Multiphase Flow of Incompressible Fluids Employing Regional Level Set and Volume Control

Túlio Ligneul Santos, Antonio Alberto Fernandes de Oliveira, Paulo Roma Cavalcanti, Claudio Esperança
Programa de Engenharia de Sistemas e Computação
COPPE / Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
http://www.lcg.ufrj.br

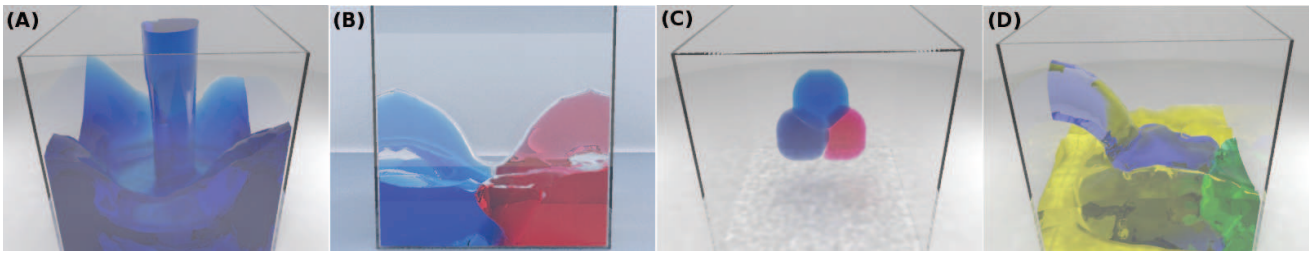

Fig. 1. Resulting simulations of at least two different interacting fluids.

***Abstract*—In this work, the regional level set is employed to track interfaces in multiphase fluid simulations, while considering the behavior of the film between distinct phases. Also, properties of each phase are tracked as they merge or split, and an algorithm to apply these operations is proposed. Furthermore, it is introduced a method to handle undesired volume changes that tries to minimize the total volume error by making local corrections trough an additional advection step. Moreover, it is put forward an accurate volume calculation technique that takes advantage of the phase distribution determined by the chosen interpolation method. Finally, particles are employed to avoid volumes losses due to discretization's problems.**

***Keywords*-multiphase simulation; volume control; level set;**

I. Introduction

Standard examples of multiphase fluid simulations encompass systems where oil floats over water, soap bubbles move on a liquid surface, or two adjacent air bubbles are in contact, separated by a liquid film. In the latter, the film thickness indicates how easily the film can be ruptured and the two adjacent phases merged. Based on this, thickness values are evolved over time and film rupture conditions are tested.

In the multiphase context, it is particularly important to avoid the volume loss induced by discretization errors and the way fluid interfaces are evolved. In fact, if unattended, the accumulated volume loss can even bring about the removal of a whole phase. Thus, it is advisable to have a way to enforce that each phase's volume is equal to its original value plus the in-and-outflows contributions over time.

Both pure Lagrangian and Eulerian single phase models, besides several hybrid versions, have been extended to the multiphase context. In hybrid models, particles can be used to track interfaces in an Eulerian framework [1], but can also interfere with the simulation [2]. Also, Guendelman [3] uses particles that have escaped from the air into the liquid medium, to create air bubbles and Mihalef [4] creates them as a by-product of the marker level set evolution method. These techniques, however, are not specifically addressed for volume preservation, though they favor it by producing more detailed surfaces. Their capacity to prevent, for example, the stop in the evolution of thin fluid streams, which is related to volume losses, depends on having an adequate parametrization of the model and a sufficient density of particles.

In this work, particles are used to prevent volume losses of phases that became too small. Yet, for larger volumes, an explicit correction scheme is applied. In these cases, phase volumes are controlled under the regional level set (*RLS*) context [5], which extends to the multiflow context, the good properties of the standard level set. Thus, volume control can be achieved without the use of volume-of-fluid based alternatives [6]. Despite directly addressing the control of the volume, these methods have some known drawbacks with respect to the interface evolution. Additionally, to prove that there is no need to increase the resolution to correct minor volume problems, this work specially focuses on low scale mesh examples.

Contributions: To detect, merge and split regions, it is introduced a flood fill segmentation algorithm that is executed in parallel for each material. Plus, a more precise volume computation method is proposed, which is based on the phase distribution determined by the *RLS* within a cell. Also, a volume correction method is formulated to minimize the total volume error trough local volume corrections determined by an additional level set advection. Moreover, the expected(target)

1530-1834/13 $26.00 © 2013 IEEE
DOI 10.1109/SIBGRAPI.2013.16

volume of a phase is updated considering inflow/outflow amounts and each particle's volume is corrected based on the volume of the phase that generated it.

A. Related work

To support multiple materials, the classic approach to track interfaces, the level set method [7], must be extended as it can only represent the interface between two materials. Accordingly, the techniques introduced by Losasso [8] and Vese [9] can evolve the interfaces of multiple fluids, despite the complexity increase when the number of materials grows.

The former employs a level set per material, always merging adjacent phases of the same fluid. Also, as the evolution of each level set in done independently, it generates contradictions in the interface's representation. In the latter, sign combinations of n level sets represent up to 2^n phases. Still, this method presents undesired artifacts, specially where two regions intersect.

To evolve film-like thin surfaces, Wojtan [10] developed hybrids of grid and explicit surfaces, albeit its complex topology change process. Also, Brochu [11] disturbed samples to capture thin features. Both methods were only used in one or two phase flows, while employing a traditional level set.

Robust advection models, such as back and forth error compensation and correction [12], modified MacCormack [13], and the use of cubic interpolated polynomials [14] can be used to reduce volume errors. Other possible solutions are hybrid approaches, like the particle level set [1], and volume of fluid methods [6].

Kim [15] proposed a volume control method that can be employed together with the previous techniques and uses the *RLS*. Its objective is to correct the volume of air bubbles by changing the field of velocity divergents. As a result, regions are inflated or deflated.

B. Technique overview

At first, pressure and velocity values sampled in a regular MAC grid [16] are updated through the solution of a variable coefficient Poisson system [17]. In this setup, surface tension forces are provided by the ghost fluid method [18]. Then, the *RLS* evolves fluid interfaces, eventually determining topological changes and turning regions into particles and vice-versa. All these aspects are described in section II, while section III focuses on the detection of new regional distributions. Afterwards, region volumes are evaluated - Section IV explains how - and an additional *RLS* advection step is executed to correct possible volume errors - Section V accounts for that point. Finally, for visualization, region interface meshes are exported to Blender [19] to be rendered using Yafaray [20] - See section VI. Section VII is dedicated to results and future works while section VIII reports the conclusions of the work.

II. Technical background

In this section it is detailed known techniques employed to perform a multiphase fluid simulation.

A. Fluid dynamics

A classical Navier-Stokes system of equations for inviscid fluids is solved through the widely known Chorin's method. Firstly, external forces are applied to the velocity field, which is, then, advected through a Runge-Kutta scheme of 2nd or higher order. Lastly, a Poisson's system is solved by the conjugate gradient method [21] to make the velocities divergent-free.

However, in the case of multiphase simulations, the coefficients of this system become variable since the density is no longer constant. In this case, a simplified version of Liu's method [22] is used as density is assumed continuous across interfaces. Also, surface tension forces are applied through the ghost fluid method [18]. Then, for each cell C, the Poisson's system is discretized as:

$$\sum_{C' \in N_C} \frac{p_{C'} - p_C}{\rho_{CC'}} = \frac{\Delta x^2}{\Delta t} \nabla \cdot \vec{u}_C + \sum_{C' \in N_C} \frac{J_{CC'}}{\rho_{CC'}}, \quad (1)$$

where Δx is the cell edge length. N_C is the set of cells adjacent to C, while CC' refers to the face between C and C'. $J_{CC'} = \sigma \kappa_{CC'}$ is the jump of pressure across the face CC' when its adjacent cells are in different phases. σ is the surface tension coefficient between the materials of these two phases and $\kappa_{CC'}$ is the interface's curvature at CC', which is obtained by using the operator $\nabla \cdot \left(\frac{\nabla \phi}{|| \nabla \phi ||} \right)$.

In the above equation, the term $\frac{J_{CC'}}{\rho_{CC'}}$ accounts for the surface tension force at CC' in accordance with the ghost fluid method. This approach extrapolates pressure profiles, from each side of an interface to the other, so that the pressure derivative becomes continuous and differentiable at the interface, even though the pressure itself is not. The sign of $J_{CC'}$ is $+$ or $-$ depending on whether C is, respectively, on the lower or higher side of the jump in the pressure graph.

The density value at the center of the face CC' is estimated based on the technique presented by Kang [17]. It is obtained by interpolating the densities of the regions of C and C', so that there is a continuous density transition between these cells. Firstly, the regional tuple $\langle r_{CC'}, \phi_{CC'} \rangle$ is computed at the center of the face. Afterwards, a continuous Heaviside function approximation H - like the one expressed in (12), is applied to $\phi_{CC'}$. In this case, the width of the interval where H is non-constant is equal to the film thickness at CC'. Then, $\rho_{CC'}$ is computed by:

$$\rho_{CC'} = \rho_{C_{out}} + (\rho_{C_{in}} - \rho_{C_{out}}) H(\phi_{CC'}), \quad (2)$$

where C_{in} and C_{out} are, between the cells C and C', respectively, the one in the region containing the center of CC' and the other cell.

To ensure the convergence of the numerical simulation, a time step is computed for each iteration following Kang's approach [17]. It enforces a CFL condition [23] that considers the joint influence of external forces, advection and surface tension in the velocities' updates.

B. Regional level set

The regional function is defined as the tuple:

$$f_R(\vec{x}) = \langle\, r(\vec{x}), \phi(\vec{x})\,\rangle\,, \tag{3}$$

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 on an interface. Zheng [5] introduced a series of operators to be applied to regional tuples. Those used in a linear interpolations are:

- Sum:

$$\langle r_1, \phi_1\rangle + \langle r_2, \phi_2\rangle = \begin{cases} \langle r_1, \phi_1 + \phi_2\rangle\,, & \text{if } r_1 = r_2 \\ \langle r_1, \phi_1 - \phi_2\rangle\,, & \text{if } \phi_1 \geq \phi_2 \\ \langle r_2, \phi_2 - \phi_1\rangle\,, & \text{if } \phi_1 < \phi_2 \end{cases} \tag{4}$$

- Multiplication by a scalar:

$$\alpha\,\langle r_1, \phi_1\rangle = \langle r_1, \alpha\phi_1\rangle\,, \alpha \in \mathbb{R}^+ \tag{5}$$

Nevertheless, it is possible that some of the properties of the standard math operations are not satisfied by the above definitions. For instance, the regional sum operator is not commutative. Thus, a trilinear interpolation (*TLI*), performed to obtain a regional tuple at any position in a cell, becomes inconsistent as the result depends on which axis is interpolated first. To avoid this, Kim [24] summed a sequence of regional tuples at once, while maintaining the addition's commutative property.

Regarding a *TLI*, for a point $\vec{p}$ with cell coordinates (x, y, z) in a unit-length cell C_{ijk}, it can be expressed as:

$$\sum_{l=0}^{1}\sum_{m=0}^{1}\sum_{n=0}^{1} |1-l-x||1-m-y||1-n-z| < r, \phi >_{i+l,j+m,k+n} \tag{6}$$

Then, defining:

$$\phi^*_{l,m,n}(x, y, z) = |1-l-x||1-m-y||1-n-z|\phi_{i+l,j+m,k+n}, \tag{7}$$

to calculate the Equation (6) with the improved sum operator, at first, the partial sum of ϕ^* values per region is computed:

$$\Psi_q = \sum_{\forall r_{(i+l,j+m,k+n)} = r_q} \phi^*_{l,m,n}(x, y, z) \tag{8}$$

Then, the two biggest values of Ψ, $\Psi_1 \geq \Psi_2$, are taken, with r_1 being the region relative to Ψ_1. The final result is given by $TLI(p) = \langle r_1, \Psi_1 - \Psi_2\rangle$.

Altogether, besides guaranteeing a mathematical property, this approach leads to a more regular segmentation of the space, as seen in Figure 2.

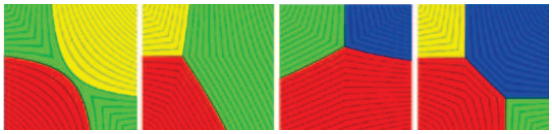

Fig. 2. Bilinear interpolation between four regional tuples using the operator proposed by Kim [24].

To obtain an initial regional distribution, for each cell C, a label identifying the fluid material at C is attributed to r_C. Also, for every cell adjacent to another containing a different fluid, let $\phi = \Delta x/2$. For the other cells, ϕ is obtained by a fast-marching propagation algorithm. This algorithm is also employed in the re-calculation process, executed after every level set advection, to make ϕ recover the property of being a distance function.

The re-calculation process starts by setting ϕ_{C_B},for every cell C_B immediately near an interface. It receives a value that estimates the minimal distance from the center of C_B to another region. There are different alternatives for that estimation. In the most employed one, only points on segments between cell centers are considered.

Next, the values of the cells near the interface are extrapolated to those farther from them. This task is executed in parallel with one thread per each cell C. It is checked whether, for an adjacent cell C', $\phi_{C'}$ is not null. If this is true, an estimation of the distance from the center of C to a border is performed based on the data relative to C'. Then, the updated value of ϕ_C is the minimum between the just computed estimate and its previous value. Thus, in $n-1$ iterations, cells distant up to $n\Delta x$ from a border will have their values defined.

C. Film handling

An essential aspect of multiphase simulations is the film between two adjacent regions. If these regions contain the same material, the presence of a film is mandatory as, otherwise, the two regions would have merged. If, instead, they are of different fluids, a film can exist or not and its destruction will not determine topological changes. In this work, adjacent regions are always interfaced by a film that is assumed to be composed of a predefined material, commonly water or a gas.

Film dynamics can be extremely complex, although acceptable results can be produced by simplified low-order methods that only consider some of the involved factors. Here, Zheng's approach [5] for realistic simulation of bubbles is applied to a more general context. By that approach, "region thicknesses" (*rt*) are evolved and film thicknesses are obtained by averaging the *rt* of adjacent regions. If a region is not created by a topological operation, its initial *rt* has a pre-established value l_0, here set to $0.1\Delta x$. This value is also the film thickness used for the visualization, since small variations would not make much visual impact. *rt* values are updated at every iteration through the equation below, where the two last terms refer, by order, to surface deformation and drainage:

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

where rt_r^t and A_r^t are, respectively, the *rt* and region border area of region r at the t-th time step, t_{drain} is the estimated maximum time for a *rt* to be reduced to l_{min}, the minimum thickness a film can have without being ruptured.

Films are ruptured not only if their thicknesses are less than l_{min}, but also when a film's density is smaller than those of

the adjacent regions. This occurs, for instance, when a falling water drop reaches the surface of a water body. If there is an indication that the film between two adjacent regions of the same material should be severed, the regions are merged into a new region, whose *rt* is the maximum between the thicknesses of the original ones.

D. Topological changes

Due to the possible occurrence of topological changes, a region neighbor graph (*RG*), which uniquely identifies regions and interfaces, is rebuilt at every step. It is computed by checking the pairs of adjacent cells that are in different regions. For a mild number of regions, the incidence matrix of the *RG* is found and its lower triangular part is stored in a boolean array. However, for a greater number of regions, a more compact model where adjacencies are listed must be used.

To identify when merge or split operations have happened and track properties accordingly, each region of the previous iteration's *RG* is mapped into those of the new *RG* that were originated from it. This mapping is obtained by examining the change of each cell's region label. A previous region with multiple correspondences identifies a split, while a region without a match is turned into a particle so that it is not lost. Also, merge situations are identified when a region at the new *RG* is inversely mapped into more than one region in the previous *RG*.

With the mapping, region features can be tracked from one iteration to the next as follows:

I) If a region R is created as a result of the split of a region S, then the target volume of R is chosen so that the ratio between the volumes of R and S is equal to the ratio θ between their target volumes. For simplicity, rt_R can be made equal to l_0 or rt_S. Alternatively, it can be given by a combination of these values, such as $l_0+\theta(rt_S-l_0)$, so that larger components tend to burst first.

II) When regions merge, the target volume of the resulting region is the sum of those of the merged regions, while the new *rt* is the maximum *rt* between those of the regions involved.

III) Particles that replace regions inherit their characteristics.

E. Particles

Spherical particles can be used to preserve volumes, besides conferring more realistic physical and visual features on the simulation. Here, the focus is to substitute small fluid components that would, otherwise, be removed from the simulation due to the adopted space discretization. A particle itself is defined by its radius, center position, type of fluid and velocity.

To move a particle p, its velocity and position are integrated over time using the forward euler scheme. Also, following the works of Kim [24] and Cleary [25], the effect of gravity is applied trough the force:

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

where V_p and ρ_p are, respectively, the volume and density of p, and $\vec{g} = 9.81\ m/s^2\ (-\hat{j})$. Besides, in accordance with Busaryev's research [26], It is applied the drag force:

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

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, particles are created from regions when they reach a minimum volume $V_p < V_{min}$ to avoid regions becoming too small and disappearing. However, if some region does so, i.e., if a region was removed during the advection process, a particle is created to replace it.

When a particle is created, it is positioned at 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. Its radius is determined so that it has the same volume as its original region. Also, to allow corrections of the particles' volumes, a target radius is defined for each particle so that their target volumes would be the same as those of their respective original regions. A region converted into a particle is marked as inactive to no longer be considered, for example, when testing film rupture conditions. 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. The first is when a particle contacts a region of its same material. In this case, the particle will be united to the region. The second situation is when the particle has a volume $V_p >= 1.5V_{min}$. Note that there is a discrepancy of $0.5V_{min}$ in relation to the minimum volume that a region can have. This is done to avoid successive changes between representations.

Finally, to convert a particle into a region, the *RLS* is rebuilt near it and the materials of the affected cells are updated. 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.

III. 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 [27] to detect air pockets and adapted by Kim [24] 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.

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 Figure 3.

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}

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

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 Figure 4. On the first image, 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 second image, a random cell has been activated for the green fluid, as it has run out of active cells. Also, film rupture conditions are tested at the transitions $C_1 \rightarrow C_1'$ and $C_2 \rightarrow C_2'$.

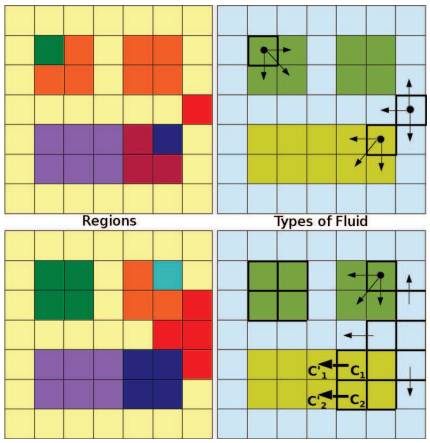

Fig. 4. Two iterations of the proposed region detection algorithm.

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. Considering splits and merges in different phases considerably simplifies the tracking of region features.

IV. Volume and Area Calculation

To calculate the volume and area of each fluid region, Kim [15] integrates a given function using, for each cell, 8 Gauss quadrature points. In the case of the volume, the Heaviside function below is computed at these points:

$$H(\Phi) = \begin{cases} 0, & \text{if } \Phi \leq -\epsilon \\ 1, & \text{if } \Phi \geq \epsilon \\ 0.5 + 0.75\frac{\Phi}{\epsilon} - 0.25\frac{\Phi^3}{\epsilon^3}, & \text{if } \mid \Phi \mid < \epsilon \end{cases}, \quad (12)$$

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 whose volume is wanted and, otherwise, is equal to its negative. Analogously, the area of the surface of a fluid region is calculated using the derivative of the Heaviside function.

The volume error produced by the above formulation has been proved to be $O(\Delta x^3)$. Nevertheless, there could be sequences of cells whose contribution to the total volume of the regions covering them is negligible when calculated by that method. Hence, it is quite inappropriate to employ it for some low scale instances.

In order to guarantee that the sum of the volume contributions computed at a cell is equal to volume of the cell itself, a possible solution would be finding, for every region r_k crossing a cell C, the volume of the set $S_k = \{p \in C |$ the r-coordinate of $TLI(p) = r_k\}$. Since that is considerably complicated, however, the calculus of TLI is reformulated by changing the definition of Ψ_q given in Equation (8) to $\Psi(k) = \max_{r_{i+l,j+m,k+n}=r_q} \phi^*_{l,m,n}(x, y, z)$. This change determines that any segment parallel to an axis crosses at most two different sets S_k. This makes it easier to find the volume of the regions crossing a cell from the areas occupied by these regions on two parallel faces, which can be accomplished as follows:

I) In C, find the intersection of each region crossing C, with two parallel faces of it, F_i, $i = 1, 2$. The boundary of each intersection consists of linear segments that separates the regions of two adjacent vertices and a rational curve separating the regions of the pair of opposite vertices whose product of ϕ-coordinates is larger. The diagonal defined by these vertices is called the major diagonal of the face. The major cell diagonal is defined similarly, as the one among the 4 cell diagonals, where the product specified above is the largest one.

II) Project one of these faces onto the other. The intersection of the partitions obtained in I form eight or nine components, each one associated to a pair of vertices laying on a different F_i. In four of them, these vertices are adjacent. Other four belong to the major diagonal of the faces orthogonal to F_i, with $i = 1, 2$. Finally, the central component, if it exists, is related to the vertices of the major cell diagonal.

III) The whole cylinder bounded by copies X_1 and X_2 of a component X found in II, one on each F_i, is immersed in the regions of the pair(v_1, v_2) of vertices associated to X. The volume of each of these regions within the cylinder can be obtained by multiplying the area of X by a factor of the form $\phi_1^*/(\phi_1^* + \phi_2^*)$. Here, ϕ_i^* - with $i = 1, 2$ - is an average of the function ϕ^* defined in Equation (7), considering only points in X_i related to vertex v_i.

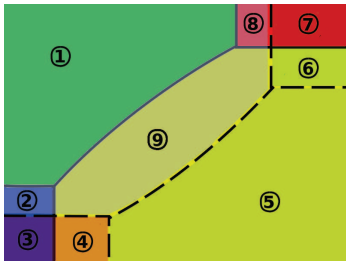

Fig. 5. Intersection of partitions of two parallel faces projected on a plane, and its 9 components.

Adding these volumes to their respective region volume accumulator, for all the components, the sum of their volume contributions is equal to the volume of the cell C itself. Besides, despite the previous long description, it can be proved that the process can be implemented with considerably less operations than the known quadrature points approach.

V. Volume Correction

Volume correction is performed trough an additional level set advection step using a corrector velocity field in a way that preserves the direction of the border's local motion. In contrast, Kim [15] corrects the volume omnidirectionally, i.e., fluid masses only inflate or deflate. This can be proper for some bubble evolution cases, but not to sustain the motion of a tiny fluid fillet where the border's motion must be constrained to its extremity.

A correction procedure is run at every iteration. Let F be a face between a cell in the region R and one in a different region R'. A component Vel_F^R of the corrector field that is orthogonal to F fosters a variation on the volume of R of magnitude:

$$Vel_R^F \Delta t(\Delta x^2), \tag{13}$$

where Vel_R^F will be valued so that the correction it determines is proportional to both the velocity at F and R's volume error. Greater corrections will, then, be made where the interface moves faster while the border is kept unchanged where the level set velocity is parallel to it. This has proved to be adequate for the contexts where volume corrections are usually most necessary, such as thin flat regions, tiny fillets and drops, besides groups of adjacent bubbles.

The volume correction performed locally at F during a Δt interval is defined as:

$$\Delta Vol = \mid\mid Vol_R^{target} - Vol_R^{current} \mid\mid \frac{\mid\mid \vec{u}_F \cdot \hat{N}_F \mid\mid}{\sum_{f \in R-border} \mid\mid \vec{u}_f \cdot \hat{N}_f \mid\mid}, \tag{14}$$

where $\hat{N}_F$ is the border normal at the center of F and $\vec{u}_F$ is the velocity at that point. Both are not necessarily orthogonal to F.

Matching Equations (13) and (14), the magnitude of Vel_R^F can be extracted. Its sign is chosen so that Vel_F^t points inwards or outwards the region R, according to if the target value is, respectively, lower or greater than the current one. This assures that no border face will contribute to the increase of R's error. In general, the rule for the sign of a velocity in the corrector field is:

$$sign(Vel_R^F) = sign(n_F(Vol_R^{target} - Vol_R^{current})), \tag{15}$$

where n_F is the component of N_F orthogonal to F.

However, the above process only corrects the volume error of a single region R, without considering the surrounding ones, whose volumes will also be affected by the correction applied to R. Thereafter, the treatment of a region can spoil the correction made in another. Thus, applying the scheme to the current regions sequentially, do not eliminate all errors, though they tend to be mostly reduced over time.

To smooth the convergence process, the calculus of Vel_R^F is based not only on the current iteration's data, but also on previous iterations' values, provided that F already belonged to the R-$border$ at those iterations. Defining Γ as the sequence of steps with this property, the magnitude of Vel_R^F can be expressed as:

$$\mid\mid Vel_R^F \mid\mid = \frac{1}{\Delta x^2 \Delta t} \left\{ \Delta Vol + \frac{1}{16} \sum_{\tau \in \Gamma} \Delta Vol^{\tau} \Delta \tau \right\}, \tag{16}$$

where ΔVol^{τ} is ΔVol at iteration τ. whose time step is $\Delta \tau$.

Towards a more global approach, it is taken the neighborhood graph (NG) of the set of regions, and its edges are oriented in the direction of the regions with greater labels. Then, let A be the incidence matrix of the (NG), whose dimension is the number n_R of regions $\times$ the number n_I of different interfaces. Consider, now, the following system:

$$A_{(n_R \times N_I)} f_{(N_I \times 1)} = Vol_{error(n_R \times 1)}, \tag{17}$$

where f, and Vol_{error} are the vectors whose components are, respectively, the flow of volume through every interface during Δt and the individual volume error of each region.

Afterwards, an initial guess f_0 is computed. Its component relative to an interface between r_1 and r_2 is given by:

$$f_0(r_1, r_2) = 0.5 \left[\frac{I_{r1,r2}}{I_{r_2}} Vol_{error_{r_2}} - \frac{I_{r_1,r_2}}{I_{r_1}} Vol_{error_{r_1}} \right], \tag{18}$$

where $I_{r_1,r_2} = \sum_{f \in \Omega_{1,2}} \mid\mid \hat{N} \cdot \vec{u} \mid\mid \Delta t \Delta x^2$. $\hat{N} \cdot \vec{u}$ is estimated at the center of each face $f \in \Omega_{1,2}$, the set of faces separating r_1 and r_2. Also, $I_{r_i} = \sum_n I_{r_i,r_n}$ where r_n is a region adjacent to r_i, with $i = 1, 2$.

In general, f_0 is not a solution of (17) since its components only take into account the errors of two adjacent regions. However, to avoid that unnecessarily large corrections are obtained, it is found the solution f closest to f_0. This result is given by the orthogonal projection of f_0 onto the linear variety which constitutes the solutions of the linear system.

It can be observed that the subspace orthogonal to the linear variety $Af = Vol_{error}$ is the image of the operator A^T, the set $\{A^T\lambda, \lambda \in \mathbb{R}^{n_R}\}$. Thus, $\mid\mid f - f_0 \mid\mid$ can be minimized by making $f = f_0 + A^T\lambda$. Then, as f solves (17):

$$A(f_0 + A^T\lambda) = Vol_{error} \rightarrow AA^T\lambda = Vol_{error} - Af_0 \tag{19}$$

Since A is an incidence matrix, AA^T is a laplacian matrix. Hence, (19) is a Poisson system that can be solved the usual way. For instance, using the conjugate gradient method. Thus, the system is solved for λ and the flow f is calculated for each interface to determine values of the corrector field. Explicitly, for a face F belonging to the interface I:

$$Vel_R^F = \frac{\mid \vec{u}_F \cdot \hat{N}_F \mid}{\sum_{G \in I} \mid \vec{u}_G \cdot \hat{N}_G \mid} \frac{f_I}{\Delta t \Delta x}, \tag{20}$$

which will be used to perform an additional level set advection step.

In turn, a particle p has its volume corrected trough the update of its radius so that it can regain the previously lost volume and even change back to a region.

$$r_p^{t+1} = r_p^t + \left(\frac{r_p^{t_0} - r_p^t}{t_{c_p}} \right), \tag{21}$$

where $t_{c_p} = 20$ determines a volume correction rate.

As a particle is created, in order to maintain the overall target volume of the regions, the volume of its original region is distributed among the other regions according to their fraction of the total target volume on the system. Analogously, when a particle is converted back to a region, part of the target volume of all regions is removed. Also, in this case, due to differences between particle and region representations, the target volume of the new region is defined as its volume after the level set reconstruction.

A final remark, the value obtained from Equation (13) is used to increment or decrement the target volume of regions as a result of fluid inflows and outflows.

VI. Visualization

Yafaray is used to render the evolution of the fluid flow. For each rendered frame, the marching cubes algorithm [28] is used the create meshes that represent the interfaces at the time of a frame. In fact, for each interface, two meshes are built to characterize region-film-region transitions. In the case of a particle, however, there is only a single mesh representing a sphere.

The surface border of a region r_i is obtained by slightly moving its representation produced by the marching cubes, towards the interior of r_i. This accounts for the presence of a film between r_i and adjacent regions r_j and makes the interface between them have a representation for each of these regions. In fact, this displacement is performed when the regional level set is evaluated. Let k be a point where the regional tuple is $\langle r_k, \phi_k \rangle$. If $r_k = r_i$, then an ϵ is subtracted from ϕ_k, or else, summed to it. If the value of ϕ_k becomes negative, ϕ_k is set to zero in order to avoid inconsistencies on the interface's representation, even tough it makes its thickness lesser than the expected.

For interfaces between fluid regions and non-fluid cells, or the exterior of the grid, the marching squares algorithm is employed. Thus, for each face F in a region r, that separates a fluid cell from a non-fluid cell, or from the exterior of the grid, all of its vertices are visited in a predefined order. Besides the points on the F-edges at the border of r, the F vertices in r are also introduced in the mesh representing the r-border.

Afterwards, each interface's diffuse color is defined and a refraction index is calculated according to the type of its adjacent materials, as required by the raytracer. Then, a file is saved with parameters such as the frame rate and, for each frame, all interfaces in a Wavefront (.obj) file. Colors and refraction indices are saved on a separate file. Finally, Blender is used to configure the scene and rendering parameters, and a Python script is executed to load and render each frame, which are grouped into a movie.

VII. 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.

A. Performances

The implementation was done using the Compute Unified Device Architecture (CUDA) [29] technology to accelerate the simulation. All simulation and rendering were executed in a computer running Ubuntu 12.04 LTS with the following specifications: Intel® Core™ i7-2630QM 2.00GHz processor, 8GB of memory, video card GeForce GT 540M. The average times spent on each iteration can be seen on Table I. Lastly, for the final render, it was spent, on average, 31 minutes per frame.

TABLE I
Average time spent per iteration of the cases in Figure 1.

Case	A	B	C	D	Mean
Average Time (s)	1.01	0.94	0.80	1.00	1.01

B. 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 Figure 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.

C. Limitation and Future Works

When applied to the multi-region context, the standard marching cubes (*MC*) approach creates holes within cells or inconsistent topologies. Thus, the implementation of an artifact-free *MC* that approximates the region distribution established by the RLS should be considered in a future work.

Also, it is suggested the use of a more robust technique to solve Poisson's equations, such as a preconditioned conjugate gradient for the variable coeficients case, and the consideration of viscous fluids. Furthermore, as the CUDA technology was

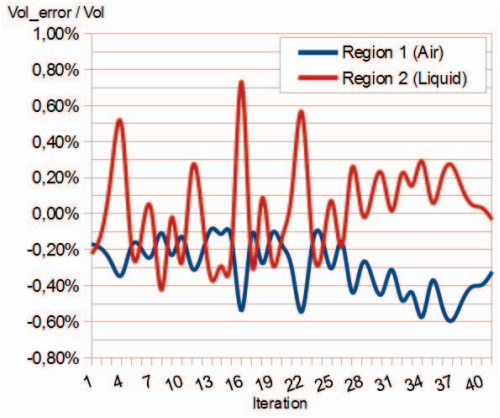

Fig. 6. Evolution of the relative volume error at an inflow/outflow simulation.

only employed to speed up the application, it can be performed further analysis and optimization of key algorithms.

Finally, the robustness of the proposed volume control method can be enhanced by combining it with other techniques, such as the particle level set (*PLS*) [1], and the back and forth error compensation and correction [12]. Also, the *PLS* can be used to create droplets from escaped particles, increasing the realism of the simulation.

VIII. Conclusion

In this work, the regional level set was employed to evolve multiple interfaces in a multiphase flow simulation together with a particle system, which is used to avoid the loss of small regions. In addition, the flood fill segmentation algorithm was modified to account for the merge and division of regions, and it was developed a more precise volume computation method.

Besides, it was proposed a volume correction method that performs corrections minimizing the global volume error on the system. Plus, it takes advantage of the error history while fostering the movement of the fluid. Also, region's target volumes were updated according to fluid inflows and outflows. Altogether, a considerable conservation of each phase's volume was observed even on a very low resolution grid.

Lastly, well-known tools were used to implement a generic solution to the final render that can be easily adapted for other applications.

References

[1] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '02. New York, NY, USA: ACM, 2002, pp. 736–744.

[2] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled sph and particle level set fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797–804, Jul. 2008.

[3] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw, "Coupling water and smoke to thin deformable and rigid shells," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 973–981, Jul. 2005.

[4] V. Mihalef, "The marker level set method: applications to simulation of liquids," Ph.D. dissertation, New Brunswick, NJ, USA, 2007, aAI3319673.

[5] W. Zheng, J.-H. Yong, and J.-C. Paul, "Simulation of bubbles," *Graph. Models*, vol. 71, no. 6, pp. 229–239, Nov. 2009.

[6] M. Sussman, "A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles," *J. Comput. Phys.*, vol. 187, no. 1, pp. 110–136, May 2003.

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

[8] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006, pp. 812–819.

[9] L. A. Vese and T. F. Chan, "A multiphase level set framework for image segmentation using the mumford and shah model," *Int. J. Comput. Vision*, vol. 50, no. 3, pp. 271–293, Dec. 2002.

[10] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Physics-inspired topology changes for thin fluid features," in *ACM SIGGRAPH 2010 papers*, ser. SIGGRAPH '10. New York, NY, USA: ACM, 2010, pp. 50:1–50:8.

[11] T. Brochu, C. Batty, and R. Bridson, "Matching fluid simulation elements to surface geometry and topology," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 47:1–47:9, Jul. 2010.

[12] T. F. Dupont and Y. Liu, "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function," *J. Comput. Phys.*, vol. 190, no. 1, pp. 311–324, Sep. 2003.

[13] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, "An unconditionally stable maccormack method," *J. Sci. Comput.*, vol. 35, no. 2-3, pp. 350–371, Jun. 2008.

[14] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki, "Realistic animation of fluid with splash and foam," *Computer Graphics Forum*, vol. 22, no. 3, pp. 391–400, 2003.

[15] B. Kim, Y. Liu, I. Llamas, X. Jiao, and J. Rossignac, "Simulation of bubbles in foam with the volume control method," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[16] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[17] M. Kang, R. P. Fedkiw, and X.-D. Liu, "A boundary condition capturing method for multiphase incompressible flow," *J. Sci. Comput.*, vol. 15, no. 3, pp. 323–360, Sep. 2000.

[18] J.-M. Hong and C.-H. Kim, "Discontinuous fluids," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 915–920, Jul. 2005.

[19] Blender, "Blender.org – home," 2012, [Online; accessed April-13-2013]. [Online]. Available:

[20] Yafaray, "Home — yafaray," 2012, [Online; accessed April-13-2013]. [Online]. Available:

[21] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[22] X.-D. Liu, R. P. Fedkiw, and M. Kang, "A boundary condition capturing method for poisson's equation on irregular domains," *J. Comput. Phys.*, vol. 160, no. 1, pp. 151–178, May 2000.

[23] R. Courant, K. Friedrichs, and H. Lewy, "Uber die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, no. 1, pp. 32–74, Dec. 1928.

[24] B. Kim, "Multi-phase fluid simulations using regional level sets," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 175:1–175:8, Dec. 2010.

[25] P. W. Cleary, S. H. Pyo, M. Prakash, and B. K. Koo, "Bubbling and frothing liquids," in *ACM SIGGRAPH 2007 papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[26] O. Busaryev, T. K. Dey, H. Wang, and Z. Ren, "Animating bubble interactions in a liquid foam," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 63:1–63:8, Jul. 2012.

[27] S. T. Greenwood and D. H. House, "Better with bubbles: enhancing the visual realism of simulated fluid," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 287–296.

[28] W. E. Lorensen and H. E. Cline, "Seminal graphics." New York, NY, USA: ACM, 1998, ch. Marching cubes: a high resolution 3D surface construction algorithm, pp. 347–353.

[29] Nvidia, "Parallel programming and computing platform — cuda — nvidia," 2012, [Online; accessed April-13-2013]. [Online]. Available: