, thus reducing computation cost. In the second line we make sure that cells with $\rho > 0.5$ are not modified. This way mass only moves from the air side to the liquid side. Then we update ρ_i using this modified $\Delta \rho_i$ in Equation 16.

We then add back $-\Delta \rho_i$ by using Algorithm 2. *TraceAlongField* determines where to put the lost mass. It starts from the cell center and follows the gradient field of the density $\nabla \rho$ until it reaches the 0.5 iso-contour. The tracing stops if a predefined distance $D\Delta x$ is reached or if it crosses a solid boundary. This is done using multiple forward Euler sub-steps. *ScatterValue* deposits $-\Delta \rho_i$ to nearby grid points using tri-linear weights. If a grid point is in a solid we set the corresponding weight to zero and re-normalize the weights. We use values of D between 1.1 to 3.1 in all of our examples.

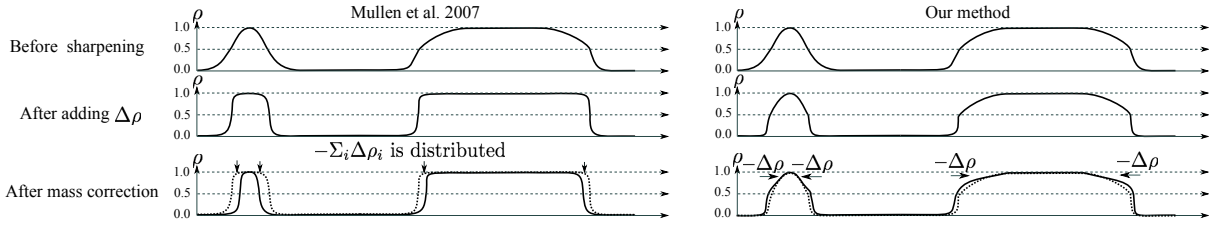

Figure 4: Comparison between the sharpening scheme of Mullen et al. 07 (Left) and ours (Right). The bigger hump has a large area with $0.5 < \rho < 1.0$. In this particular case, $-\sum_i \Delta\rho_i$ is negative in the scheme of Mullen et al. 2007. This negative mass is distributed to all the cells near interface, causing the smaller hump to become even smaller. Our method does not have this problem because $-\Delta\rho > 0$ is only added to the nearby cells around the 0.5 iso-contour. This prevents mass from being transported from one hump to another.

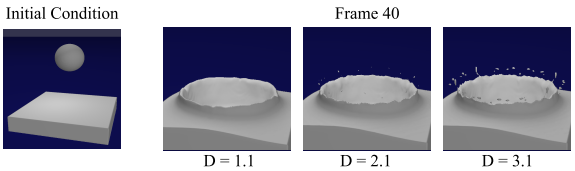

Figure 5: Left most) Initial condition of a ball dropping into a liquid pool. Others) Liquid surface at frame 40 of simulations with various values of parameter D.

Figure 5 shows the result of ball dropping into a pool using various values of D. Increasing D visually resembles the effect of surface tension.

Algorithm 2 Local mass conservation for sharpening

1: **for** each cell i **do**
2: $p = TraceAlongField(\text{Position}(i), \rho, \nabla\rho, D\Delta x)$
3: $ScatterValue(p, -\Delta\rho_i)$.
4: **end for**

3.6. Handling Solid Boundaries

So far, the method does not take solid fraction and solid velocity into account. We use $\mathbf{u}^s = (u^s, v^s, w^s)$ for the solid velocity and V_i for the fraction of non-solid matter, i.e. fluid and air in cell i. The scalars $V^f_{i+(\frac{1}{2},0,0)}$, $V^f_{i+(0,\frac{1}{2},0)}$, and $V^f_{i+(0,0,\frac{1}{2})}$ represent the fraction of non-solid area of the positive x, y, and z faces respectively.

$$\delta_i^{x^+} \approx -(\rho_i V^f_{i+(\frac{1}{2},0,0)} - \rho_{i-(1,0,0)} V^f_{i-(\frac{1}{2},0,0)})\Delta x \Delta T, \quad (18)$$

$$\delta_i^{x^-} \approx -(\rho_{i+(1,0,0)} V^f_{i+(\frac{1}{2},0,0)} - \rho_i V^f_{i-(\frac{1}{2},0,0)})\Delta x \Delta T. \quad (19)$$

During the simulation, the value of ρ_i can become larger than V_i in some cells which is a non-valid state. We handle the situation differently depending on whether the cell is partially solid ($V_i < 1$) or completely non-solid ($V_i = 1$). If the cell is partially solid, we first compute the excess density $d = \rho_i - V_i$. When then follow the gradient of the solid signed distance function away from the solid for a distance of $S\Delta x$ and scatter d to nearby grid points. After this we subtract d from ρ_i. This method keeps $\rho_i \leq V_i$ in most cells near solid boundary and guarantees $\rho_i = 0$ inside the solid. We use $S = 1$ in all of our examples. With this choice excess density gets removed from solid quickly enough to not cause visual artifacts. The case where $V_i = 1$ is handled in the incompressibility enforcement step described in the next section.

3.7. Enforcing Incompressibility

To enforce incompressibility, we first compute the pressure using a variational framework [BBB07] and then use the pressure gradient to make velocity field divergence free. The tricky part in our case is to determine the fraction of liquid in each cell. This fraction is used to decide whether a cell is included in the linear pressure solve. It is also needed in the ghost fluid method [ENGF03] to accurately handle the liquid-air boundary. However, we cannot directly use ρ because a cell with $V < 0.5$ will likely have $\rho < 0.5$ causing the solver to treat it erroneously as air. To fix this, we define ρ' as follows:

$$\rho'_i = \begin{cases} 0 & \text{if } V_i = 0 \\ \frac{\rho_i}{V_i} & \text{otherwise} \end{cases}. \quad (20)$$

Notice that cells that are completely solid ($V = 0$) have $\rho' = 0$. We then extrapolate ρ' from cells that have $V > 0$ to adjacent cells with $V = 0$ so they are included in the linear system. For the ghost fluid method, we also need a signed distance function near the free surface. We approximate this field by defining $\phi_i = -(\rho'_i - 0.5)\Delta x$ and use the method of [CM11a] to compute the coefficients of linear system for pressure projection.

To handle the cells with $\rho'_i > 1$ (whether or not $V = 1$ or $V < 1$), we add $\frac{\min(\lambda(\rho'_i - 1), \eta)}{\Delta x}$ to the divergence, where we use $\lambda = 0.5$ and $\eta = 1$ in all our examples. This artificial divergence pushes the excess density away from the cells whose $\rho' > 1$. Mullen et al. [MMTD07] also added this term to the divergence but with $\lambda = 1$ and $\eta = \infty$ which can cause stability problems when ρ' is much larger than 1. A scenario in which this happens is when liquid flows very fast towards a solid boundary and gets reflected due to our method for removing excess density from the solid.

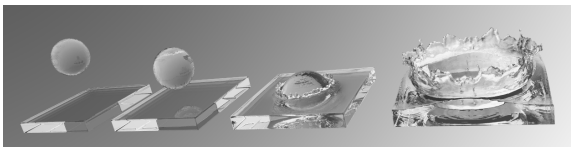

Figure 6: A crown splash simulated with our method at resolution of 128^3 cells. The density field is post-processed with the method proposed in this paper to enhance sub-grid details.

Adding additional divergence is important because in our case, $\rho' > 1$ results in visual volume loss. With the method described above, this problem gets gradually corrected over time. We solve for the pressure p with the multigrid method of [CM11a] which enforces separating solid boundary conditions. Finally, we use the pressure field to make the velocity field divergence free.

3.8. Density Post Processing

For rendering, we extract the triangle mesh of the 0.5 iso-contour of ρ using the marching cubes method [LC87]. This approach is typically used in level-set based liquid simulations as well to extract the zero contour of the signed distance field [EMF02].

The surface density ρ contains small scale details that are not captured by the 0.5 iso-contour. This problem is clearly visible on the right side of Figure 1 and in the bottom row of Figure 12. Here, the regions where $0 < \rho < 0.5$ represent features such as small splashes and thin sheets that are too small to be captured with the grid resolution used. In the level-set approach, these features are destroyed by the re-distancing step.

To bring out these small scale details in surface rendering, we propose a post processing method. An important observation is that regions in which $0 < \rho < 0.5$ do not necessarily represent small scale features. They also exist on the air side of the surface of large liquid regions. In the latter case, we want to leave ρ unchanged but in the former we want to scale up ρ so that the features appear in the 0.5 iso-surface.

To this end, we define an additional function $\gamma_i = 2\min(\rho_i, 0.5)$ and define the regions in which ρ needs to be scaled up as the regions where $\gamma \leq 0.5$. So far, the two cases above are not distinguished. However, this can be achieved by applying a Gaussian blur filter to γ. Now, since $\gamma > 0.5$ inside liquid, those values spread across the interface and cause γ to raise toward 1. In contrast, since $\gamma < 0.5$ everywhere inside small scale features, blurring will still result in γ being small. We then define $\rho_i'' = \frac{\rho_i}{\min(\max(\gamma_i, \theta), 1)}$ and extract the liquid surface as the 0.5 iso-surface of this modified density field.

The effect of this post processing method is shown in Figure 6 and in the accompanying video. We used $\sigma = 2\Delta x$ for the Gaussian blur filter and $\theta = 0.01$ in this example. A way to improve the results further would be to apply thinning to the parts of the surface that come from region with $\rho < 0.5$ in order to compensate for the density up-scaling. This is part of our future work.

4. Results

We implemented our method using CUDA and ran the simulations on an NVIDIA GTX 680. For all the examples we used a time step size of $1/30s$, $\Delta x = 0.05m$, gravity $10m/s^2$ and $D = 2.1$. Density post-processing was turned off unless otherwise stated. Our code run at interactive rates in all examples. The simulation times and CFL numbers are listed in Table 2. Parameter tuning to get visually appealing results did not take much time.

We compared our method with the particle level set (PLS) approach [EMF02]. The results of this comparison are shown in Figure 7 and in the accompanying video. Our method conserves the liquid's mass as expected and prevents the water level from decreasing. In contrast, with PLS, most of the liquid disappears in the course of the simulation due to the large time step size used. We used the PLS implementation of [MF] and set the number of particles per cell to 64.

Figure 1 shows a simulation of a liquid jet in a rectangular tank. The jet has a very fast flow rate and generates fast moving liquid splashes and sheets. The accompanying video of this example also shows how we fill the tank from a completely dry state by adding liquid balls. These are difficult cases for level set approaches while our method handles them without any problem.

With our approach we were able to create, for the first time, a 3d water demo that is both simulated and ray-traced in real time. The scene starting with a dam break initial setup and subsequent additions of water balls is shown in Figure 9. We achieved a frame rate of over 30fps with two GPUs, one for simulation and one for ray-tracing.

Figures 8 and 12 show a dam break and dropping balls in a spherical container. In the accompanying video we shake the container. These examples demonstrate the ability of our method to simulate liquid in a non-axis aligned moving container.

One way coupling with fast moving solids is shown in Figure 11 and the accompanying video. Several solid objects move at high speed across the tank sloshing the liquid up to the air. Our method conserves mass and prevents volume loss in this difficult case as well.

We computed the mass and the volume enclosed by the 0.5 iso-contour of the liquid over time in various examples. The corresponding plots are shown in Figure 10. The total mass is computed by integrating ρ over the whole simulation grid. To measure the volume we used marching cubes to extract the 0.5 contour triangle mesh of ρ and determined the enclosed volume. Our method conserves mass in all examples

Figure 7: Liquid ball dropped inside a box simulated on a 128^3 resolution grid. Left) Initial condition. Shape of the surface at frame 40 computed with our method (middle) and with PLS (right). PLS loses most of the mass due to the large time step used.

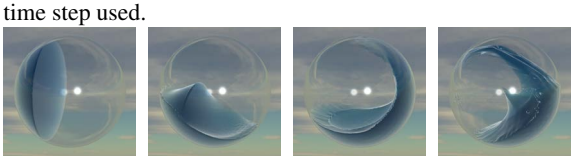

Figure 8: Snapshots of a dam breaking scene in a spherical container simulated at a resolution of 128^3 cells.

and generally keeps the volume close to the true liquid volume. However, there are several situations where our method loses volume visually. One such case is when a liquid ball hits the ground and spreads out until it becomes thinner than the grid spacing. Even though the density values are non-zero, marching cubes does not generate surface meshes in those regions. Another case is when the ratio of surface area to volume is large. In this case, there are large regions with $\rho < 0.5$ that do not contribute to the volume because the 0.5 iso-contour is empty. However, in contrast to PLS, when such features join the main body of water again, they correctly contribute to its volume so that the global level remains constant.

Name	Grid	CFL	Time (ms)
Figure 1	256x128x128	25	113.2
Figure 5	128x128x128	8	54.2
Figure 9	128x128x64	24	26.7
Figure 8	128x128x128	14	53.4
Figure 11	256x128x128	32	118.6
Figure 12	128x128x128	20	53.8

Table 2: CFL Number and simulation time per frame for various examples. We use the time step of 1/30s in all examples. All timing are done on GTX680.

5. Conclusion and Discussion

We proposed a method for simulating liquids that conserves mass and is effective in keeping the volume defined by the 0.5 iso-contour close to constant. We have demonstrated the strength of our technique in various scenarios. The method has its limitations as well. First, although our sharpening scheme ensures that the $\rho = 0.5$ interface is sharp, it does not modify regions where $\rho > 0.5$. It could theoretically be possible that the region with ρ slightly above 0.5 expands so that the volume defined by the 0.5 iso-contour grows by a factor of two while keeping its original mass. This, however, does not happen in practice because the divergence free velocity field prevents the liquid from expanding significantly. An alternative to our sharpening method is to perform anti-diffusion step [SHA11], which is an interesting avenue for future work. Another limitation is the possibility of losing local volume temporarily as discussed in the previous section. The density post processing method we proposed is an effective way to alleviate this effect. Even though our method cannot guarantee complete volume conservation at all times, it reduces this problem significantly in comparison to all the previous methods we have investigated.

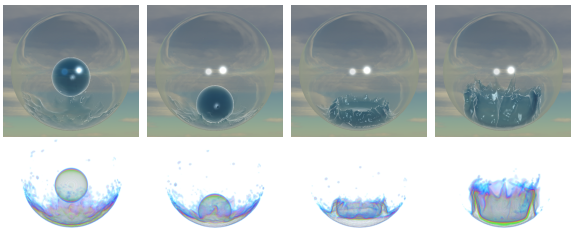

Figure 12: Simulation of a liquid ball dropping inside a spherical container at a resolution of 128^3 cells. Top) Surface rendering. Bottom) Volume rendering, showing many sub-grid details not visible in the surface rendering.

References

[AGDJ08] ANDERSON J. C., GARTH C., DUCHAINEAU M. A., JOY K.: Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum (Proc. of Eurographics/IEEE-VGTC Symposium on Visualization 2008) 27*, 3 (2008). 2

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26* (July 2007). 2

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493. 2

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. In *Proc. SIGGRAPH* (2007), p. 100. 6

[BGOS05] BARGTEIL A. W., GOKTEKIN T. G., O'BRIEN J. F., STRAIN J. A.: A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics* (2005). 1, 2

[CM11a] CHENTANEZ N., MÜLLER M.: A multigrid fluid pressure solver handling separating solid boundary conditions. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 83–90. 6, 7

[CM11b] CHENTANEZ N., MÜLLER M.: Real-time eulerian water simulation using a restricted tall cell grid. In *Proc. SIGGRAPH* (2011), p. 82. 3

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. In *Proc. SIGGRAPH* (2002), pp. 736–744. 1, 2, 7

[ENGF03] ENRIGHT D., NGUYEN D., GIBOU F., FEDKIW R.: Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In

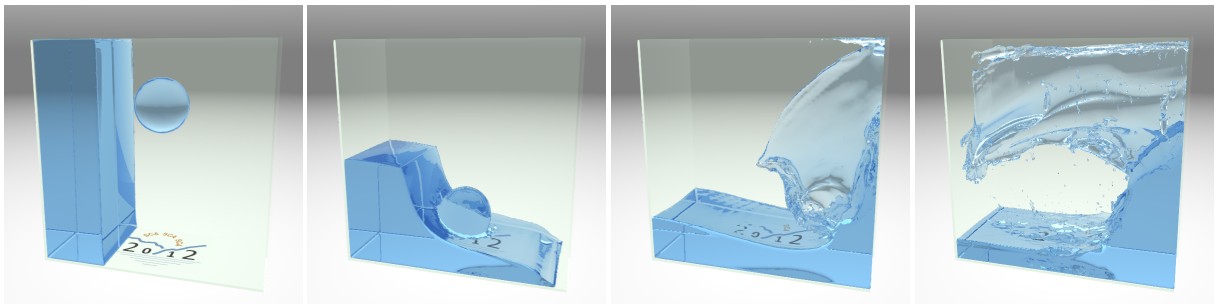

Figure 9: Snapshots of a dam break and ball drop scene in a glass box at a resolution of $128x128x64$ cells. The simulation and raytracing run in parallel on dual NVIDIA GTX680 GPUs in real-time at 30fps.

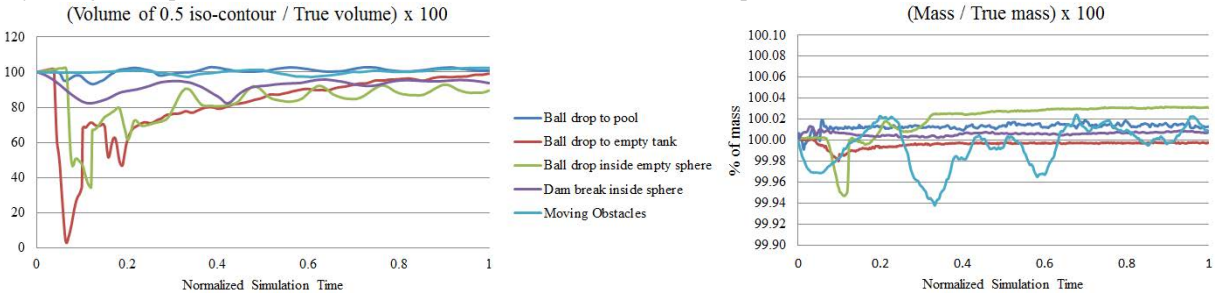

Figure 10: Time evolution of mass and volume relative to the ground truth in various examples. In all examples, the time axis is re-scaled to start at 0 and end at 1. The volume of the liquid was measured by evaluating the volume enclosed by the marching cubes 0.5 iso-contour triangle mesh of ρ. Our method conserves mass up to an arithmetic error. The volume generally stays close to the true value, but can decrease significantly in some cases. For example, when the liquid ball spreads out on the floor and becomes thinner than the grid spacing or when the ratio of surface area to volume is large.

In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003Ű45144. ASME (2003), pp. 2003–45144. 6

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. SIGGRAPH* (Aug. 2001), pp. 23–30. 1, 2

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483. 2

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. In *the Proceedings of ACM SIGGRAPH 2003* (July 2003), pp. 708–715. 2

[HK10] HEO N., KO H.-S.: Detail-preserving fully-eulerian interface tracking framework. *ACM Trans. Graph. 29* (December 2010), 176:1–176:8. 2

[HN81] HIRT C. W., NICHOLS B. D.: Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, 1 (Jan. 1981), 201–225. 2

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids 8* (1965), 2182–2189. 3

[ISF07] IRVING G., SCHROEDER C., FEDKIW R.: Volume conserving finite element simulations of deformable models. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2

[JRW07] JEONG W.-K., ROSS, WHITAKER T.: A fast eikonal equation solver for parallel systems. In *SIAM conference on Computational Science and Engineering* (2007). 3

[KLL*07] KIM B., LIU Y., LLAMAS I., JIAO X., ROSSIGNAC J.: Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. 26* (July 2007). 2, 5

[KLLR05] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Flowfixer: Using bfecc for fluid simulation. In *NPH* (2005), pp. 51–56. 2

[KPyNS10] KANG N., PARK J., YONG NOH J., SHIN S. Y.: A hybrid approach to multiple fluid simulation using volume fractions. *Comput. Graph. Forum 29*, 2 (2010), 685–694. 2

[KSK08] KIM D., SONG O.-Y., KO H.-S.: A semi-lagrangian cip fluid solver without dimensional splitting. *Computer Graphics Forum 27*, 2 (April 2008), 467–475. 2

[LAF11] LENTINE M., AANJANEYA M., FEDKIW R.: Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, August 2011), SCA '11, ACM, pp. 91–100. 1, 2, 3, 4

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21* (August 1987), 163–169. 7

[LGF11] LENTINE M., GRÉTARSSON J. T., FEDKIW R.: An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys. 230* (April 2011), 2857–2879. 2, 3, 4

[LR09] LONG B., REINHARD E.: Real-time fluid simulation using discrete sine/cosine transforms. In *Proc. ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2009), pp. 99–106. 2

[MÖ9] MÜLLER M.: Fast and robust tracking of fluid surfaces. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009). 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *ACM*

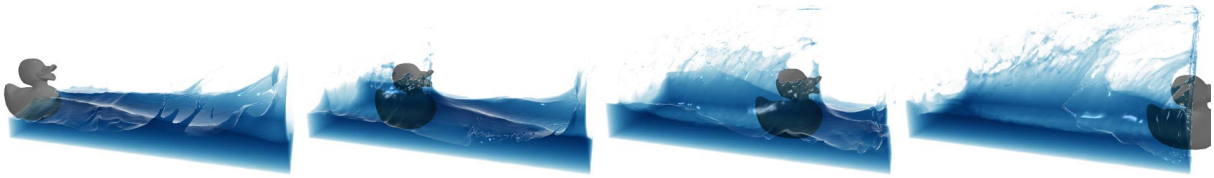

Figure 11: Simulation of a solid duck moving across a liquid tank at high speed, sloshing the liquid up in the air on a grid of 256x128x128 cells.

SIGGRAPH/Eurographics Symposium on Computer Animation (2003), pp. 154–159. 2

[MF] MOKBERI E., FALOUTSOS P.: A particle level set library. 7

[MMTD07] MULLEN P., MCKENZIE A., TONG Y., DESBRUN M.: A variational approach to eulerian geometry processing. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 1, 2, 3, 5, 6

[PP04] PILLIOD JR. J. E., PUCKETT E. G.: Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *J. Comput. Phys. 199*, 2 (Sept. 2004), 465–502. 2

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A. E., WHITAKER R. T.: Particle-based simulation of fluids. *Comput. Graph. Forum 22*, 3 (2003), 401–410. 2

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable MacCormack method. *J. Sci. Comput. 35*, 2-3 (2008), 350–371. 2

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Trans. Graph. 30* (Aug. 2011), 81:1–81:8. 2

[SHA11] SO K. K., HU X. Y., ADAMS N. A.: Anti-diffusion method for interface steepening in two-phase incompressible flow. *J. Comput. Phys. 230*, 13 (June 2011), 5155–5177. 8

[SP00] SUSSMAN M., PUCKETT E. G.: A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys. 162*, 2 (Aug. 2000), 301–337. 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. In *Proc. SIGGRAPH* (2009), pp. 1–6. 2

[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH* (Aug. 1999), pp. 121–128. 2

[TR04] THÜREY N., RÜDE U.: Free Surface Lattice-Boltzmann fluid simulations with and without level sets. *Proc. of Vision, Modelling, and Visualization VMV* (2004), 199–207. 2

[TR09] THÜREY N., RÜDE U.: Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Computing and Visualization in Science 12 (5)* (2009). 2

[WMFB11] WOJTAN C., MÜLLER-FISCHER M., BROCHU T.: Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 8:1–8:84. 1

[WTGT10] WOJTAN C., THÜREY N., GROSS M., TURK G.: Physics-inspired topology changes for thin fluid features. In *Proc. SIGGRAPH* (2010), no. 4, pp. 1–8. 2

[YT10] YU J., TURK G.: Enhancing fluid animation with adaptive, controllable and intermittent turbulence. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010). 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *Proc. SIGGRAPH* (2005), pp. 965–972. 2