ENHANCED VOLUME CONTROL IN MULTIPHASE FLUID SIMULATIONS

Túlio Ligneul Santos

Antonio Alberto Fernandes de Oliveira

Paulo Roma Cavalcanti

tulioligneul@poli.ufrj.br

oliveira@cos.ufrj.br

roma@lcg.ufrj.br

COPPE-Sistemas, Universidade Federal do Rio de Janeiro

Cidade Universitária - Ilha do Fundão,
Caixa Postal 68530, CEP 21945-970,
Rio de Janeiro, RJ, Brazil

Abstract. *In this work, it is employed the regional level set as a solution to the problem of tracking interfaces in multiphase fluid simulations, with special attention to the treatment of the thin film between distinct phases. In this context, properties of each phase are tracked as they are merged or split, and a new algorithm to apply these operations is proposed. Furthermore, it is brought up two new methods for handling undesired changes in volume or inducing controlled volume changes, being the first a selfish method and the latter a globally optimized one. In both cases, localized volume corrections are performed through the evolution of interfaces based on auxiliary velocities that are estimated near them and are determined from the historical data of the volumetric error. Also, it is defined how the volume of each phase should vary due to fluid inflows or outflows. Finally, particles are generated from small phases that, because of limitations of the adopted discretization, would have disappeared.*

Keywords: *Multiphase fluid simulation, Volume control, Regional level set*

1 INTRODUCTION

In computer graphics, there are two main approaches to simulate fluid flows. The first is the lagrangian perspective (Müller et al., 2003), where the fluid is represented as a set of particles. The second, the eulerian approach, evolves the state of the fluid on each point of the space it fills, and has Foster et al. (1996, 2001) and Stam (1999) as its initial works. Alongside these approaches, there are hybrid models where, for instance, particles help track the surface of an eulerian fluid (Enright et al., 2002) or even interfere with the simulation (Hong et al., 2008; Losasso et al., 2008). For example, Greenwood et al. (2004) and Guendelman et al. (2005) use escaped particles from the air into a liquid medium, as classified by the particle level set (Enright et al., 2002), to create air bubbles. Still, Mihalef (2007) does the same, but employing marker level sets (Mihalef et al., 2009). In this work, it is employed a hybrid method where the simulation is eulerian and particles are employed to avoid volume losses.

In regards to the context of multiphase simulations, it is sought to simulate situations where, for instance, oil floats over water, soap bubbles fly through the air, or air bubbles are in contact while separated by a liquid film. Note that this film is an important feature as it defines whether two adjacent phases of the same fluid will merge or not. Thereby, it is needed means to evolve each film over time and detect if certain conditions are met so it ruptures and the two adjacent phases merge. Altogether, films can be considered as interfaces between two regions so that tracking their positions and shapes define the current state of the fluids system.

Finally, errors introduced by the numeric simulation should be considered. One of the effects it induces is the loss of volume over time that, in multiphase simulations, can even bring about the removal of a whole phase from the simulation. Consequently, it is reasonable to consider a way to maintain each phase's original volume.

Therefore, a flood fill segmentation algorithm is proposed to detect different phases, while splitting and merging phases as needed. In this last case, it is established how to calculate a film thickness value for the resulting phase. Also, two methods to correct undesired volume changes or to make it vary in a controlled way are proposed. While the first one consists of a selfish method that locally care for the volume error of a single phase, the second tries the minimize the global error in the system. In addition, it is defined how to update the target volume of each phase according to fluid inflows and outflows in relation to the grid that contains all simulated fluids. Additionally, the particles' volumes are corrected based on their original phases. Finally, it is introduced a procedure that generates triangle meshes with their respective render parameters by iterating trough each interface, exports them to an external application and renders frames to build an animation.

1.1 Related work

When working with multiphase simulation, a known problem is that, to work with more than two materials, the classic approach to accompanying interfaces, the level set (Osher et al., 2003) must be extended, as it only represents the interface between two materials. Accordingly, the techniques introduced by Losasso et al. (2006) and Vese et al. (2002) evolve the interfaces of multiple fluids, despite having increased complexity as the number of different materials grows. The first

one employs one level set per material and, as different components of the same material are considered as an unique phase, they are always merged when in contact. Besides, as the evolution of each level set in done independently, it can generate contradictions in the interface's representation that must be corrected. In the second approach, n level sets represent up to 2^n phases. For example, two level sets can represent up to four distinct phases, classified by sign combinations of a scalar function (i.e. "$++$", "$+-$", "$-+$" e "$--$"). Still, this method presents undesired artifacts, specially where two regions intersect.

Other works have also proposed solutions to evolve thin features. Wojtan et al. (2009, 2010) developed hybrids of grid and explicit surfaces to perform topology changes, even though the process is overly complex. Also, Brochu et al. (2010) disturbed samples to capture thin features. Despite allowing the existence of thin features at any position, these methods only work with one or two phase flows, while employing the traditional level set method.

To reduce undesired volume losses, it is possible to use more robust advection models, such as back and forth error compensation and correction (Dupont et al., 2003; Kim et al., 2005), modified MacCormack (Selle et al., 2008), and the use of cubic interpolated polynomials (Song et al., 2005; Takahashi et al., 2003). Besides, hybrid approaches like the particle level set (Enright et al., 2002) can also be applied. Lastly, it is possible to combine the level set with volume of fluid methods (Sussman, 2003). In this case, there is no volume loss as volume fractions are tracked in each grid cell.

Nonetheless, this work's main reference in relation to volume correction is the work of Kim et al. (2007), that also employs the regional level set and can be used together with the aforementioned methods to improve volume correction or induce controlled volume changes. However, in his work, the focus is on the volume of air bubbles only. Also, he doesn't present a solution for regions smaller than a threshold and, as a result, small droplets still disappear. Trough his method, it is enforced a divergent on the velocity field of each region so that a region is omnidirectionally inflated or deflated to correct its volume.

1.2 Technique overview

In regards to the multiphase fluid simulation itself, values of pressure and velocity are sampled in a regular MAC grid (Harlow et al., 1965). Taking in consideration a variable density profile, in order to solve a Poisson equation with variable coefficients, it is employed the method described by Kang et al. (2000), which uses techniques developed by Liu et al. (2000). However, in this work, the fluid's viscosity is disconsidered. Like Kim (2010), the ghost fluid method (Hong et al., 2005) accounts for surface tension forces.

To evolve fluid interfaces, the regional level set (Zheng et al., 2009) is employed. This technique implicitly defines the interface between each phase and is independent of the number of materials in the simulation. As done by Kim (2010), a connectivity graph identifies each region and film and enables the tracking of fluid properties, such as the target volume, over time.

In order to perform volume correction, after the advection of the regional level set, an additional advection step is performed using an auxiliary velocity field whose values are non-zero only near

interfaces. The way these velocities are defined by each proposed volume control method are different. For the selfish method, these velocities intend to correct only the the volume of the region that contains the position where each value is estimated. In contrast, in the global method, an ideal flow is estimated per interface and optimized so the global volume error is minimized.

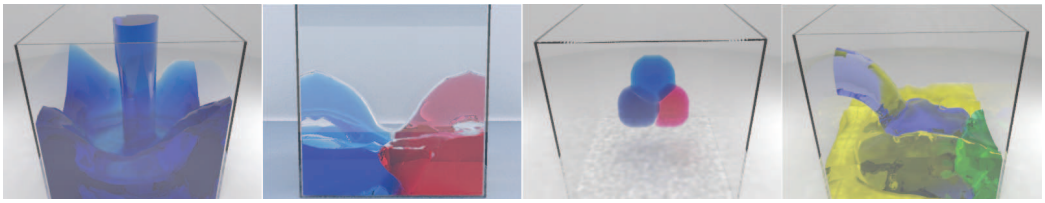

Figure 1: Examples of resulting simulations employing, at least, two different fluids.

For the implementation, the Nvidia's CUDA technology is employed to minimally accelerate the simulation and debug views. To achieve a final visualization, triangles meshes are exported each frame from the main application and imported to the Blender suite of tools. Finally, frames rendered with the Yafaray raytracer and are combined into an animation. Figure 1 shows final renders of simulations with two, three and four fluids.

2 TECHNICAL BACKGROUND

In this section, the techniques employed to perform the simulation are detailed.

2.1 Regional level set

The regional function is defined as:

$$f_R(\vec{x}) = \langle\, r(\vec{x}), \phi(\vec{x})\, \rangle\,, \tag{1}$$

where $\vec{x}$ is a position in space, $r(\vec{x})$ is the identifier of the region that contains $\vec{x}$ and $\phi(\vec{x})$ is the distance of $\vec{x}$ to the nearest point in an interface. By convention, estimates of the function are kept at the center of each cell. Also, a known property is that the curvature at a point of a level set is defined as $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|| \nabla \phi ||} \right)$. In addition to the regional function, operators introduced by Zheng et al. (2009) are employed to perform regional tuple calculations. Those used in linear interpolations are the sum and multiplication by a scalar operators:

$$\langle r_1, \phi_1 \rangle + \langle r_2, \phi_2 \rangle = \begin{cases} \langle r_1, \phi_1 + \phi_2 \rangle\,, & \text{if } r_1 = r_2 \\ \langle r_1, \phi_1 - \phi_2 \rangle\,, & \text{if } \phi_1 \geq \phi_2 \\ \langle r_2, \phi_2 - \phi_1 \rangle\,, & \text{if } \phi_1 < \phi_2 \end{cases} \tag{2}$$

$$\alpha \langle r_1, \phi_1 \rangle = \langle r_1, \alpha \phi_1 \rangle\,, \alpha \in \mathbb{R}^+ \tag{3}$$

Nevertheless, it can be observed that the regional sum operator is not commutative. For example, $(\langle 1,5\rangle+\langle 2,3\rangle)+\langle 3,1\rangle = \langle 1,2\rangle+\langle 3,1\rangle = \langle 1,1\rangle$, but $\langle 1,5\rangle+(\langle 2,3\rangle+\langle 3,1\rangle) = \langle 1,5\rangle+\langle 2,2\rangle = \langle 1,3\rangle$. As a result, a trilinear interpolation performed to obtain a regional tuple at any position stops being consistent as the result depends on which axis is interpolated first. To avoid that this happens, the operator proposed by Kim (2010) sums multiple tuples at once, while maintaining the addition's commutative property. For instance, in the case of a bilinear interpolation:

$$\begin{aligned}\langle r,\phi\rangle =& \langle r_1,(1-m_x)(1-m_y)\phi_1\rangle + \langle r_2, m_x(1-m_y)\phi_2\rangle \\ &+ \langle r_3,(1-m_x)m_y\phi_3\rangle\,,\langle r_4, m_x m_y\phi_4\rangle = \sum_{i=1}^{n}\langle r_i,\phi_i'\rangle\,,\end{aligned} \tag{4}$$

Firstly, a partial sum per region is computed as $\Psi_k = \sum_{\forall r_i=r_k}\phi_i$, and the two biggest values of Ψ are chosen, i.e, it is defined Ψ_1 and Ψ_2, such that $\Psi_1 \geq \Psi_2 \geq \Psi_k$, with $k = 3,...,m$. Then, the result is obtained as:

$$\sum_{i=1}^{n}\langle r_i,\phi_i\rangle = \sum_{k=1}^{m}\langle r_k,\Psi_k\rangle = \langle r_1,\Psi_1-\Psi_2\rangle\,. \tag{5}$$

Altogether, it can be observed that, besides the guarantee of a mathematical property, it leads to a more regular segmentation of the space, as seen on the Fig. 2.

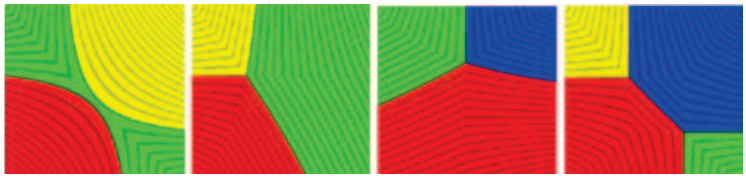

Figure 2: Example results of the bilinear interpolation between four regional tuples using the operator proposed by Kim (2010).

To create an initial regional function distribution on the computational grid, as each material is defined, a region identifier is created to represent it and to classify grid cells that are inside such material. Then, $\phi = \Delta x/2$ is defined for every cell, where Δx is the edge length of a cubic cell, and a re-initialization routine is performed. This task updates the ϕ value of each cell so that it is a distance field, and is always executed after each second order Runge-Kutta level set advection step because it usually doesn't preserve the fact that ϕ is a distance.

The re-initialization routine, at first, for each cell that has a neighbour of a different region, discovers the exact position of the interface by interpolating the cells' center positions using their current estimates of ϕ to find where $\phi = 0$ and setting the new value of ϕ as the distance from the cell center to the interface. After that, it extrapolates the values of the cells near the interface to those further from them. To take advantage of the parallelization allowed by the CUDA technology, this task is performed in the following way: having one thread per cell, for each cell, it is tested if an adjacent cell already has a defined distance value. It this is true, it computes the distance from its center to the nearest point on surface relative to the adjacent cell and chooses for itself the

minimum among all the estimates calculated in relation to each adjacent cell. Therefore, in $n-1$ iterations, cells distant up to $n\Delta x$ from the surface will have their values defined.

2.2 Calculating volumes and areas

To calculate the volume and area of each fluid region, following the approach of Kim et al. (2007), the 8-point Gauss quadrature method is employed to integrate, in the volume's case, the heaviside function:

$$H(\Phi) = \begin{cases} 0, & \text{if } \Phi \leq -\epsilon. \\ 1, & \text{if } \Phi \geq \epsilon. \\ 0.5 + 0.75\dfrac{\Phi}{\epsilon} - 0.25\dfrac{\Phi^3}{\epsilon^3}, & \text{if } \mid\mid \Phi \mid\mid < \epsilon. \end{cases}, \tag{6}$$

where $\epsilon = 0.1\Delta x$ is the thickness of the film and $\Phi(x, y, z)$ is equal to the distance function for positions inside the region in question and, if otherwise, it is equal to the distance function's negative.

For example, a cell with index (i, j, k) whose minimum and maximum points are, respectively, (x_i, y_j, z_k) and $(x_{i+1}, y_{j+1}, z_{k+1})$, has its volume computed trough:

$$Vol = \int_V H(\Phi(x,y,z))dV = \int_{z_k}^{z_{k+1}} \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} H(\Phi(x,y,z))dx\, dy\, dz. \tag{7}$$

Note that to use the tabled values of the Gauss-Legendre quadrature, a linear transformation must be performed to change the integration limits so to integrate in the interval $[1, -1]$. Then the volume is computed trough the discretization:

$$Vol = \frac{\Delta x^3}{8} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k \; g(x_i, y_j, z_k), \tag{8}$$

where the weights w_i, w_j, w_k and positions (x_i, y_j, z_k) are known tabled values.

Analogously, the area of the surface of a fluid region is calculated trough the derivative of the heaviside function.

2.3 Film handling

It is assumed that there is a thin film composed of a predefined material separating every two different regions. This way, adjacent regions of the same material can exist and, eventually, be merged together. To find out when there is a film separating two regions, film thicknesses are evolved and it is evaluated when a film should rupture so that the adjacent regions of the same fluid merge.

However, it is known that the film's behavior is extremely complex. In fact, it gets thinner due to stretching, drainage and evaporation while its rupture can be delayed because of pressure

variations or superficial tension forces that minimize the interface's area. Despite this inherent complexity, the approximate method proposed by Zheng et al., 2009 is employed to evolve film's thicknesses and test when two regions should merge together.

To start, each film is given a constant thickness $h = 0.1\Delta x$, to be used with visualization purposes, since small thickness changes wouldn't be visible on the rendered image. Then it is defined, by region, a fictitious thickness l that will be evolved over time. So, each iteration, the l_r of each region r is updated trough surface deformation and drainage:

$$l_r^t = l_r^{t-1} - l^{t-1}\frac{(A_r^t - A_r^{t-1})}{A_r^t} - \left(\frac{l_0 - l_{min}}{t_{persist}}\right)\Delta t, \tag{9}$$

where l_r^t and A_r^t are, respectively, the thickness and the area of a region r at the t-th time step, $l_{min} = 0.01\Delta x$ is the smallest thickness a region can have, $l_0 = 0.1\Delta x$ is the initial film thickness, $t_{persist}$ is the maximum duration a region can stay without having its film ruptured. Nevertheless, the effect of the drainage, which ensures a linear variation in thickness over time, is only taken into account when two adjacent regions are of the same material.

As film thicknesses are updated every iteration, they are tested against rupture conditions. Indeed, if the average thickness of two regions of the same material is less than the threshold l_{min}, the film that separates them is broken and the two regions merge. Also, the film thickness of the resulting region is equal to the maximum between the original regions' thicknesses.

Additionally, if the density of the film is smaller than that of adjacent regions of the same fluid, those regions are merged. This latter condition is intended to depict situations where, for example, a drop of water falls into a large body of water. As the fluid that separates the two water regions consists of air, which is much less dense, the junction of the two regions is instantaneous.

2.4 Regional graph

Each film and region is uniquely identified at the iteration t trough the regional graph G^t. Its nodes are the different regions and, its edges link adjacent regions. It is computed by checking each cell's neighbourhood for cells of regions different from its own. For n existing regions, an upper triangular matrix is built where, for two regions r_i and r_j, with $i < j$, the position (i, j) keeps a boolean value indicative of the adjacency between these regions.

The regional graph addresses the problem of how to track properties from the regional distribution from one iteration to the next. In fact, graphs of successive iterations are mapped so properties can be transferred between them. Trough this mapping, it is retrieved which regions merged and which were split. Also, if a region was not mapped into at least one region of the next iteration, it is assumed that it was removed during advection and a particle is created with the same characteristics of such region.

One of the tracked properties is the fictitious film thickness. If a region is not the product of a merge or split operation, it is transferred directly from one graph to the other. If, otherwise, it is the product of a split operation, it is set as the initial value l_0. Yet, when it is created as a result of a film rupture, its thickness is the maximum between the thicknesses of the merged regions.

Another property tracked is the previous area of a region, which is used to update its film's thickness due to stretching. Analogously to the case of the thickness itself, this value is directly transferred when a region is not the outcome of a junction or a division, but, otherwise, it is set equal to the region's current area.

Finally, the target volume of each region is tracked while guaranteeing that the volume of each region is conserved over time. As a matter of fact, when a region splits, the target volume of the resulting region is proportional to the volume fraction of each new region in relation to the original region's volume. Also, when two regions merge, the target volume of the resulting region is the sum of those of the regions involved.

2.5 Particles

Spherical particles are employed to confer more realistic physical and visual features to the simulation, while aiding volume correction. Meanwhile, the focus is to substitute small fluid components that would otherwise be removed from the simulation due to the adopted space discretization. That being said, each particle is defined by its radius, center position, type of fluid and velocity.

To move each particle trough the t-th time step, their velocities and positions are integrated over time using the forward euler integration scheme. Also, following the works of Kim (2010) and Cleary et al. (2007), to propel the particles, it is accounted the effects of gravity trough the force:

$$\vec{F_b} = Vol_p(\rho_p - \rho_{fluidAt(\vec{x}_p)})\vec{g}, \tag{10}$$

where Vol_p and ρ_p are, respectively, the volume and density of the particle p, and $\vec{g} = 9.81m/s^2(-\hat{j})$. Besides, in accordance with the research of Busaryev et al. (2012),a drag force is applied:

$$\vec{F_d} = c_d r_p^2 \parallel \vec{u}(\vec{x}_p) - \vec{u}_p \parallel (\vec{u}(\vec{x}_p) - \vec{u}_p), \tag{11}$$

where $c_d \in [0.05, 0.5]\, kg/m^3$, r_p is the radius of p, and $\vec{u}(\vec{x}_p)$ is a velocity vector estimated at p's position using the velocity components sampled on the grid.

In this work, when a region reaches a minimum volume $Vol_p < Vol_{min}$, with $Vol_{min} = 27\Delta x^3$, a particle is created to avoid regions becoming too small and disappearing. However, if any region does so, i.e., if a region was removed during the advection phase, a particle is created to replace it.

When a particle is created, it is positioned on the estimated center of the original region. This position is defined as the average of the positions of the centers of the region's cells, weighted by their respective ϕ values. Their radii are determined so that they have the same volume as their original region. Also, to allow corrections of the particles' volumes, a target radius is defined for each particle so that its target volume is the same as the one from its original region. As for a region converted into a particle, it is marked as inactive to no longer be considered, for example, when testing conditions for film rupture. Plus, its target volume is reduced to zero, so that a volume controlling method would shrink it until it disappears.

Moreover, particles can be converted back to regions in two specific cases. In the first one, a particle contacts a region of the same material. This is tested by comparing the material of the region that contains the position of center of the particle and the particle's material. If both are the same, the particle will be united to the region. The second situation represents when the particle has volume $Vol_p >= 1.5Vol_{min}$. Note that there is a discrepancy of $0.5Vol_{min}$ in relation to the minimum volume that a region can have. This is done to avoid successive changes between the representations.

So, to convert a particle to a region, the regional level set is rebuilt near it and the materials of the affected cells are updated. Hence, in parallel, each cell checks if it is near any particle that will be converted. If it is distant up to $r_p + \sqrt{3}\Delta x$ from the center of such particle, its regional tuple and fluid type are modified accordingly.

2.6 Fluid dynamics

In order to perform multiphase simulations, a variable coefficient Poisson equation should be solved, since the density is no longer constant inside the simulation grid. Thereby, a simplified version of the method presented by Liu et al. (2000) is used, as it is assumed that the density is continuous across interfaces and surface tension forces are applied trough the ghost fluid method (Hong et al., 2005). Hence, the conjugate gradient method (Hestenes et al., 1952) can be applied to solve a linear system in which each cell is discretized as:

$$\frac{\Delta t}{\Delta x^2}\left[\left(\frac{p_{i+1,j,k}-p_{i,j,k}}{\rho_{i+1/2,j,k}}\right)+\left(\frac{p_{i-1,j,k}-p_{i,j,k}}{\rho_{i-1/2,j,k}}\right)+\left(\frac{p_{i,j+1,k}-p_{i,j,k}}{\rho_{i,j+1/2,k}}\right)+\left(\frac{p_{i,j-1,k}-p_{i,j,k}}{\rho_{i,j-1/2,k}}\right)\right.$$
$$\left.\left(\frac{p_{i,j,k}-p_{i,j,k+1}}{\rho_{i,j,k+1/2}}\right)+\left(\frac{p_{i,j,k-1}-p_{i,j,k}}{\rho_{i,j,k-1/2}}\right)\right]=\nabla\cdot\vec{u}_{i,j,k}\left(\pm\frac{\Delta t}{\rho}\frac{J}{\Delta x^2}\right), \tag{12}$$

where $J = \sigma\kappa_I$ is the jump of pressure across the interface, σ is the surface tension coefficient between the two materials and κ_I is the curvature at the interface.

In the above equation, the term $\dfrac{\Delta t}{\rho}\dfrac{J}{\Delta x^2}$ accounts for surface tension forces in accordance with the ghost fluid method. Basically, what this method does is extrapolate pressures profiles through interfaces so that it makes the derivative of the pressure continuous across the interface even tough the pressure itself is not. The implementation of the method itself is fairly simple: when using the previous equation for a specific cell, for each of its adjacent cells that are in a different region than its own, the term between parenthesis is added or subtracted to the Eq. (12), depending if the adjacent cell is the next or the previous cell in a given direction. Also, ρ must be estimated at the face between the two cells in question, which is where the derivatives are computed.

Knowing how to solve the equation, a time step duration Δt have to be defined for the t-th time step to ensure the convergence of the numeric simulation. Following the approach of Kang et al. (2000), the CFL condition (Courant et al., 1928), which guarantees the convergence, is enforced trough the equation:

$$\Delta t\left(\frac{A_{cfl}+\sqrt{A_{cfl}^2+4G_{cfl}+4S_{cfl}}}{2}\right)\leq 1, \tag{13}$$

where $A_{cfl} = \frac{|u|_{max} + |v|_{max} + |w|_{max}}{\Delta x}$, $G_{cfl} = \frac{|g|}{\Delta x}$ and $S_{cfl} = \frac{\sigma\kappa}{min(\rho_{fluid^1}, ..., \rho_{fluid^n})\Delta x^2}$. In addition to the use of Eq. (13), the maximum time step size is limited to the intended duration of each frame on the desired animation's frame-rate. Additionally, Δt is limited to a minimum value, even tough when this threshold is reached, the simulation may be quantitatively imprecise, albeit plausible.

Finally, density values must be estimated at the center of the each face of every grid cell. This is done based on the technique presented by Kim et al. (2007) and Kang et al. (2000) where these values are given by an interpolation between the densities of the fluids in the two adjacent cells, resulting in a continuous density profile. For instance, in the case of two cells i and $i+1$, initially, it is computed the regional tuple $\langle r_{face}, \phi_{face} \rangle$ in the center of the face between them. Afterwards, it is applied the Eq. (6), using ϵ as half of the film thickness h, to calculate:

$$\rho_{i+1/2,j,k} = \rho_{out} + (\rho_{int} - \rho_{out})H(\phi_{face}), \tag{14}$$

where $\rho_{in} = \rho_{face}$ is the density of the fluid that contains the center of the face in question and ρ_{out} is density of the fluid in the region on the other side of the film.

3 REGION DETECTION ALGORITHM

As a means to know which regions exist at any given time, it is used a flood fill segmentation algorithm, once employed by Greenwood et al. (2004) to detect air pockets and adapted by Kim (2010) to detect regional configurations. While Kim performs regional fusion on a separate phase after region detection, here it is incorporated in the flood fill algorithm. Additionally, the algorithm is executed with one thread per cell, while flooding different fluids at the same time.

if c_{to} is not Fluid **then**
  $r^t(c_{to}) = r^t(c_{from})$
else if $r^{t-1}(c_{from}) = r^{t-1}(c_{to})$ **then**
  $r^t(c_{to}) = r^t(c_{from})$
  activate c_{to}
else if $fluidAt(c_{from}) = fluidAt(c_{to})$ **then**
  $l_{film} = 0.5 * (l_{r^t(c_{from})} + l_{r^{t-1}(c_{to})})$
  if ($l_{film} < l_{min}$ **Or** $\rho_{fluidAt(c_{from})} > \rho_{film}$) **then**
    $l_{r^t(c_{to})} = max(l_{r^t(c_{from})}, l_{r^{t-1}(c_{to})})$
    $r^t(c_{to}) = r^t(c_{from})$
    activate c_{to}

Figure 3: Procedure visit(c_{from}, c_{to}) that visits the cell c_{to} coming from the cell c_{from}.

The general idea is to always have at least an active cell for each type of fluid, from which the flood will be made, and to only stop when all cells have been visited. Hence, every time there is not an active cell for a given fluid, a random unvisited cell of the fluid is activated and it is created

a new region for it. Then, each active cell floods to adjacent unvisited cells of the same fluid, while testing film rupture conditions for cells that have different original regions. The visit procedure, by itself, can be seen on Fig. 3.

The detection algorithm works basically with two maps, one of the current regional configuration and one with the disposition of the different fluids. Examples of iterations of the algorithm can be seen at the Fig. 4. On the top images, a cell of each fluid has been activated at the start of the algorithm. Then, adjacent cells of the same respective material will be flooded. On the bottom images, a random cell has been activated for the green fluid, as it has run out of active cells, and a new region is created. Also, film rupture conditions are tested at the transitions $C_1 \rightarrow C_1'$ and $C_2 \rightarrow C_2'$.

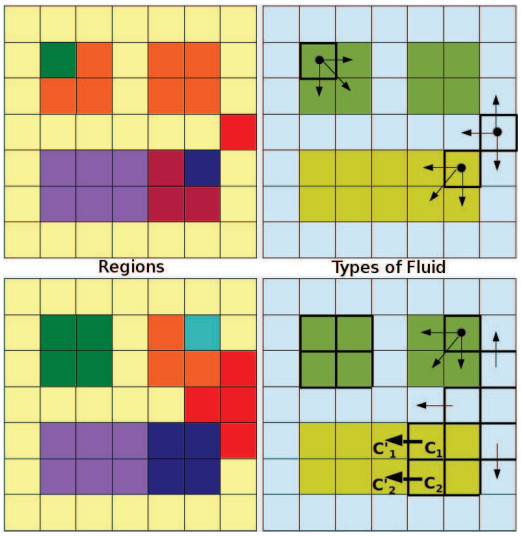

Figure 4: Two iterations of the proposed region detection algorithm.

Because of the need to track properties between successive iterations, the previous algorithm is, in practice, carried out in two phases that have their own steps of characteristics transfer. At the first one, a simple segmentation of the space is performed based on region and material dispositions. At the second, film ruptures are tested and regions are merged. This division into steps is performed because considering splits and merges in different phases considerably simplifies the tracking of region features.

4 SELFISH VOLUME CORRECTION

It is performed volume correction trough the evolution of the interfaces by an auxiliary velocity field in such a way that it is preserved the interfaces' direction of displacement. In contrast, the technique presented by Kim et al. (2007) corrects the volume omnidirectionally, which makes it seem that fluid masses are simply inflated or deflated. In the presented method, an additional advection step is performed for each time step t, using an auxiliary velocity field, in which values are non-zero only where relative to positions near an interface.

To create the auxiliary field, it is defined is components Vel^t_{face} - that may refer to a component u, v or w of the velocity - as those located on the center of the faces whose adjacent cells are in different regions. Then, regions r can be obtained at these component's positions. Their respective values can be defined so that each one fosters a variation of volume on its respective region r of:

$$(Vel^t_{face}\Delta t)\Delta x^2. \tag{15}$$

At this point, it is important to notice that each local correction tends to the individual error of a single region. So, as this method does not take in consideration the error of the surrounding regions, it is possible that, by correcting one region's volume, errors are introduced into the adjacent regions' volumes. Therefore, it would be necessary to have an optimal error correction method that takes into account the volume error of each region and creates auxiliary velocities that would minimize the global error on the system.

According to the selfish method, trough each auxiliary velocity component, the correction it performs on its region r is forced to be a fraction of the total correction that should be done on the interfaces of r. Indeed, its value is defined from the region's volume error and the ratio between $||\ \vec{u}^t \cdot \hat{N}^t\ ||$, estimated at the components position and the total sum of these values for all components of the same region:

$$||\ Vol^{target}_r - Vol^t_r\ ||\ \frac{||\ \vec{u}^t \cdot \hat{N}^t\ ||}{\sum_{face\in r} ||\ \vec{u}^t \cdot \hat{N}^t\ ||}. \tag{16}$$

Using this formulation, the smaller the angle between $\vec{u}^t$ and the normal $\hat{N}^t$, the greater will be the component's contribution to the volume's correction. This approach can be justified by the fact that only normal velocity components at a surface can change a region's volume.

Then, by equating Eq. (15) Eq. (16), $||\ Vel^t_{face}\ ||$ is defined as:

$$||\ Vel^t_{face}\ || = \frac{||\ Vol^{target}_r - Vol^t_r\ ||}{\Delta t\Delta x^2}\frac{||\ \vec{u}^t \cdot \hat{N}^t\ ||}{\sum_{face\in r} ||\ \vec{u}^t \cdot \hat{N}^t\ ||}. \tag{17}$$

Notwithstanding, the sign of Vel^t_{face} still needs to be defined. The idea behind this choice is to make the movement of the interface accelerate or decelerate in distinct portions of the interfaces in a way to assure the desired volume correction. To aid this definition, n_{face} is defined as the component perpendicular to the face in question of the normal vector $\hat{N}$, which is estimated on the center of the face and points inwards. Then, as a general rule, it is assigned:

$$sign(Vel^t_{face}) = sign(n_{face}(Vol^{target}_r - Vol^t_r)). \tag{18}$$

Besides the simple employ of the current error to correct the volume, it is also taken in consideration its history near interfaces. In fact, corrections are accumulated over time for each velocity component, from when it becomes near an interface. However, when a component moves away

from the interface, its accumulated correction factor is discarded. So the auxiliary velocity is defined as a weighted sum, whose weights are based on the paper by Kim et al. (2007):.

$$Vel^t_{face} = \frac{1}{\Delta x^2}\left\{ k_p \left[|| \, Vol^{target}_r - Vol^t_r \, || \, \frac{|| \, \vec{u}^t \cdot \hat{N}^t \, ||}{\sum_{face \in r} || \, \vec{u}^t \cdot \hat{N}^t \, ||} \right] + k_l \left[\sum_{\tau'=t_0^{face}}^{t} \left(|| \, Vol^{target}_{r^{\tau'}} - Vol^{\tau'}_{r^{\tau'}} \, || \, \frac{|| \, \vec{u}^{\tau} \cdot \hat{N}^{\tau'} \, || \, \Delta\tau'}{\sum_{face \in r^{\tau'}} || \, \vec{u}^{\tau} \cdot \hat{N}^{\tau'} \, ||} \right) \right] \right\} \tag{19}$$

where $k_p = \dfrac{1}{\Delta t}$, $k_l = \dfrac{k_p^2 \Delta t}{16} = \dfrac{1}{16\Delta t}$. Also, t_0^{face} indicates when a component Vel_{face} is estimated in a face whose adjacent cells are on different regions, and $r^{\tau'}$ is the region that contains the position of the Vel^t_{face} in the sub-step τ' of size $\Delta\tau'$ of the iteration τ.

5 BENEVOLENT VOLUME CORRECTION

Here it is introduce a method that determines the value of the components of the auxiliary velocity field so that, trough an additional advection step, the volume of each region is corrected while minimizing the overall error in the fluid system.

Firstly, it is defined an orientated graph in which the edge (r_1, r_2), with $r_1 < r_2$, exists if both regions are adjacent, and is oriented from r_1 to r_2. This graph defines the positive orientation of the flow of volume trough each interface, i.e., determines that a positive flow represents a transfer of volume from r_1 directly to r_2. This graph is stored as the matrix A of dimension $n_{regions} \times n_{interfaces}$, where $n_{regions}$ and $n_{interfaces}$ are respectively the current number of regions and interfaces.

Then, the following system is built:

$$A_{(n_{regions} \times n_{interfaces})} f_{(n_{interfaces} \times 1)} = \Delta Vol_{(n_{regions} \times 1)}, \tag{20}$$

where f is the flow of volume trough each interface and ΔV is the volumetric error of each region. Actually, for a region r, $\Delta Vol_r = V^{target}_r - V^t_r$.

Afterwards, it is specified an initial guess f^0 for the volume flow trough each interface in order to try to correct the underlying volume error. For this purpose, it is calculated the mean of the estimates from both sides of the interface in regards to the volume flow needed to correct the volume of each individual region. Again it is used a ratio involving the projection of the velocity onto the direction normal to the surface. For the interface between r_1 and r_2:

$$f^0_{r_1,r_2} = 0.5 \left[\frac{I_{r1,r2}}{I_{r_2}} (V^{target}_{r_2} - V^t_{r_2}) - \frac{I_{r_1,r_2}}{I_{r_1}} (V^{target}_{r_1} - V^t_{r1}) \right] \tag{21}$$

where $I_{r_1,r_2} = \sum_{f(r_1,r_2)} || \, \hat{N} \cdot \vec{u} \, || \, \Delta t \Delta x^2$ in which it is summed the $\hat{N} \cdot \vec{u}$ estimated at each center of a face $f(r_1, r_2)$ whose adjacent cells are one from r_1 and the other from r_2. Also, $I_{r_1} = \sum_n I_{r_1,r_n}$ where r_n is an adjacent region to r_1.

This initial solution only considers two regions and is not necessarily a solution of the Eq. (20) in the sense that it may not correct the volume of all fluid regions. To overcome this, the Eq. (20) is solved for the flow f so that it is as close to f^0 as possible. This is achieved trough the orthogonal projection of f^0 over the linear variety which constitutes the solution to the linear system.

To analysis further, one can observe that the subspace orthogonal to the linear variety $Af = \Delta V$ is the image of the operator A^T, $A^T\lambda$ with $\lambda \in \mathbb{R}^{n_{regions}}$. Thus, from the initial solution, $|| f - f^0 ||$ can be minimized by making $f = f^0 + A^T\lambda$. As $f \in Af = \Delta V$:

$$A(f^0 + A^T\lambda) = \Delta V \rightarrow AA^T\lambda = \Delta V - Af^0 \tag{22}$$

It can be observed, then, that the matrix AA^T is the laplacian matrix. So, actually, Equation 22 is a Poisson's equation $\nabla^2\lambda = \Delta V - Af^0$ that can be solved the usual way. For instance, using the conjugate gradient method. Thus, the equation is solved for λ, the flow f is calculated for each interface and, then, it is determined the value of each non-zero component on the auxiliary velocity field as:

$$Vel_{face} = \frac{|| \vec{u} \cdot \hat{N} ||}{\sum_{interface} || \vec{u} \cdot \hat{N} ||} \frac{f_{interface}}{\Delta t \Delta x}, \tag{23}$$

which will be used to perform an additional advection step.

6 INFLOWS AND OUTFLOWS

In order to use a volume control method when there are fluids flowing in or out of the computational domain, the target volume of the regions must be updated according inflows and outflows. Thus, a cell face at the boundaries of the computational domain is defined as an input(output) face, if its velocity $Vel_{input}(Vel_{output})$ points inwards(outwards).

Then, for each input face, it is added $(Vel_{input}\Delta t)\Delta x^2$ to the target volume of the region to which its cell belongs, equivalent to the volume that passes trough the face over a time interval Δt. Therefore the total volume added is $\sum_{inputs}(Vel_{input}\Delta t)\Delta x^2$.

Similarly, the total amount of fluid that exits the grid can be measured, but, in order to define the decrease of target volume of any region, some assumptions are made. To begin, it is possible that, for whatever reasons that allows volume losses, the total volume that exits the grid might be different from the one that enters. As all simulated fluids are incompressible, these two values must be the same. Therefore, the target volume removed from the region of each cell that has an output face is set as a function of the total inflow volume:

$$\frac{Vel_{output}}{\sum_{outputs} Vel_{output}} \sum_{inputs} (Vel_{input}\Delta t \Delta x^2). \tag{24}$$

7 PARTICLES

The volume of a particle p is evolved trough the equation:

$$r_p^{t+1} = r_p^t + \left(\frac{r_p^{target} - r_p^t}{t_{c_p}} \right), \tag{25}$$

where $t_{c_p} = 20$ defines the expected number of iterations needed so its volume is corrected. So, if a region shrinks and becomes a particle, the latter can regain the volume previously lost and even turn back to a region.

In relation to the overall target volume of the regions, one issue that needs attention is what happens to the target volume of the existing regions when a particle is created or when it is converted to a region. In the first case, the target volume of the region converted is distributed among the remaining regions according their fraction of the total target volume on the system. In the second case, it is removed some of the target volume of all regions that already existed, also based on volume fractions so the total volume of fluid in the grid stays the same. In addition, due to the differences between the particle and region representations, the target volume of the newly created region is defined as the volume occupied by the new region after the level set's reconstruction.

8 VISUALIZATION

For the final visualization, it is employed the raytracer Yafaray to render images from scenes in Blender composed of meshes imported from the main application. To create meshes from the implicit surface representation defined by the level set, the property of every interface where there is a change of medium must be discriminated. So, instead of creating a single mesh for the system using the marching cubes algorithm (Lorensen et al., 1998), one is created for each interface and their correct properties are assigned.

Besides, as it is considered a film between each phase, actually, each level set's interface is equivalent to two actual interfaces: one which separates the first medium from the film and one that dissociates the latter from the second medium. Thus, the film between two regions is created by displacing the level set's interface to within each region so that two surfaces are created. However, in the particles' cases, as they have no film, only the surface of the sphere is considered.

Therefore, for each region r_i, it is built its interfaces with its adjacent interfaces r_j employing the marching cubes algorithm, but shifting such surfaces by $\epsilon = 0.1\dfrac{\Delta x}{2}$ to the interior of r_i so that the resulting film has thickness $h = 0.1\Delta x$. This displacement is performed when the level set is evaluated at a point k of regional tuple $\langle r_k, \phi_k \rangle$. If $r_k = r_i$, then ϵ is subtracted from ϕ_k and, otherwise, it is summed. Due to the subtraction, the value of ϕ_k can become negative. If this happens, phi_k is set to zero to avoid inconsistencies on the interface's representation, even tough it makes its thickness lesser then the desired near k.

Additionally, meshes relative to the interfaces of a fluid region with a solid cell or the exterior of the simulation grid are created with the marching squares algorithm, since these are all bidimensional surfaces. Also, In the molds of the procedure described previously, in these situations, interfaces are also displaced to within each region.

These bidimensional meshes are created for one region at a time. Thus, for each face that separates a fluid cell from either a solid one or the exterior of the grid, each of its vertices is visited in a predefined order to obtain the vertices of the triangles to be created. During this path, a triangle vertex is created at each face vertex of the region in question and at the edges where, among the adjacent vertices, one is from the desired region and the other is from a different one. In this case, the new vertex position is at the zero interpolant position between the edge's vertices. Thus, the first three vertices created form a triangle and then, for each new couple of vertices, a new triangle is formed together with the last edge of the previous triangle. Examples of possible triangle configurations can be see on Fig. 5

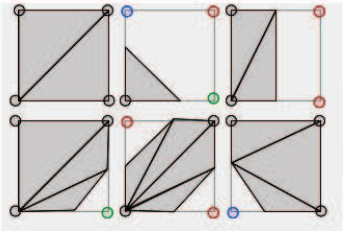

Figure 5: Example of triangulation of a chosen region on a cell's face employing the marching squares algorithm.

Thereafter, for each mesh created, it is computed its refraction index according to the adjacent materials in the way required by Yafaray, and its diffuse color is defined based on the inside region's fluid. As a ground rule, for an interface between mediums with indices of refractions (IOR) n_1 and n_2, and where the direction normal to the surface $\hat{N}$ points to the medium of n_1, its IOR is calculate by $(\text{IOR})_\Gamma = \dfrac{n_2}{n_1}$.

To effectively create an animated video, it is initially stored a text file with the intended frame rate, the center and the dimensions of the grid. Then, taking in consideration the defined frame rate, the application exports, for each frame, a Wavefront .obj file whose different objects are the currently existing interfaces' meshes. Additionally, a text file is filled with each interface's diffuse color and index of refraction. For future distinction, each file is named after its respective frame number and the distinct objects and properties are paired according to the order in which they are defined.

Afterwards, a script in the Python language is executed on the Blender environment to load each frame and render using Yafaray. Before this script is run, it is defined desired parameters such as lights, camera positions and the raytracer's parameters while taking advantage of Blender's graphical interface. Finally, all created images are automatically grouped into a movie with the intended frame rate.

9 RESULTS AND DISCUSSION

In this section, it is exposed some results of the methods described and some of their limitations, while possible future works are suggested. To better understand the results, consider that all

the computation and render were performed on a $22 \times 22 \times 22$ low resolution grid. Still, considerably good results were obtained.

9.1 Performances

All simulation and render were executed in a computer with the following specifications: Intel® Core™ i7-2630QM 2.00GHz processor, 8090MB of memory, video board GeForce GT 540M, running Ubuntu 12.04 LTS. The average time of execution of each simulation's iteration was 1.07433 seconds. Also, to simulate a frame of a animation at 30 frames per second, it took an average of 7.15506 seconds. Finally, for the final render, it was spent, in average, 31 minutes per frame.

9.2 Quality

As a measure of the quality of the techniques proposed, it was evaluated the evolution of the volume error in a system having fluid inflow and outflow. The chart illustrated in Fig. 6 shows the evolution of the volume error as a percentage of the current volume of each region. Note that the error oscillates but does not get bigger, as expected when not using a volume control method.

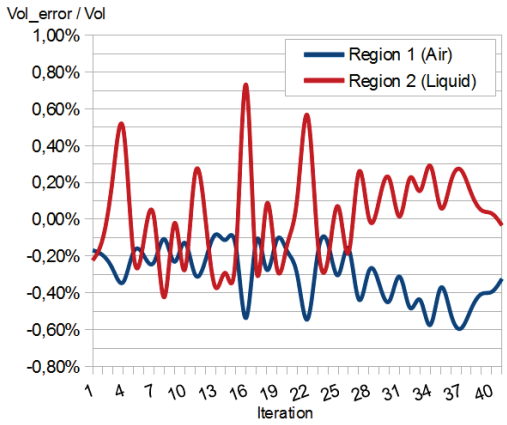

Figure 6: Evolution of the relative volume error at an inflow/outflow simulation.

9.3 Limitation and Future Works

In this work, it was imposed that there always exists a film of a predefined material between two phases. A natural sequence would be the consideration that a film might not exist between phases of different materials and that it could be composed of multiple fluids according to the surrounding phases. In addition, even if these films have minimal dimensions, thin phases cannot be represented with the same precision.

Also, it is suggested the use of a more robust technique to solve Poisson's equations, such as the preconditioned conjugate gradient, and the consideration of viscous fluids. Plus, as the

implementation using CUDA's technology was not the focus of this work, it is needed further analysis and optimization of key algorithms.

In terms of the robustness of the proposed volume control methods, it could be enhanced by also applying other techniques, such as the particle level set (PLS) (Enright et al., 2002) and the back and forth error compensation and correction (Dupont et al., 2003; Kim et al., 2005). Besides, in relation to the PLS, it could be used to create droplets from escaped particles, increasing the visual realism of the simulation.

Furthermore, the volume computation method employed is but an approximation of the real volume. In fact, there is no guarantee that, inside a cell, the sum of the volume of the different regions it contains is equal to the volume of the cell itself. It would be good, then, to have a way to precisely calculate the volume.

10 CONCLUSION

The regional level set approach was adopted to care for the challenge of evolving multiple interfaces in a multiphase system. Within this context, a modified flood fill segmentation algorithm was proposed to account for the merge and division of regions while executed in parallel for each fluid type.

Also, it was proposed two volume control methods that operate by performing additional advection steps to correct volume errors. The corrections of the first method are concerned mainly with the individual error of each region, while the second technique seeks to minimize the global error. The main advantage of these methods is that they perform volume corrections in the direction of the fluid's movement and do not simply inflate or deflate regions. Altogether, it was observed considerable conservation of each phase's original volume even on very low resolution grids.

In addition to the previous methods, particles were employed to replace small fluid regions that might have lost their original volume and their radii were adjusted over time so they could recover the target volume of their respective original regions. Moreover, it was defined how the target volume of each phase should vary due to the inflow and outflow of fluids in relation to the simulation grid.

Finally, in terms of the visualization, while triangle meshes were created from the main application, a generic solution for the final render of fluids was implemented using well-known tools like Blender, Yafaray and the Wavefront .obj standard. Hence, other applications could easily take advantage of it as long as they provide the required inputs.

REFERENCES

Brochu, T., Batty, C., & Bridson, R., 2010. Matching fluid simulation elements to surface geometry and topology. In *ACM Trans. Graph*. 29.4, 47:1–47:9.

Busaryev, O. et al., 2012. Animating bubble interactions in a liquid foam. In *ACM Trans. Graph*. 31.4, 63:1–63:8.

Cleary, P. W. et al., 2007. Bubbling and frothing liquids. In *ACM SIGGRAPH 2007 papers*.

Courant, R., Friedrichs, K., & Lewy H., 1928. Uber die partiellen differenzengleichungen der mathematischen physik. In *Mathematische Annalen* 100.1, pp. 32–74.

Dupont, T. F., & Liu, Y., 2003. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. In *J. Comput. Phys.* 190.1, pp. 311–324.

Enright, D., Marschner, S., & Fedkiw, R., 2002. Animation and rendering of complex water surfaces. In *proceedings of the 29th annual conference on computer graphics and interactive techniques*. SIGGRAPH '02, pp. 736–744.

Foster, N., & Fedkiw, R., 2001. Practical animation of liquids. In *proceedings of the 28th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '01, pp. 23–30.

Foster, N., & Metaxas, D., 1996. Realistic animation of liquids. In *proceedings of the conference on Graphics interface. GI '96. Canadian Information Processing Society*, pp. 204–212.

Greenwood, S. T.,& House, D. H., 2004. Better with bubbles: enhancing the visual realism of simulated fluid. In *proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '04. Eurographics Association, pp. 287–296.

Guendelman, E. et al., 2005. Coupling water and smoke to thin deformable and rigid shells. In *ACM Trans. Graph*. 24.3, pp. 973–981.

Harlow, F. H., & Welch, J. E., 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. In *physics of fluids* 8.12, pp. 2182–2189.

Hestenes, M. R., & Stiefel, E., 1952. Methods of conjugate gradients for solving linear systems. In *Journal of research of the National Bureau of Standards 49*, pp. 409–436.

Hong, J.-M., & Kim, C.-H., 2005. Discontinuous fluids. In *ACM Trans. Graph*. 24.3, pp. 915–920.

Hong, J.-M. et al., 2008. Bubbles alive. In *ACM SIGGRAPH 2008 papers*. SIGGRAPH '08, 48:1–48:4.

Kang, M., Fedkiw, R. P., & Liu, X.-D., 2000. A Boundary Condition Capturing Method for Multiphase Incompressible Flow. In *J. Sci. Comput.* 15.3, pp. 323–360.

Kim, B., 2010. Multi-phase fluid simulations using regional level sets. In *ACM Trans. Graph.* 29.6, 175:1–175:8.

Kim, B. et al., 2005. FlowFixer: using BFECC for fluid simulation. In *proceedings of the First Eurographics conference on Natural Phenomena*. NPH'05. Eurographics Association, pp. 51–56.

Kim, B. et al., 2007. Simulation of bubbles in foam with the volume control method. *In ACM Trans. Graph*. 26.3.

Liu, X.-D., Fedkiw, R. P.,& Kang, M., 2000. A boundary condition capturing method for Poisson's equation on irregular domains. In *J. Comput. Phys.* 160.1, pp. 151–178.

Lorensen, W. E. & Cline, H. E., 1998. Seminal graphics. In New York, NY, USA: ACM. Chap.

Marching cubes: a high resolution 3D surface construction algorithm, pp. 347–353.

Losasso, F. et al., 2006. Multiple interacting liquids. In *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06, pp. 812–819.

Losasso, F. et al., 2008. Two-Way Coupled SPH and Particle Level Set Fluid Simulation. In *IEEE Transactions on Visualization and Computer Graphics* 14.4, pp. 797–804.

Mihalef, V., 2007. *The marker level set method: applications to simulation of liquids*. PhD thesis. New Brunswick, NJ, USA: Rutgers University.

Mihalef, V., Metaxas, D. N., & Sussman, M., 2009. Simulation of two-phase flow with sub-scale droplet and bubble effects. In *Comput. Graph. Forum* 28.2, pp. 229–238.

Müller, M., Charypar, D., & Gross, M.., 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. SCA '03. Eurographics Association, pp. 154–159.

Osher, S., & Fedkiw, R., 2003. *Level set methods and dynamic implicit surfaces*. Springer Verlag.

Selle, A. et al., 2008. An Unconditionally Stable MacCormack Method. In *J. Sci. Comput*. 35.2-3, pp. 350–371.

Song, O.-Y., Shin, H., & Ko, H.-S., 2005. Stable but nondissipative water. In *ACM Trans. Graph*. 24.1, pp. 81–97.

Stam, J., 1999. Stable fluids. In *proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

Sussman, M. 2003. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. In *J. Comput. Phys.* 187.1, pp. 110–136.

Takahashi, T. et al., 2003. Realistic Animation of Fluid with Splash and Foam. In *Computer Graphics Forum* 22.3, pp. 391–400.

Vese, L. A., & Chan, T. F., 2002. A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model. In *Int. J. Comput. Vision* 50.3, pp. 271–293.

Wojtan, C. et al., 2010. Physics-inspired topology changes for thin fluid features. In *ACM SIGGRAPH 2010 papers*. SIGGRAPH '10, 50:1–50:8.

Wojtan, C. et al., 2009. Deforming meshes that split and merge. In *ACM Trans. Graph*. 28.3, 76:1–76:10.

Zheng, W., Yong J.-H., & Paul J.-C., 2009. Simulation of bubbles. In *Graph. Models* 71.6, pp. 229–239.