Multiphase Surface Tracking With Explicit Contouring

Xiaosheng Li *†ϟ Xiaowei He†ϟ Xuehui Liu†ϟ Baoquan Liu‡ Enhua Wu†§ϟ

†State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences §FST, University of Macau ‡Dept. of CST, University of Bedfordshire ϟUniversity of Chinese Academy of Sciences

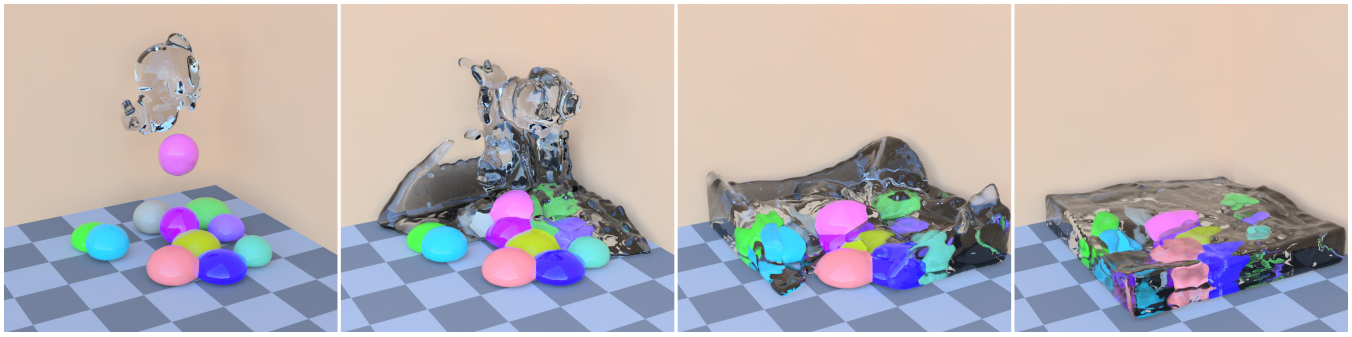

Figure 1: *Coupling of our surface tracker with a viscosity fluid simulator, 13 phases. The bunnies were assigned with low viscosities while the balls assigned with high viscosities. Fluid simulation was computed on a* 64^3 *grid while the surface tracker computed on a* 128^3 *grid.*

Abstract

We introduce a novel framework for tracking multiphase interfaces with explicit contouring technique. In our framework, an unsigned distance function and an additional indicator function are used to represent the multiphase system. Our method maintains the explicit polygonal meshes that define the multiphase interfaces. At each step, distance function and indicator function are updated via semi-Lagrangian path tracing from the meshes of the last step. Interface surfaces are then reconstructed by polygonization procedures with precomputed stencils and further smoothed with a feature-preserving non-manifold smoothing algorithm to stay in good quality. Our method is easy to be implemented and incorporated into multiphase simulation, such as immiscible fluids, crystal grain growth and geometric flows. We demonstrate our method with several level set tests, including advection, propagation, etc., and couple it to some existing fluid simulators. The results show that our approach is stable, flexible, and effective for tracking multiphase interfaces.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: multiphase simulation,level set,surface tracking

1 Introduction

There are many phenomena in daily life involving the interactions between multiphase components, such as dry foams, beer bubbles, and oil water mixing. Since the movements of the multiphase interfaces are so complicated with lots of degeneracies, such as multiple junctions, triple lines, capture of these interfaces is highly challenging.

A variety of methods have been proposed to handle this problem, including front tracking methods, volume of fluid (VOF) methods, level set methods, et al. These methods were normally built on the traditional two-phase methods and extended to handle many-phase problem with complicated modifications. Although there are both advantages and disadvantages to each of these methods, it remains challenging to robustly and accurately handle the complex motions of the multiphase interfaces.

On the other hand, the multiphase interfaces are essential for visualization, however, the abstraction of interface surfaces is not trivial accounting for more than two phases. If we apply ordinary polygonization procedures for each phase, we will probably get inconsistent surfaces between neighboring phases, which requires some post-processing techniques to fix.

To address these problems, we present a new framework to track the multiphase interfaces with explicit contouring technique. In our framework, an unsigned distance function and an additional indicator function are used to represent the multiphase system on a fixed Eulerian grid. We maintain explicit polygonal meshes that define the multiphase interfaces. At each step, distance function and indicator function are updated via semi-Lagrangian path tracing from the meshes of the last step. Interface surfaces are then reconstructed and further processed to stay in good quality.

There are several advantages of our method. Firstly, as we update the distance function by explicitly computing the distance to the surface meshes we avoid interpolations of distance field across the interfaces, which are often difficult to accomplish with high accuracy in traditional level set methods. Secondly, the reconstruction of the interface surfaces enables us to automatically handle any topological changes of the multiphase interfaces. Topological changes occur naturally and no gaps, overlaps, or ambiguities occur when multiphase interfaces evolve. The interface surfaces are consistent across the phases and can be used for physical computation and visualization directly. Finally, our method is mostly very intuitive

*e-mail:lixs@ios.ac.cn

VRST 2014, November 11 – 13, 2014, Edinburgh, Scotland, UK.

http://dx.doi.org/10.1145/2671015.2671017

and easy to be implemented and incorporated into multiphase simulation (Fig. 1).

The combination of explicit contouring with semi-Lagrangian method has been used by Bargteil et al. [2006] and named as *Semi-Lagrangian Contouring* (SLC) method. However, its application to multiphase surface tracking has not been explored ever and its extension to multiphase is far from trivial as we need to identify each phase and a reliable method to reconstruct the multiphase interface surfaces. We thus augment SLC method with an additional indicator function and a method to reconstruct the multiphase interface surfaces.

To reconstruct the multiphase interface surfaces accurately and efficiently, we explore the *Voronoi Implicit Interface Method* (VIIM) presented by Saye and Sethian [2011], which points out that the motions of the interfaces are determined by nearby level sets. We thus propose constructing the multiphase interfaces utilizing this property with custom polygonization and smoothing procedures. Though our method shares similar benefits of VIIM by employing the idea of Voronoi interface, our method has a few differences. As we take an explicit way to compute the distance function, we don't need to extrapolate distance function across the interface and we completely avoid the interpolations near the interface which may require a high order interpolant. Besides, we reconstruct the multiphase interface surfaces by employing two polygonizations and specially designed smoothing procedure.

Our contribution is to develop a unified, robust method with explicit contouring techniques for multiphase surface tracking and demonstrate its effectiveness. Our approach automatically handles the complex topological changes of the interfaces (see Fig. 2) and the interface surfaces is well ready for visualization.

In next section, we briefly discuss some related works. Section 3 provides an overview of our method. Section 4 and Section 5 details how we update the multiphase system and cast multiphase interface tracking as an interface reconstruction problem, followed by our feature-preserving non-manifold mesh smoother in Section 6. Finally we demonstrate the results of our method in Section 7 and conclude our paper in Section 8.

2 Related Works

2.1 Multiphase Surface Tracking

Level set methods [Osher and Sethian 1988; Osher and Fedkiw 2002] are popularly used for surface tracking. Losasso et al. [2006] proposed to handle multiphase with a modified level set method. The method is effective and robust, however this method tends to use increasing memory when the number of phases getting larger while our method is capable of handling arbitrary number of phases with only a unified representation for all phases.

Zheng et al. [2006] proposed a regional level set method to track multi-manifold surface, by defining new operators for level sets. Though regional level set was used for bubble simulation in [Zheng et al. 2006; Kim et al. 2007], its application for more-than-two-phase fluids was not presented. Kim et al. [2010] further improved regional level set a regional level set graph to track the thin film between neighboring phases so as to address the problem of many-phase fluid simulation. The regional level set graph needs to be carefully updated and maintained, which complicates the method while our method is simple and is mostly very intuitive.

Recently, Saye and Sethian [2011; 2012] presented VIIM to track multiphase interfaces built on classical level sets with detailed numerical analysis. Their excellent works greatly inspired our idea of developing our multiphase surface tracker. Our work can be viewed as an explicit version of VIIM to an extent, but we can avoid the interpolation across the interface so as to eliminate the necessity of a highly accurate interpolant near the interfaces.

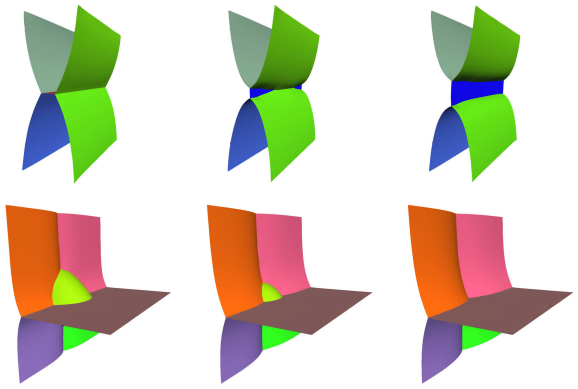

Figure 2: ***T1 process****: a short edge between vertices on distinct triple-junction curves collapses (top) and* ***T2 process****: a region shrinks to a point and disappears (bottom) under mean curvature flow. Our method automatically handles the complex topological changes of the interfaces* (128^3 *grid).*

Da et al. [2014] proposed the first triangle mesh-based multimaterial front tracking method. Misztal et al. [2012] tracked the multiphase interface using the *deformable simplicial complex*. Explicit surface tracking normally requires complicated mesh operations. As our method reconstructs the surface every step, our method can avoid such complicated explicit operations.

2.2 Interface Reconstruction

There has been a great deal of excellent works in the context of interface reconstruction. We will mainly focus on interface reconstruction related to multiphase.

Bloomenthal and Ferguson [1995] presented one of the first approaches for generating non-manifold surfaces defined by multiple regions of space. Ronald and Kevin [2005] proposed an adaptive polygonization of non-manifold implicit surfaces with octree subdivision. Hege et al. [1997] extended the basic marching cubes algorithm by allowing multiple vertex classes with an automatic method for generating topologically correct triangulations. Bertram et al. [2005] generated a quadrilateral for every edge connecting voxels with two different materials on a dual grid composed of voxels. Bernhard et al. [2005] made use of domain subdivision to construct non-manifold meshes from multi-labeled volumetric datasets. Wu and Sullivan [2003] proposed the *multi-material marching cubes* (M3C) algorithm, which extracted boundary surfaces between different materials. Most of these works deal with multi-label data without distance value and generate surfaces that are suffered from stair-stepped characteristics. Our polygonization procedure takes the information of level set into consideration and approximates the normal direction in cell corners to improve the accuracy of intersection computation. We also design a new set of stencils which is suitable for the polygonization of arbitrary number of phases.

Dey et al. [2012] applied a recent Delaunay refinement algorithm to generate high quality triangular interface surfaces. Bronson et al. [2013] introduced a new algorithm for generating tetrahedral meshes that conform to volumetric domains of multiple materials. Saye and Sethian [2012] extracted the multiphase interfaces with

piecewise linear interpolation and further quality-improved [Saye 2013] under the framework of VIIM, which requires the continuous piecewise interpolation of every distance function and mesh abstraction and chopping. Anderson et al. [2008; 2010] addressed the *Material Interface Reconstruction* problem in standard VOF method.

2.3 Surface Smoothing

One prevalent approach to surface smoothing is Laplacian smoothing, which moves every vertex towards a weighted average of its neighboring vertices. Original Laplacian smoothing severely shrinks the volume or smears the sharp features. Vollmer et al. [1999] proposed an HC-algorithm to prevent the effect of shrinking. Taubin [1995] introduced two-pass non-shrinking filter under a signal processing framework. Desbrun et al. [1999] introduced an implicit integration of the diffusion equation for the smoothing of meshes. Recent work of Jiao [2007] and Brochu [2009] suggested using *null-space* smoothing for features preserving.

Hubeli and Gross [2000] designed a multilevel fairing algorithm to remove noise for non-manifold meshes based on existing fairing operators along with an exact local volume preservation strategy and a method for feature preservation. We take the similar idea for non-manifold feature preservation, but we use different feature abstraction strategy and smoothing algorithm.

3 Method Overview

At each point x in the domain, let $\phi(x)$ be the positive distance from x to the nearest point on the interface, $\chi(x)$ be the indicator function, which indicates the fluid phases. Multiphase interfaces are now lying on where $\phi(x) = 0$.

We formulate the tracking of multiphase interfaces as a surface contouring problem. At each time step. $\phi(x)$ is traced backward along the streamline and its value is recomputed with the *semi-Lagrangian contouring* method [Bargteil et al. 2006]. $\chi(x)$ is is traced back similarly, but is only assigned with the nearest phase id without interpolation.

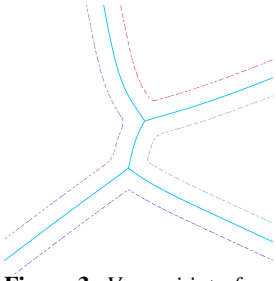

Figure 3: *Voronoi interface (solid line) is defined as the Voronoi diagram of nearby ϵ-surfaces (dashed lines).*

To reconstruct the interface surfaces, we employ two passes of explicit polygonizations. In the first pass, we reconstruct the ϵ-surfaces from the unsigned distance field $\phi(x)$, i.e. $\phi = \epsilon, \epsilon > 0$, via *Marching Cubes* (MC) algorithm [Lorensen and Cline 1987], which remain hypersurfaces in a single phase during the interface evolution [Saye and Sethian 2011]. Then, we are able to reconstruct the multiphase interfaces as the Voronoi interface of the ϵ-surfaces (Fig. 3). During the reconstruction, rather than directly computing the Voronoi interface of ϵ-surfaces, we propose a new polygonization algorithm based on *Marching Tetrahedra* (MT) algorithm [Gueziec and Hummel 1995] to approximate the interfaces efficiently. Interface surfaces are subsequently smoothed and quality-improved with a feature-preserving non-manifold mesh smoother.

Here we outline our method formally. Given an N-phase problem marked with an unsigned distance function $\phi(x)$ as well as an indicator function $\chi(x)$ defined on an Eulerian grid, our multiphase surface tracking consists of the following steps:

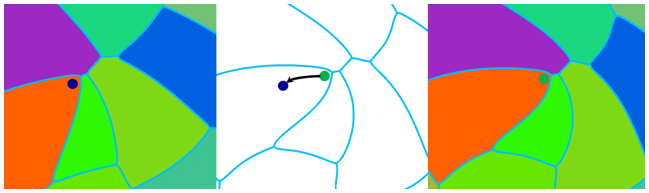

Figure 4: *Update of the distance and indicator functions: For a point (green circle) in space, we trace back a step to find its position (blue circle) in last step (middle); distance function are then updated by computing the distance to interface surfaces in last position, indicator function are also updated with the nearest indicator (left); once the distance and indicator function are updated, we process to construct the new interface surfaces (right).*

1. Update ϕ and χ.
2. Reconstruct nearby ϵ-surfaces.
3. Reconstruct multiphase interfaces as the Voronoi interface of ϵ-surfaces, referred to as interface surfaces hereinafter.
4. Smooth interface surfaces to improve the mesh quality.

Step 1 is performed just the same as the ordinary semi-Lagrangian method except that we update ϕ by directly computing the nearest distance to the interfaces as [Bargteil et al. 2006]. χ is is simply assigned to the nearest id since it makes no sense to interpolate with different phase ids (Sec. 4).

Step 2 is to triangulate surfaces lying on where $\phi = \epsilon$. The algorithm is basically the same as the standard MC algorithm, except we record the phase ids during our reconstruction for later use.

The key to step 3 is an accurate and fast way to compute the Voronoi interfaces of ϵ-surfaces. Since it's difficult to compute Voronoi Diagram of triangles directly, we propose to use an approximate but very efficient method to extract Voronoi interfaces of the ϵ-surfaces for computer graphics (Sec. 5).

For step 4, we provide a feature-preserving non-manifold mesh smoother to get high quality surface meshes (Sec. 6).

Fig. 4 shows the update process of step 1 and Fig. 5 gives an overview of the multiphase interface reconstruction in later steps.

4 Multiphase Semi-Lagrangian Contouring

To track multiphase surfaces, traditional level set methods meet some difficulties in handling the multiphase interfaces. If we only use one level set for all phases, the interpolation near the interface can be very inaccurate. Otherwise, if we use level set for each phase, the memory usage will increase quickly as the number of phase increases. Besides, it is difficult to deal with the inconsistency between neighboring phases. The abstraction of consistent interface meshes is also difficult in such case.

To solve these problems, we apply the *semi-Lagrangian contouring* to compute the distance to interface directly from the surface meshes and formulate the multiphase surface tracking as an explicit surface contouring problem. There are several benefits with this treatment. First, we do not need to worry about the interpolation errors across the interfaces, especially in the junctions of multiphase interfaces. Besides, we are not required to solve the level set equation explicitly but can still benefit from the nice features of level

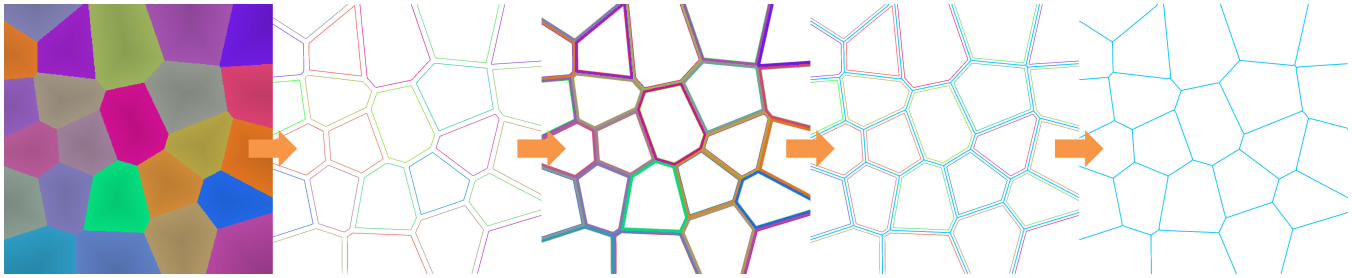

Figure 5: *Overview of our multiphase interface reconstruction : 1. Given ϕ and χ; 2. ϵ-level sets is triangulated; 3. Narrow band cells are labeled; 4. Interfaces are reconstructed and smoothed; 5. Final interface surfaces.*

set methods, which handles the topological changes automatically. Finally, by this formulation, there is no inconsistency between neighboring phases and the interface surfaces are well defined, i.e. no gaps, overlaps or ambiguities between neighboring phases.

However, there still remains a problem: The update of indicator function is not accurate enough as desired since we update it simply with the nearest indicator. This treatment may result in low accuracy near the multiphase interfaces. As demonstrated in the work of Saye and Sethian [2011; 2012], the indicator function is accurate inside the ϵ-surfaces (where $\phi > \epsilon$), thus we can reconstruct the interface surfaces as the Voronoi interface of ϵ-surfaces (Fig. 3). In this way, we do not have to use the information of indicator function near the interfaces and we do not have to explicitly deal with the topological changes of the interface surfaces, which is handled by the level set method implicitly. The only thing left is how to construct the interface surface meshes, which will be detailed in later sections.

Be noted that, though it does not affect the reconstruction of the interface surfaces, we may have to reinitialize χ when there are numerous phase merging or separations. We can reinitialize χ by checking relative orientations with respect to the triangulated interface surfaces.

5 Polygonization of Multiphase Level Set

Once we have triangulated the ϵ-surfaces, we can reconstruct multiphase interfaces as the Voronoi interface of ϵ-surfaces. By definition, points on the multiphase interfaces are those have equal distances to at least two different ϵ-surfaces. Based on this property, we first compute the nearest point on ϵ-surfaces for each cell corner and label them with phase id of that nearest point. Then we identify the cells that have different ids in the corners as boundary cells, from which the surfaces are reconstructed.

As interfaces can only lie between ϵ-surfaces of different phases, we can restrict our computation to a small narrow band where $\phi <= \epsilon$. In our implementation, we record the nearest point on ϵ-surfaces, the nearest distance, and the phase id for each cell corner in the narrow band. To compute the nearest point, we build a kd-tree of the surface meshes for fast retrieval of nearest triangles. This procedure thus is greatly accelerated by excluding unnecessary cells and space subdivision technique.

To remove ambiguities, we assume that there are no multiple intersections between two neighboring cell corners. This assumption is taken by most of polygonization methods and works well in practice.

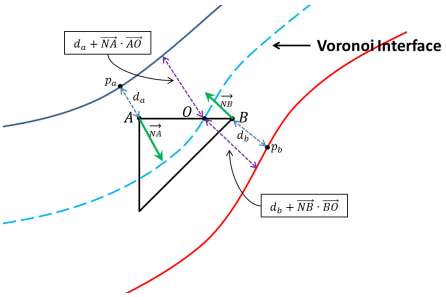

Figure 6: *Intersection computation between A and B: $d_a + \vec{NA} \cdot \vec{AO} = d_b + \vec{NB} \cdot \vec{BO}$.*

5.1 Intersection Computation

Intersection between two neighboring corners with different ids is computed as follows (see Fig. 6): Suppose we want to compute the intersection between corner A and B, denoted as O and $\overrightarrow{BO} = \alpha\overrightarrow{BA}$. The nearest points on ϵ-surfaces are p_a and p_b, with their distances d_a and d_b, respectively. By definition, O is equidistant to the ϵ-surfaces on both sides, we can get:

$$d_a + \overrightarrow{NA} \cdot [(1-\alpha)\overrightarrow{AB}] = d_b - \overrightarrow{NB} \cdot (\alpha\overrightarrow{AB}) \qquad (1)$$

where $\overrightarrow{NA}$ and $\overrightarrow{NB}$ are the normals at point p_a and p_b, respectively, we approximately assume point A and B have the same normals as their nearest points on ϵ-surfaces.

Equ. 1 estimates the distances from ϵ-surfaces to the intersection point on both sides along the normal directions. As it's only linear interpolation, the surfaces might be quite rough especially for multiphase interfaces. In our implementation, we improve this by estimating the normals in the corners and the results are improved to be quite smooth.

5.2 Precomputed Stencils

To get high quality surface meshes, we divide the cubes into tetrahedrons which can be polygonized with stencils depicted in Fig. 7. Our treatment makes it possible to precompute stencils and we can locate the junctions within a tetrahedron accurately. Besides, since there are no ambiguities in MT algorithm [Gueziec and Hummel 1995], the reconstructed surface meshes are guaranteed to be topologically consistent.

We briefly discuss the stencils here. For simplicity, we use a triple (id_0, id_1, id_2) or a tetrad (id_0, id_1, id_2, id_3) to represent the configuration of a triangle or a tetrahedron,respectively. Each component represent an id.

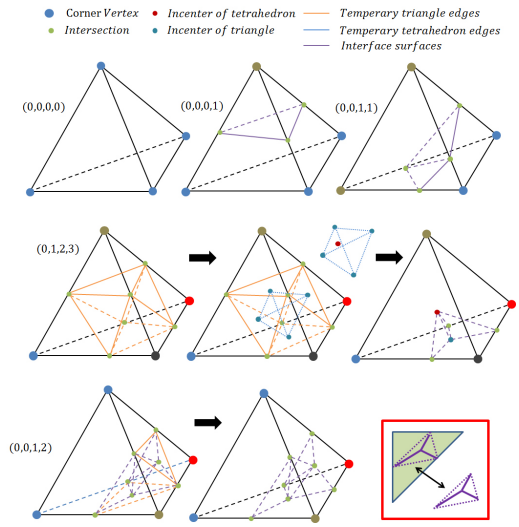

Figure 7: *Stencils for our multiphase polygonization. Note that in* $(0, 1, 2, 3)$ *we only show interface surfaces connected to one face of the tetrahedron for better illustration.*

We subdivide the triangle or the tetrahedron by taking advantage of the properties of the incenter of a triangle and tetrahedron (see lower right corner of Fig. 7). Take $(0, 1, 2, 3)$ as an example, as the incenter has equidistance to the edges or faces, we can construct four temporary triangles on faces of tetrahedron after getting the intersection points on all edges. Then, we connect their incenters together to form a temporary tetrahedron. For each temporary triangle, we connect the incenter of the temporary tetraheron and its incenter with its three intersection points to form three triangles. In this case, 12 triangles will be generated in all (Fig. 7).

Though our set of stencils is quite small, our polygonizer can deal with arbitrary number of phases without extra efforts.

6 Mesh Smoothing

Though we have carefully designed the stencils, it's still necessary to smooth the interface surfaces to mitigate the visible stair-stepped artifacts, which are caused by some inaccurate interpolations near the multiphase interfaces in intersection computation. Notice that most bumps appear in the non-manifold curves. Based on this observation, we present a feature-preserving non-manifold smoothing algorithm to remove the stair-step artifacts.

6.1 Feature Abstraction

For multiphase surface tracking, it's most important to track the multiple junctions, triple points, triple lines, etc., thus we design our feature abstraction especially for these features.

We will describe our framework with the following structures:

- *Vertices* V: Vertices in interface surfaces.
- *Edges* E: Edges in interface surfaces.
- *Feature vertices* V_F: Vertices on boundary, curve corners or non-manifold edges.
- *Anchor feature vertices* V_A: Vertices connected to only one or more than two feature edges.
- *Feature edges* E_F: Boundary edges or non-manifold edges.

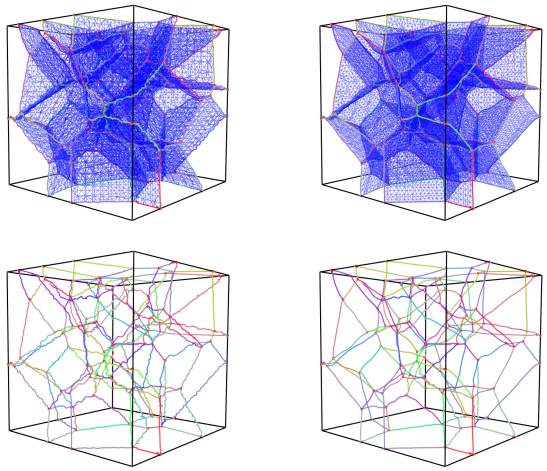

Figure 8: *A rough multiphase interface surfaces smoothed by our algorithm with 5 iterations. Anchor feature vertices (red spheres) and curves (different colors) are shown in the bottom row.*

- *Feature curves* C_F: Curves formed by feature edges.
- *Manifold planes* P_M: Near-planar planes formed by planes incident on non-manifold curves.

Algorithm 1 lays out our feature abstraction algorithm and relies on the data structures described above.

Algorithm 1 Feature abstraction

1: Locate E_F in E
2: Locate V_F, V_A in E_F
3: Construct C_F from V_A and E_F
4: Construct P_M from C_F
5: Compute supplement of V_F from C_F

It's straightforward to locate feature edges and feature vertices by definitions. Anchor feature vertices are then computed by checking the number of feature edges incident on. As its name indicated, they are anchored in the whole process. [Line 1, 2 of Algorithm 1]

To compute the feature curves, we have to group feature edges into different connected components. Since anchor feature vertices, corresponding to junction points, are important in multiphase surface tracking, we use them as separations of different feature curves.

Our feature curve construction algorithm works by traversing from one anchor vertex to another or an endpoint along feature edges. This is efficiently done with depth first searching on interface surfaces. Some feature edges cannot be connected to any anchor feature vertex, e.g. boundary edges. For these feature edges, we randomly pick one and traverse to construct feature curve until all feature edges are attached to a curve. By construction, all feature edges appear in one and only one feature curve. [Line 3]

After constructing the feature curves, we perform additional corner feature vertex abstraction for feature curves by checking the angle formed by two edges incident on a vertex. We take it as a feature vertex if the angle exceeds a maximum value (like $\pi/3$). Curves are temporarily smoothed to remove noise for more accurate extraction. See Fig. 8 for an example of feature vertices and curves. [Line 5]

A feature curve can be embedded into a two-manifold plane if two planes incident on this curve form a very large dihedral angle

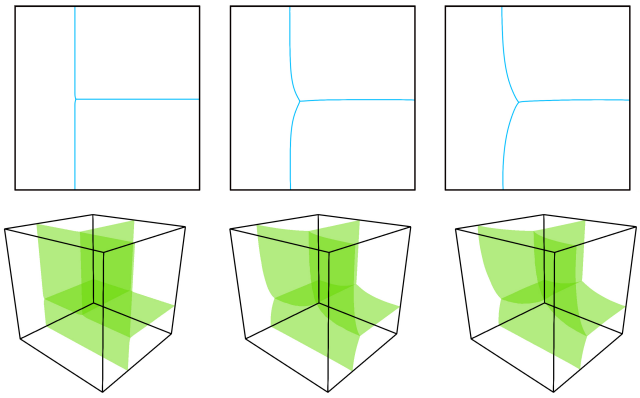

Figure 9: *Evolution of a T-junction to Y-junction subject to curvature flow. Note that the Y-junction will converge to triple lines and points with* 120° *angles according to Young's law.*

(near π). Thus we compute a near-planar manifold plane for each non-manifold curve. We check the planes incident on each non-manifold curve, and construct a near-planar plane by combining the two planes, which form the maximum dihedral angles and also exceed a certain value (like 0.9π). We accomplish this by traversing triangles incident on feature curves to construct incident planes and then check the angles they form. This plane would enable us to smooth across the feature curves. [Line 4]

6.2 Smoothing Algorithm

With mesh feature abstracted, we are now ready to smooth the interface surfaces while preserving the features. The smoothing algorithm could be detailed in Algorithm 2 below.

Algorithm 2 Smoothing algorithm

M = interface surfaces, n = iteration
for $i = 0; i < n; i++$ **do**
  Fix V_F, perform 2D-Laplacian smoothing for M
  Fix V_A, perform 2D-Laplacian smoothing for P_M
  Fix V_A, perform 1D-Laplacian smoothing for C_F
end for

For Laplacian smoothing, the original version shrinks the volume severely. We have tried HC-algorithm [Vollmer et al. 1999] and non-shrinking filter in [Taubin 1995] for volume preservation. We found that HC-algorithm outperformed the latter one in our application, thus we adopt HC-algorithm for our algorithm. Note that HC-algorithm is not completely free from shrinkage, but yields quite satisfactory results. If exact volume preservation is required, more complicated local volume preserving technique from [Hubeli and Gross 2000] might be used.

7 Results

We have performed several tests to demonstrate the capabilities of our method. For all our experiments, we set $\epsilon = 2h$ (h is the grid cell size) and use 5 iterations smoothing for a tradeoff between accuracy and computation cost. All computations were computed on a PC with 2.7GHz CPU and 12 GB memories in single thread. Resolution in all examples ranged from 64^3 to 256^3. Computation times for distance computation and smoothing ranged from less than a second to a few minutes depending on the resolution and also size

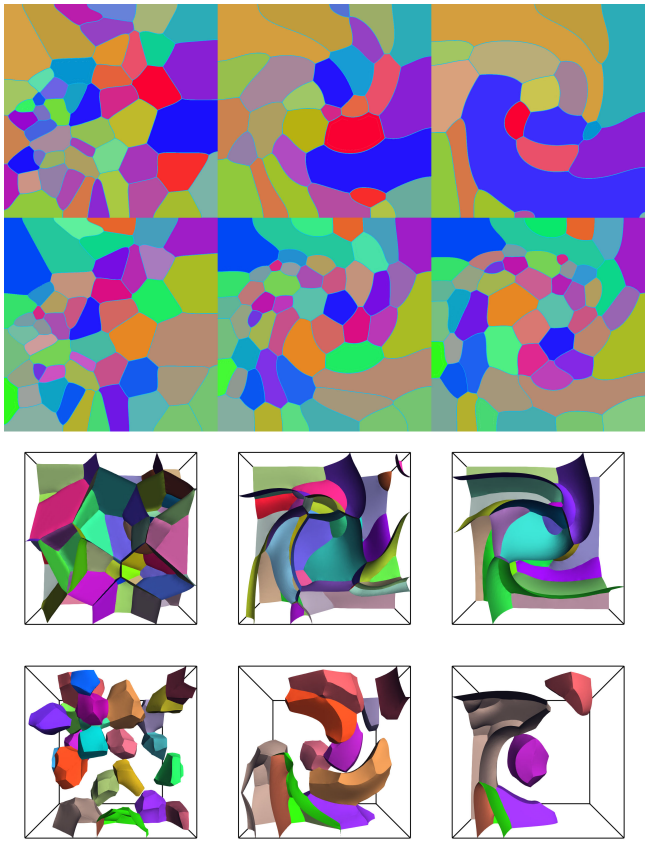

Figure 10: *Advection of Voronoi cells with agitator velocity (from top to bottom): 2D motion of 50 random phases with/without diffusion (interface shown in blue line), 3D motion of 50 random with diffusion (interface surfaces shown), 3D motion of 200 random phases with diffusion (subset of phases shown).* 128^3 *grid.*

of the narrow band (related to the multiphase interfaces). Polygonizations normally took a few seconds. Some detailed timings are summarized in Table 1 and some analysis in Sec. 7.4.

Here we first specify how to choose the value of ϵ. We couple ϵ and h ($\epsilon = \beta h$) and choose $\epsilon \geq 2h$ so that any finite difference stencils used to evolve the unsigned distance functions stay completely in one phase and do not across the interface and the indicator function is accurate away from the interface. In addition, the size of narrow does not need to increase as h decrease and this choice also fits naturally into simulations with CFL-type (i.e. fixed λ) requirement relating time step Δt to h. We suggest to use $\beta \geq \lambda$ in these simulations to ensure accuracy. However, larger ϵ gives larger numerical errors generally. We use $\lambda = 1$ for our simulations, thus we choose $\epsilon = 2h$. For more numerical experiments on the choice of ϵ, please refer to the original VIIM[Saye and Sethian 2012].

7.1 Convergence Test

As in multiphase, exact solutions are less well-known, we test our method with the motion of a single T-junction evolving into a Y-junction under curvature flow. It's well known that the motion will result in 120° angles, as indicated by the Young's law. We consider Neumann boundary conditions in both 2D and 3D. Fig. 9 shows the snapshot of the evolution, which ultimately converged to triple points and lines. Thus convergence can be achieved by our method.

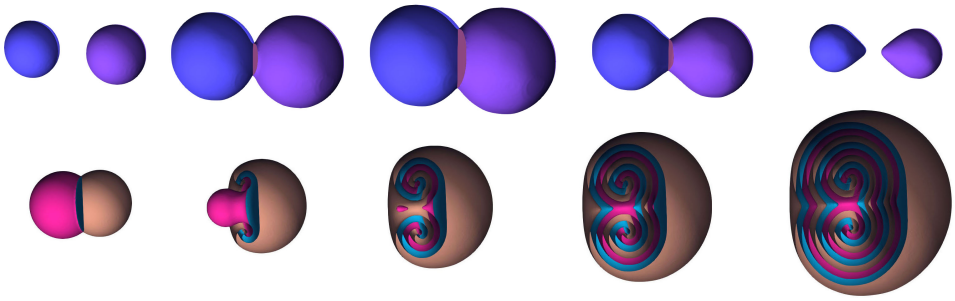

Figure 11: *Normal flows: Two spheres underwent outward/inward normal flow(top, 128^3 grid); Two overlapping spheres underwent normal flow with a cyclical ordering (bottom, 256^3 grid). The interface surfaces are cut away for visualization.*

Example	Resolution	Advection	Interface reconstruction		Interface smoothing		Total
			Labeling	Polygonization	Feature abstraction	Smoothing	
1	128^3	9.22	3.47	0.34	2.85	0.42	16.3
2	256^3	10.19	2.35	0.23	1.52	0.11	14.4
3	128^3	-	5.48	0.64	5.23	0.78	12.13

Table 1: *Average computation time (sec/step) of our algorithm in selected examples: 1. Advection of 50 random Voronoi phases in Fig. 10; 2. Cyclical normal flow in Fig. 11; 3. Propagation of 200 random Voronoi phases in Fig. 12.*

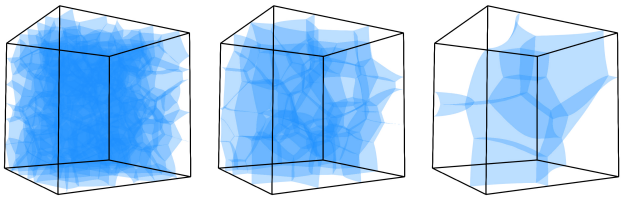

Figure 12: *200 random Voronoi cells evolved under mean curvature flow, leading to large number of complex topology changes. 128^3 grid.*

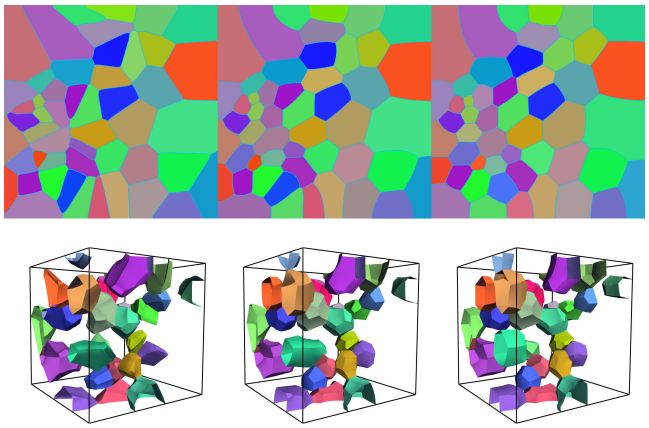

Figure 13: *Random Voronoi cells motions under mean curvature flows: 2D motion of 50 random phases with area conservation (top), 3D motion of 200 random phases with volume conservation (bottom). 128^3 grid.*

7.2 Advection

Several advection tests were performed for our surface tracker. Instead of using a distance tree, we only maintain a kd-tree for fast distance computation both in advection and polygonization procedure to simplify our implementation. We use governing equation:

$$\phi_t + \mathbf{u} \cdot \nabla\phi = M\sigma\kappa|\nabla\phi| \tag{2}$$

for advection test, where $\mathbf{u}$ is the external driving velocity, κ is the mean curvature of the interface and σ is the surface tension coefficient. M denotes the amount of permeability. $M = 0$ means no permeability and the equation describe pure advection of ϕ along $\mathbf{u}$ while $M > 0$ adding some diffusions of ϕ.

We first considered advecting random Voronoi cells defined on domain $[0, 1]$ with an prescribed external velocity:

$$\mathbf{u}(x, y, z) = (-sin^2(\pi x)sin(2\pi y), sin^2(\pi y)sin(2\pi x), 0) \tag{3}$$

with/without diffusions ($M = 1/M = 0$) using zero Neumann boundary conditions, as showed in Fig. 10. External velocity caused significant shearing, phase rearrangement and topological changes, which were all automatically handled by our method. In the case of diffusions, some phases were collapsed while others grew larger.

We also conducted a series test of normal flows. We first compute velocity as $\mathbf{u} = k\mathbf{N}$, where k is an adjustable parameter and $\mathbf{N}$ the normal of the interface and apply the advection equation without diffusion to simulate normal flow. Top row of Fig. 11 shows the results of two spheres driven by outward/inward normal flow. Bottom row of Fig. 11 shows a difficult case in level set methods. Three phases were restricted to flow in a cyclical order to form a stationary triple line. Our method is capable of handling the complex motions of this case.

It should be noted that in the case of multiphase simulation, the normal should be carefully defined. We use a simple but consistent way to define it. We flip the unsigned level set of neighboring cell to negative if it has a different id comparing to the center cell when computing the normal. In this way, we do not have to construct a signed level set for each phase and the normals between different phases can be naturally interpolated across the interfaces without extra special treatment. The surface tension is defined in the same manner.

7.3 Propagation

In level set propagation tests, our tracker was used to construct interface surfaces and frequently reinitialize ϕ and χ. The governing

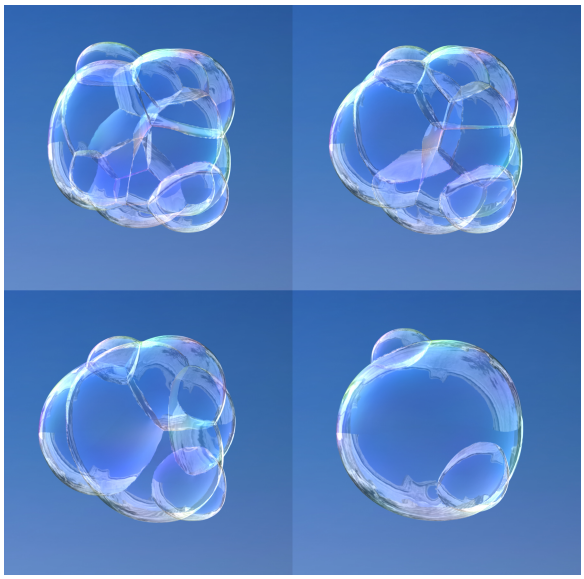

Figure 14: *Bubble formulation: a cluster of bubbles randomly merged together and driven by volume conservation mean curvature flow (64^3 grid).*

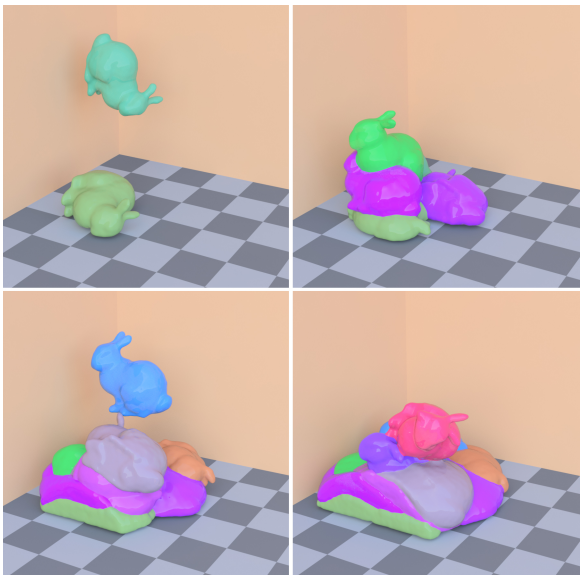

Figure 15: *11 viscous bunnies dropped together into a pile under gravity (128^3 grid).*

equation for propagation is $\phi_t = \sigma\kappa|\nabla\phi|$, which is integrated with forward Euler stepping. This is actually the mean curvature flow.

Fig. 2 demonstrates two basic configurations to test the accuracy of our method under mean curvature flow, *T1* and *T2* process were reproduced correctly without any extra efforts. Fig. 12 and Fig. 13 show the results of Voronoi cells driven by mean curvature flow, leading to a large number of complex topological changes. Volume conservation technique in [Saye and Sethian 2011] was used in example of Fig. 13. Areas and volumes can be easily computed in polygonization procedure with a few lines of code. Neumann boundary conditions were used for all examples above.

We also performed a test to predict the bubble formulation with volume conservation mean curvature flow. Bubbles were randomly merged during the formulation. Mean curvature flow minimizes the total surface area of all interfaces subject to volume conservation constraints for each phase. Our method correctly predicted the shapes of the bubbles, as showed in Fig. 14. The bubbles tend to minimal their surface areas and makes about 120° at triple lines (Plateau's laws).

7.4 Coupling with Fluid Simulator

We coupled our surface tracker with several fluid simulators loosely. Fluids were modeled as immiscible. The fluid simulator provided the surface tracker with a velocity field while the surface tracker provided the simulator with the unsigned distance and indicator function in turn. In this way, the fluid simulator and surface tracker can run with independent resolution.

Fig. 1, Fig. 15 and Fig. 16 showed the results of coupling our surface tracker to a viscosity fluid simulator [Batty and Bridson 2008]. Different viscosities were assigned to bunnies and spheres in Fig. 1 while same high viscosities were used for all phases in Fig. 15 and Fig. 16. Fig. 15 demonstrated eleven bunnies dropped into a pile while Fig. 16 demonstrated a more complicated scene with sixty spheres dropped together to create complex interfaces.

Fig. 17 depicted five viscoelastic balls fell on a pool by coupling our method with the viscoelastic fluid simulator in [Goktekin et al. 2004]. The balls jumped on the surface of the pool, leading to fast changing interfaces.

Fig. 18 showed two layers of fluid where the bottom layer fluid is lighter than the top layer fluid, which causes Rayleigh-Taylor instability when two layers of fluid switch places. This is an extremely challenging case to test our surface tracker. Our tracker is able to track the interface but still suffers from some detail losses.

Fig. 19 depicted dropped five balls into a pool separated by different phases. The simulation is driven by an ordinary inviscid fluid simulator and the mean curvature flow. The multiphase interfaces finally converged to an equilibrium with various topological change occurred.

Finally, we simulated two liquid droplets colliding in zero gravity in Fig. 20. The two liquid droplets are separated in the beginning and driven by surface tension. In the process of evolution, they touched each other and merged together under the surface tension force. After that, external forces were applied to tear them apart.

To get an intuitive impression of the computational efficiency of our method, we compare the timing of fluid simulator to that of the surface tracker. In the viscosity fluids simulation of Fig. 16, the simulation took average 2 minutes each step while the surface tracker took about 43 seconds (28s for advect, 6s for labeling, 7s for smoothing and rest for others) for 60 phases (right lower of Fig. 16). The surface tracker thus took less than 26.38% of the total computational time. In fact, timing for surface tracker would be much smaller when there are fewer phases. For the Rayleigh-Taylor instability example, the fluid simulator took average 185 seconds each step while the surface tracker took average 32 seconds, 14.75% of the computational cost. In examples with a small number of phases or less complicated interfaces, the surface tracker only took a small fraction of the total computational time.

The interfaces between different fluids with various merging and separations were clearly captured by our method as expected. Topological changes were handled automatically. The interface surfaces were used to render the final images directly.

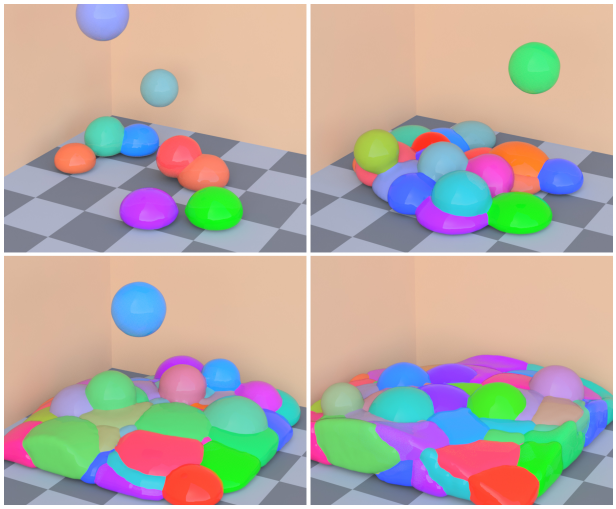

Figure 16: *60 viscous spheres dropped together to form complex interfaces (128^3 grid).*

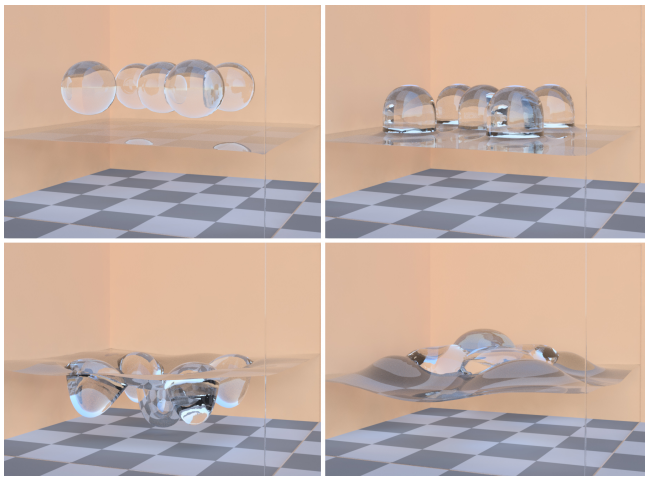

Figure 17: *Coupling of our multiphase surface tracker with a viscoelastic fluid simulator: five balls fell on a quiet pool (128^3 grid).*

8 Discussion and Conclusions

We provide an elegant and effective tool for multiphase surface tracking, which is capable of handling topological changes automatically. It's robust, flexible and easy to implement.

Though we have shown its properties, there are still some limitations. We use a fixed Eulerian grid for computation, which occupies large memories. One promising direction is to use memory-efficient data structure or adaptive grid. Since we rely on ϵ-surfaces to reconstruct the interface surfaces, thin features that can be captured are limited by the choice of ϵ. Interface surfaces involving regions, whose interior level set values are smaller than ϵ, will be totally missed. ϵ also affects the largest time step we can use. Visual flickers can occur since we reconstruct the interface surfaces every time step. This may be mitigated by exploring the temporary coherence of the evolving interfaces. Finally, our method still inherits the limitation of the original level set method with regards to volume loss. This can be addressed by increasing the resolution or employing other methods like particle level set to improve it.

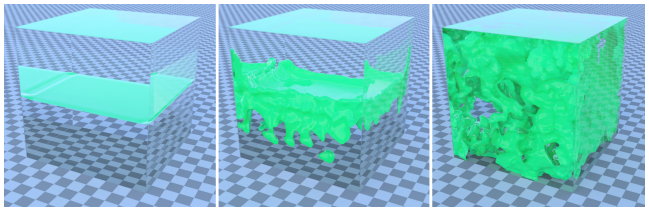

Figure 18: *Rayleigh-Taylor instability: Our surface tracker was able to capture the complex interfaces caused by the turbulent fluid (128^3 grid).*

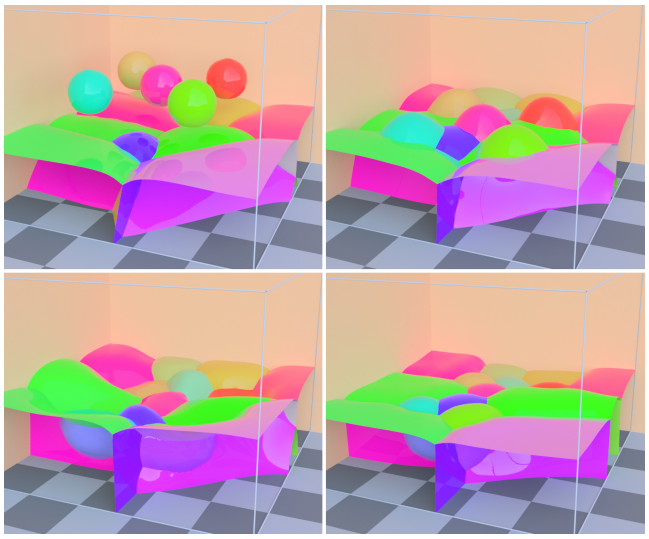

Figure 19: *Five balls dropped into a pool separated by several different fluids. The simulation is driven by an ordinary inviscid fluid simulator and the mean curvature flow. The multiphase interfaces finally converged to an equilibrium (64^3 grid).*

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This research is supported by NSFC grant (61272326) and the grant of University of Macau.

References

ANDERSON, J. C., GARTH, C., DUCHAINEAU, M. A., AND JOY, K. I. 2008. Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum 27*, 3, 1015–1022.

ANDERSON, J. C., GARTH, C., DUCHAINEAU, M. A., AND JOY, K. I. 2010. Smooth, volume-accurate material interface reconstruction. *IEEE Trans. Vis. Comput. Graph. 16*, 5, 802–814.

BALSYS, R. J., AND SUFFERN, K. G. 2005. Adaptive Polygonisation of Non-Manifold Implicit Surfaces. In *Computer Graphics, Imaging and Vision*, 257–263.

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25*, 1 (Jan.), 19–38.

BATTY, C., AND BRIDSON, R. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings*

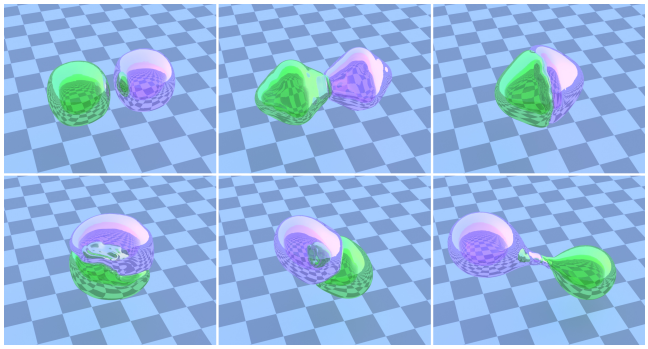

Figure 20: *Two droplets collided together under the surface tension and then torn apart by external forces (64^3 grid).*

of the 2008 ACM/Eurographics Symposium on Computer Animation, 219–228.

BERTRAM, M., REIS, G., LENGEN, R. H. V., K?HN, S., AND HAGEN, H. 2005. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. In *In Eurovis 05*, Eurographics Association, 1–10.

BLOOMENTHAL, J., AND FERGUSON, K. 1995. Polygonization of non-manifold implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '95, 309–316.

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4, 2472–2493.

BRONSON, J., LEVINE, J., AND WHITAKER, R. 2013. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *Proceedings of the 21st International Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds. Springer Berlin Heidelberg, 191–209.

DA, F., BATTY, C., AND GRINSPUN, E. 2014. Multimaterial mesh-based surface tracking. *ACM Trans. on Graphics (SIGGRAPH 2014)*.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 317–324.

DEY, T. K., JANOOS, F., AND LEVINE, J. A. 2012. Meshing interfaces of multi-label data with delaunay refinement. *Eng. with Comput. 28*, 1 (Jan.), 71–82.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004) 23*, 3, 463–468.

GUEZIEC, A., AND HUMMEL, R. 1995. Exploiting triangulated surface extraction using tetrahedral decomposition. *Visualization and Computer Graphics, IEEE Transactions on 1*, 4, 328–342.

HEGE, H.-C., SEEBASS, M., STALLING, D., AND ZÖCKLER, M. 1997. A generalized marching cubes algorithm based on non-binary classifications. Tech. Rep. SC-97-05, ZIB, Takustr.7, 14195 Berlin.

HUBELI, A., AND GROSS, M. 2000. Fairing of non-manifolds for visualization. In *Proceedings of the conference on Visualization '00*, IEEE Computer Society Press, Los Alamitos, CA, USA, VIS '00, 407–414.

JIAO, X. 2007. Face offsetting: A unified approach for explicit moving interfaces. *J. Comput. Phys. 220*, 2 (Jan.), 612–625.

KIM, B., LIU, Y., LLAMAS, I., JIAO, X., AND ROSSIGNAC, J. 2007. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. 26*, 3 (July).

KIM, B. 2010. Multi-phase fluid simulations using regional level sets. *ACM Trans. Graph. 29*, 6 (Dec.), 175:1–175:8.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (Aug.), 163–169.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. In *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, SIGGRAPH '06, 812–819.

MISZTAL, M. K., ERLEBEN, K., BARGTEIL, A., FURSUND, J., CHRISTENSEN, B. B., BAERENTZEN, J. A., AND BRIDSON, R. 2012. Multiphase flow of immiscible fluids on unstructured moving meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 97–106.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces (Applied Mathematical Sciences)*, 2003 ed. Springer, Nov.

OSHER, S., AND SETHIAN, J. A. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys. 79*, 1 (Nov.), 12–49.

REITINGER, B., BORNIK, A., AND BEICHEL, R. 2005. Constructing Smooth Non-Manifold Meshes of Multi-Labeled Volumetric Datasets. In *International Conference in Central Europe on Computer Graphics and Visualization*, 227–234.

SAYE, R. I., AND SETHIAN, J. A. 2011. The voronoi implicit interface method for computing multiphase physics. *Proceedings of the National Academy of Sciences 108*, 49, 19498–19503.

SAYE, R., AND SETHIAN, J. 2012. Analysis and applications of the voronoi implicit interface method. *Journal of Computational Physics 231*, 18, 6051 – 6085.

SAYE, R. 2013. An algorithm to mesh interconnected surfaces via the voronoi interface. *Engineering with Computers*, 1–17.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '95, 351–358.

VOLLMER, J., MENCL, R., AND MLLER, H. 1999. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum 18*, 3, 131–138.

WU, Z., AND SULLIVAN, J. M. 2003. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering 58*, 2, 189–207.

ZHENG, W., YONG, J.-H., AND PAUL, J.-C. 2006. Simulation of bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '06, 325–333.