Multiphase Interface Tracking with Fast Semi-Lagrangian Contouring

Xiaosheng Li, Xiaowei He, Xuehui Liu, Jian J. Zhang, Baoquan Liu, and Enhua Wu, *Member, IEEE*

Abstract—We propose a semi-Lagrangian method for multiphase interface tracking. In contrast to previous methods, our method maintains an explicit polygonal mesh, which is reconstructed from an unsigned distance function and an indicator function, to track the interface of arbitrary number of phases. The surface mesh is reconstructed at each step using an efficient multiphase polygonization procedure with precomputed stencils while the distance and indicator function are updated with an accurate semi-Lagrangian path tracing from the meshes of the last step. Furthermore, we provide an adaptive data structure, multiphase distance tree, to accelerate the updating of both the distance function and the indicator function. In addition, the adaptive structure also enables us to contour the distance tree accurately with simple bisection techniques. The major advantage of our method is that it can easily handle topological changes without ambiguities and preserve both the sharp features and the volume well. We will evaluate its efficiency, accuracy and robustness in the results part with several examples.

Index Terms—Multiphase interface tracking, semi-Lagrangian contouring, fluid simulation

1 Introduction

Physical phenomena involving interconnected moving interfaces are common in our daily life, including dry foams, beer bubbles, and mixing of multiple immiscible fluids. However, how to capture the motion of these interfaces is challenging as their movements are often complex, and include substantial topological changes. Various methods have been proposed to track multiphase interfaces, mainly including the *front tracking* method [1], the *volume of fluid* method [2] and the *level set* method [3]. Unfortunately, as the phase number increases, it remains a challenge for these methods to robustly and accurately handle the wide range of interface evolutions, especially at the degeneracies where multiple interfaces meet.

Recently, Saye and Sethian [4] proposed an implicit scheme, which is referred to as *Voronoi Implicit Interface Method* (VIIM), to track multiphase interfaces based on the combination of Voronoi diagrams and implicit level set methods [3], [5]. Instead of using multiple level set (MLS) functions, the VIIM applies an unsigned level set distance function together with a material indicator function to track the entire multiphase system. Compared to the level set methods, the VIIM can easily handle tens of thousands of separate phases and the topological changes without worrying about the memory overhead. For a more detailed numerical analysis, we refer to Saye and Sethian's recent work [6]. However, since the VIIM is built on the implicit level set method, several disadvantages are inherited. First, the VIIM suffers an excessive amount of volume loss in under-resolved regions of the flow. Second, since the VIIM relies on ϵ-surface (whose distance value is ϵ) to reconstruct multiphase interfaces, thin features such as the interfacial filaments and regions of high curvature will be smeared out. Finally, it is also not clear how to extend the method to an adaptive implementation as ϵ is required to be constant over the entire domain.

In this paper, we propose a new method which combines the best properties of a standard VIIM and an explicit contouring technique for improved accuracy and efficiency. Along with an unsigned distance function and an indicator function that are used to track the materials implicitly, we introduce an extra surface mesh to denote where exactly the interfaces are located at each time step. During the advection, we first use the surface mesh at the previous step as a reference to update the distance function and the indicator function with a semi-Lagrangian scheme, and then reconstruct the new surface mesh from the distance function and the indicator function. To get an accurate estimation of the distance function and the indicator function, we improve the *semi-Lagrangian contouring* (SLC) method proposed by Bargteil et al. [7] to make it suitable for multiphase interface tracking. We also develop a fully adaptive scheme to make the interface tracking as efficient as possible.

Our contributions in this paper include

- A material identification procedure to compute exact indicator function at any point of the domain.
- A multiphase polygonization procedure combined with bisection to contour the multiphase system.

- *X. Li and X. He is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences and the University of Chinese Academy of Sciences, Beijing 100190, China. E-mail: lixs@ios.ac.cn.*
- *X. Liu is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. E-mail: lxh@ios.ac.cn.*
- *J.J. Zhang is with the National Centre for Computer Animation, Bournemouth University. E-mail: jzhang@bournemouth.ac.uk.*
- *B. Liu is with the Department of Computer Science and Technology, University of Bedfordshire. E-mail: Baoquan.Liu@beds.ac.uk.*
- *E. Wu is with the State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences and the University of Macau. E-mail: ehwu@umac.mo.*

Manuscript received 14 Feb. 2015; revised 10 Apr. 2015; accepted 23 Aug. 2015. Date of publication 3 Sept. 2015; date of current version 6 July 2016. Recommended for acceptance by Y. Yu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2015.2476788

- A multiphase distance tree structure to resolve the distance function and indicator function efficiently.

By combining the best properties of VIIM and SLC, our method is capable of tracking multiphase interfaces with complex topological changes while preserving the surface details. Both accuracy and efficiency are greatly improved compared to the standard level set methods and the VIIM, as is demonstrated in the results part. In the next section, we briefly discuss some related work. Section 3 provides an overview of our method, followed by several sections that explain each part of our method in detail. Finally, we give the results of our method in Section 7 and conclude our paper in Section 8.

2 Related Work

2.1 Multiphase Interface Tracking

Level set methods [3], [5], [8], [9], [10] are commonly used in fluid simulation. For multiphase interface tracking, each phase is represented as a separate level set function, i.e., the multiple level set method. However, this may cause inconsistencies in the interfacial regions. To solve this problem, Merriman et al. [11] presented a predictive-corrective scheme and Losasso et al. [12] used a post-projection operation. Zhang et al. [13] further reduced the numerical errors to the grid resolution level by applying multiple corrector iterations. Starinshak et al. [14] also presented a new level set model to reduce the numerical errors. Memory overhead is one of the main concerns of MLS methods as it increases with the number of the phases, which easily becomes its bottleneck. Alternatively, Zheng et al. [15] proposed a regional level set method to track multi-manifold surfaces, by defining new operators for the level sets. Although the regional level set method has been widely used in bubbling dynamics [15], [16], the extension to more-than-two-phase applications was not clear. Later, Kim [17] improved the regional level set method by applying a regional level set graph to track thin films between adjacent phases to address the problem in multiphase fluid simulation. Saye and Sethian [4], [6] presented the VIIM, which is also built on classical level sets, to track multiphase interfaces with detailed numerical analysis.

More recently, Da et al. [18] proposed the first mesh-based multimaterial front tracking method. Misztal et al. [19] tracked the multiphase interface using the *deformable simplicial complex*. These mesh-based tracking methods can preserve thin features well, but also induce lots of complicated mesh operations. Our method avoids the mesh operations by reconstructing the mesh surfaces at each time step.

2.2 Interface Reconstruction

Bloomenthal and Ferguson [20] proposed one of the first approaches for generating non-manifold meshes defined by multiple regions of space. Balsys and Suffern [21] proposed an adaptive polygonization of non-manifold implicit surfaces with octree subdivision. Hege et al. [22] extended the basic marching cubes algorithm by allowing multiple vertex classes with an automatic method for generating topologically correct triangulations similar as our polygonization procedure but with larger stencils for up to only three classes. Bertram et al. [23] generated a quadrilateral for every edge connecting voxels with two different materials on a dual grid composed of voxels. Reitinger et al. [24] made use of domain subdivision to construct non-manifold meshes from multi-labeled volumetric datasets. Wu and Sullivan [25] 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 suffer from stair-stepped artifacts. The artifacts may be removed with post processing, such as, smoothing. However, it may also smear out detailed features. Our polygonization procedure takes all the information of the distance and indicator into consideration and improves the accuracy of intersection computation with bisection. To make it fast enough for simulation, we design a new set of stencils which is suitable for the polygonization of arbitrary number of phases.

Dey et al. [26] applied a recent *Delaunay refinement* algorithm to generate high quality triangular interface surfaces. Bronson et al. [27] introduced a new algorithm for generating tetrahedral meshes that conform to volumetric domains of multiple materials. Saye and Sethian [6], [28] extracted the multiphase interfaces with piecewise linear interpolation and further quality-improved under the framework of VIIM, which requires the continuous piecewise interpolation of every distance function and mesh abstraction and chopping. Ju et al. [29] presented a *dual contouring* method for Hermite data, which might also be used for multiphase contouring. Anderson et al. [30], [31] addressed the *Material Interface Reconstruction* problem in the standard VOF method.

3 Overview

Since our method is built on the VIIM and SLC, we first briefly review these two methods.

The VIIM applies an unsigned distance function together with a material indicator function to track the entire multiphase system on a regular Eulerian grid. Instead of tracking the distance and indicator function directly, VIIM tracks the movement of ϵ-surface and reconstructs the multiphase interface as the Voronoi interface of the ϵ-surface. VIIM handles multiphase interfaces with a large number of phases easily and handles the topological changes automatically without special treatments. The main drawback of VIIM is the loss of volume and thin features. Besides, it's also difficult to be made adaptive. The uniform representation of multiphase system used in VIIM greatly inspired our work.

The SLC was developed under the framework of semi-Lagrangian level set method. Several intelligent modifications were made to improve the accuracy. It maintains an explicit mesh that defines the surface and replaces the interpolation of distance function with exact distance computation, which eliminates substantial interpolation errors. To make the computation efficient, an octree data structure is carefully designed to provide fast retrieval of the mesh and a means for approximating the signed distance. Additionally, SLC uses bisection to locate the intersections along the edges with exact evaluation of the distance function. SLC is excellent in preserving surface features and volumes. However, SLC only works for two-phase problem, so we improve it to handle multiphase problem. For more details

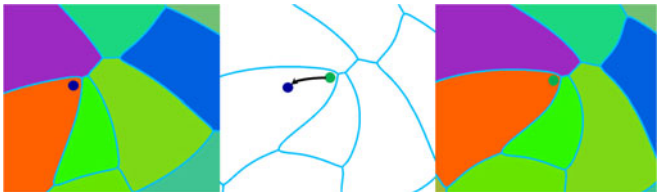

Fig. 1. Update of the distance and indicator functions. Middle: For a point (green) in space, we trace back a step to find its previous position (blue). Left: Distance and indicator functions are then updated by exact computation from the interface meshes in previous position. Right: Once the distance and indicator functions are updated, we process to construct a new interface mesh with the contouring algorithm.

of the development of SLC, please refer to the theoretical work of Strain [32] and Bargteil et al. [7]

Now we outline our method, formally. Given an N-phase problem (see Fig. 1) equipped with an unsigned distance function $\phi(\mathbf{x})$, which denotes the distance measured from $\mathbf{x}$ to the nearest point on the interface, an indicator function $\chi(\mathbf{x})$, which indicates the phase material, and an explicit surface mesh, we formulate the tracking of multiphase interfaces as a contouring problem which consists of three steps as follows:

1) Update ϕ and χ using semi-Lagrangian method. ϕ and χ are computed exactly to improve the accuracy.
2) Contour ϕ and χ to get the new interface meshes. This is performed with precomputed stencils combined with bisection to get accurate estimations of intersections and junctions.
3) Redistance ϕ and χ. As ϕ is most accurate near the interface, we propagate the values near the interface to the whole domain to stay in good accuracy.

To advect $\phi(\mathbf{x})$ and $\chi(\mathbf{x})$ forward, we use the semi-Lagrangian method [33] to backtrace the point $\mathbf{x}$ through the streamline of the velocity field $\mathbf{v}$ over a time Δt and assign the new value with the value at its previous location, i.e., $\phi(\mathbf{x}) = \phi(\mathbf{x} - \Delta t\mathbf{v})$, $\chi(\mathbf{x}) = \chi(\mathbf{x} - \Delta t\mathbf{v})$. Here, $\phi(\mathbf{x} - \Delta t\mathbf{v})$ is computed from the explicit surface mesh at the previous time step, according to [7], and the calculation of $\chi(\mathbf{x} - \Delta t\mathbf{v})$ requires a special treatment to the representation of the surface meshes (Section 4). Then, we will reconstruct the surface meshes from ϕ and χ instead of advancing the meshes forward to avoid the complex remeshing during the topological changes. The key point of this part is to guarantee that the reconstructed surface meshes are watertight and smooth (Section 5). Finally, since it is expensive to track multiphase interfaces on a uniform grid, we use an octree (quadtree in 2D) to store ϕ and χ. We only maintain the finest cells near the interfaces. As long as the interfaces are moved, we will adjust the tree's structure to keep it balanced. Section 6 gives the details how to build the octree to well resolve ϕ and χ and how to compute ϕ and χ for the tree structure efficiently.

4 Distance and Indicator Computation

Our multiphase system consists of two parts: an implicit part, which contains an unsigned distance function ϕ as well as an indicator function χ, and an explicit part, which corresponds to the surface meshes. Since the implicit part is similar to the one in VIIM, we only discuss the explicit part here.

To avoid complicated remeshing operations, we will always reconstruct the surface meshes from the isocontour $\phi(\mathbf{x}) = 0$, from which both the distance function and the indicator function can be updated accurately. The explicit representation of the surface mesh is similar to [7], except that the triangle mesh in our method may be non-manifold (one edge is shared by more than two triangles) and contains complex structures such as the T-junctions. Then, it raises a question: how can we identify a material at any point $\mathbf{x}$ without using a binary inside/outside classification of space? In our implementation, we follow Da and the colleagues' work [34] to assign a unique integer label to each material and apply different labels to the front and back of each triangle. Besides, we store an extra normal on each face of the triangle mesh to facilitate the calculation of the indicator function. The normal will be always pointing from a higher material label to a lower one. Now, we will show how such information is used to divide the entire problem domain into separate regions of materials.

4.1 Material Identification

Since the distance function can be computed according to [7], we only discuss how to get the indicator function here. We first consider a 2D case. For an arbitrary point $\mathbf{x}$, we denote its closest point on the curve as $\mathbf{y}$ and the corresponding normal as $\mathbf{n}(\mathbf{y})$, which is assumed to point from material $\mathbf{A}$ to $\mathbf{B}$. To identify the material label for $\mathbf{x}$, we first calculate the following quantity:

$$\mathcal{E} = (\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}). \tag{1}$$

If $\mathbf{y}$ lies inside the edge, the material label can easily be determined from the sign of $\mathcal{E}$. More specifically, $\mathcal{E} > 0$ means the point $\mathbf{x}$ belongs to material $\mathbf{B}$ while $\mathcal{E} < 0$ means it belongs to material $\mathbf{A}$. However, if the point $\mathbf{y}$ lies on the vertex, different values of $\mathcal{E}$ can be obtained if we use the normal defined on different edges. We remove this ambiguity by choosing the one with the maximum value of $|\mathcal{E}|$. That is because the point $\mathbf{y}$ can only lie on a vertex if it is located inside the Monge cone formed by two adjacent edges with an angle no less than 180 degrees (see Fig. 2).

The method above can be generalized to the 3D case where the segment is now a triangle and the vertex is a triangle edge. $\mathbf{y}$ is the closest point on the surface. When $\mathbf{y}$ lies strictly inside a triangle, we compute the indicator value using the sign of $\mathcal{E}$ from Equation (1). When $\mathbf{y}$ lies on the edge, we compute the indicator value using the maximum $|\mathcal{E}|$. In both cases, non-manifold meshes can also be handled properly as the 2D cases.

However, when $\mathbf{y}$ lies on the vertex, simply using the numerically greatest $|\mathcal{E}|$ would easily identify vertex with wrong indicator in case of very sharp corner, leading to regions of spurious interfaces around the vertex [35]. Thus we need a new method to compute the indicator value for such case. Baerentzen and Aanas [35] provided an exact arithmetic signing procedure by computing an angle-weighted pseudonormal for each vertex and use these pseudonormals to decide the sign. We adopt their method here by precomputing pseudonormals for mesh vertices and use the pseudonormals to compute the indicator value. This works for most two-manifold vertices. However, when

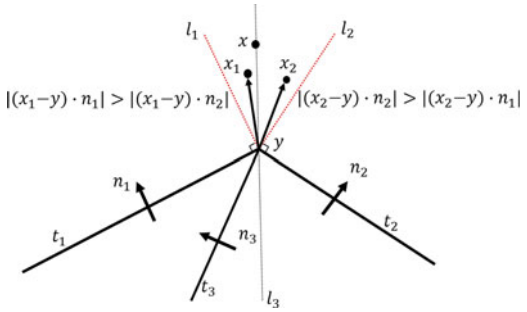

Fig. 2. Indicator computation near a vertex. Only when two neighboring segments form an angle no less than 180 degrees, the closest point $\mathbf{y}$ can lie on the vertex. In such case, $\mathbf{x}$ must lie in the region form by l_1 and l_2. l_1 and l_2 are perpendicular to t_1 and t_2 respectively. For $\mathbf{x}_1$, $\mathbf{n}_1$ is used to compute the indicator value while $\mathbf{n}_2$ is used for $\mathbf{x}_2$ according to our method. In a non-manifold setting, there can be a third segment incident on a vertex, say t_3. Imagine rotating t_1 and t_2 about vertex $\mathbf{y}$ to t_3, we can easily see that $|\mathcal{E}| = |(\mathbf{x}-\mathbf{y})\cdot\mathbf{n}_3|$ is always smaller than that of t_1 or t_2 as $|\mathcal{E}|$ keep decreasing to zero when rotating t_1 and t_2 to l_3 until one of them gets over t_3. Thus, t_1 or t_2 is always used to compute the indicator value accurately.

the vertex is on a non-manifold polyline, i.e., in the boundary of more than two phases, the pseudonormal is not well defined. In such a case, we compute multiple pseudonormals for that vertex, each corresponding to a phase (see Fig. 3). Then we check all these pseudonormals to decide which phase $\mathbf{x}$ is strictly inside. We only compute multiple pseudonormals when a vertex is queried the first time and store them for later queries. In practice, there is only a small amount of queries on such vertices, so the computational expense is negligible. We summarize the whole computational procedure for 3D in Algorithm 1. In implementation, we can keep track of the nearest point if it is on an edge and simply replace $\mathcal{E}$ if a numerically larger one is found and use the new one to decide the indicator value. We also record if the nearest point is on a vertex to avoid redundant computation.

The ability to exactly compute the distance function and indicator function is the significant difference of our method compared to previous methods. If the indicator computation is not accurate near the surfaces, we have to rely on ϵ-surface to construct the interface, as was done in [36]. As the distance function and indicator function are accurate, we can now process to contour them directly.

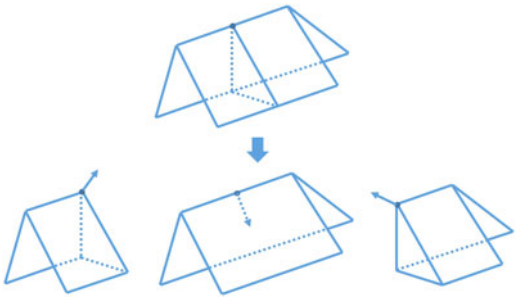

Fig. 3. An example of pseudonormals for a vertex on non-manifold polyline. The vertex lies in three phases and three pseudonormals are computed. For each phase, an angle-weighted psudonormal is computed from the normals of the vertex's incident faces that are also touching the phase. Before computing, the normals may need to be flipped so that they are ensured to be pointing out from the phase. In this way, each pseudonormal will be such oriented that it is pointing out from the corresponding phase.

Algorithm 1. Distance and Indicator Computation

T is the list of triangles, the number is N_T, $\mathbf{x}$ is the queried point, $\mathbf{y}$, $\phi(\mathbf{x})$, and $\chi(\mathbf{x})$ are to be computed.
$\phi(\mathbf{x}) \leftarrow \infty, \chi(\mathbf{x}) \leftarrow -1, \mathcal{E} \leftarrow 0, e \leftarrow -1, v \leftarrow -1$
for $i \leftarrow 1$ **to** N_T **do**
  $\mathbf{y}' = ClosestPoint(\mathbf{x}, T_i)$, $d = Distance(\mathbf{x}, \mathbf{y}')$
  $onEdge = ((edge = GetEdge(\mathbf{y}', T_i)) \neq -1)$
  $onVert = ((vert = GetVert(\mathbf{y}', T_i)) \neq -1)$
  $\mathbf{n}' = GetFaceNormal(T_i)$
  if $onVert$ && $vert = v$ **then**
    continue
  end if
  if $onEdge$ && $edge = e$ **then**
    $\mathcal{E}' \leftarrow (\mathbf{x}-\mathbf{y}')\cdot\mathbf{n}'$
    if $|\mathcal{E}'| > |\mathcal{E}|$ **then**
      $\mathcal{E} \leftarrow \mathcal{E}'$
      if $\mathcal{E} > 0$ **then**
        $\chi(\mathbf{x}) \leftarrow T_i \rightarrow \mathbf{B}$
      else
        $\chi(\mathbf{x}) \leftarrow T_i \rightarrow \mathbf{A}$
      end if
    end if
  else
    if $d < \phi(\mathbf{x})$ **then**
      $\phi(\mathbf{x}) \leftarrow d, \mathbf{y} \leftarrow \mathbf{y}'$
      if $onVert$ **then**
        $\chi(\mathbf{x}) = ComputeIndicator(vert)$
        $v \leftarrow vert, \mathcal{E} \leftarrow 0, e \leftarrow -1$
      else
        $\mathcal{E} \leftarrow (\mathbf{x}-\mathbf{y}')\cdot\mathbf{n}'$
        if $\mathcal{E} > 0$ **then**
          $\chi(\mathbf{x}) \leftarrow T_i \rightarrow \mathbf{B}$
        else
          $\chi(\mathbf{x}) \leftarrow T_i \rightarrow \mathbf{A}$
        end if
        if $onEdge$ **then**
          $v \leftarrow -1, e \leftarrow edge$
        else
          $v \leftarrow -1, \mathcal{E} \leftarrow 0, e \leftarrow -1$
        end if
      end if
    end if
  end if
end for

Use $\mathcal{E}$ with maximum $|\mathcal{E}|$ to compute indicator.

compute indicator for vertex.

Closet point is on face or edge now. Use $\mathcal{E}$ to compute indicator and record the index.

5 Contouring

In a multiphase system, the contouring problem can be very complex due to the large number of phases. On one hand, designing the stencils for all the possible cases is difficult. On the other hand, the intersection evaluation between the surface meshes and the distance function may not be accurate enough, especially around multiple junctions as shown in Fig. 4.

5.1 Precomputed Stencils

To simplify the contouring process, we subdivide the cubes/squares into tetrahedrons/triangles with the Marching Tetrahedra (MT) algorithm [37], and perform the contouring on the tetrahedrons/triangles. Now, we only require a small number of stencils as Fig. 5 shows, i.e., three

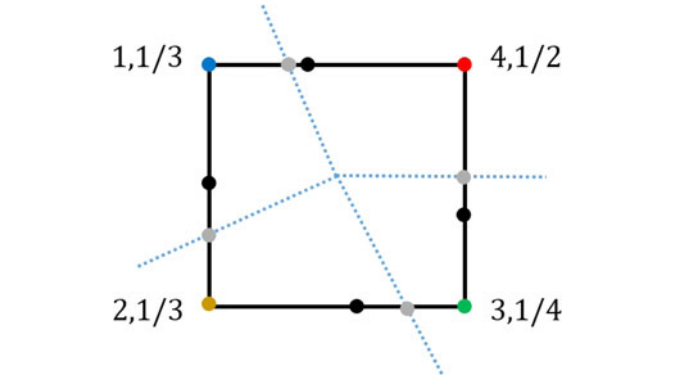

Fig. 4. An example showing the inaccuracies linear interpolation around a multiple junction. Vertices are annotated with indicators and distance values, i.e., (χ, ϕ). Linear interpolation incorrectly chooses the dark points as the zero crossing along the edges. The grey points are the actual zero crossing, which can be found with our bisection method.

stencils for the 2D case and five stencils for the 3D case. We use a triple (id_0, id_1, id_2) and a tetrad (id_0, id_1, id_2, id_3) to represent the material configuration of a triangle and a tetrahedron, respectively. During the reconstruction, we will apply different stencils according to the material configurations. Since there are no ambiguities in MT algorithm [38], the reconstructed surface meshes are guaranteed to be topologically consistent in the sense that triangles meet at junctions without overlap or gaps.

Note that we also adjust the order of the vertices of the generated triangles, which can be easily configured in the template codes, so that its normal points from a higher material label to a lower one as described in Section 4.

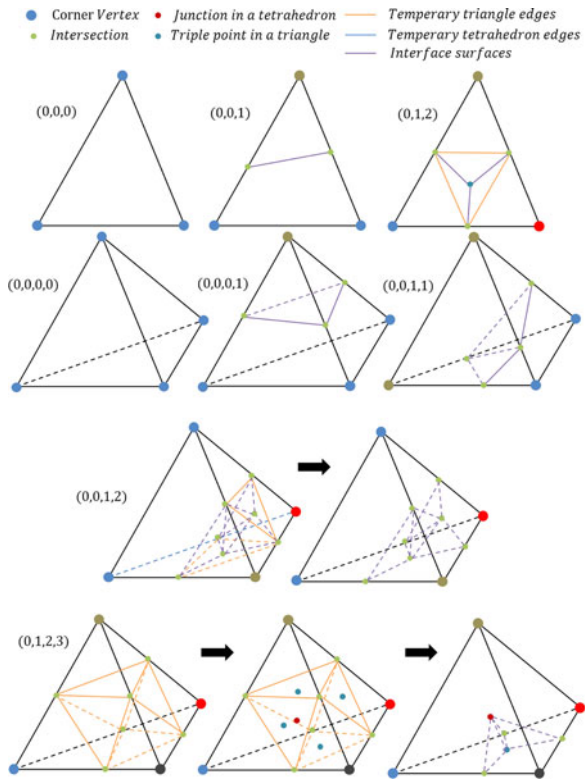

Fig. 5. Stencils for our multiphase polygonization. Note that in $(0, 1, 2, 3)$ we only show the interfaces connected to one of the faces of the tetrahedron for better illustration.

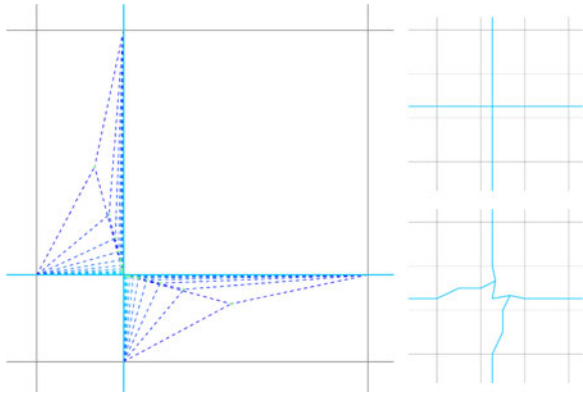

Fig. 6. Left: Bisection process inside a cell, showing how the triple points converge to the correct position. Right: Comparison of the results between bisection (top) and linear interpolation (bottom).

5.2 Intersection Computation

The key to a good contouring algorithm is an accurate evaluation of the intersection. A nice feature of SLC is that the intersections can be accurately located, with a simple bisection strategy [7], [32]. Thus, in the following context, we will improve SLC to make it suitable for use in a multiphase system.

We first briefly discuss how to calculate the intersection on an edge. Assume the two vertices of an edge are point A and B, the corresponding distances to the interface are ϕ_A, ϕ_B, and their indicators are χ_A and χ_B respectively. The intersection O on the edge $\overrightarrow{AB}$ can be linearly interpolated as: $O = \alpha A + (1-\alpha)B$, where $\alpha = \phi_B/(\phi_A + \phi_B)$. Then we apply the bisection strategy with the update equations:

$$\begin{array}{llll} A = O, & \phi_A = \phi_O, & \text{if} & \chi_A = \chi_O \\ B = O, & \phi_B = \phi_O, & \text{otherwise.} & \end{array}$$

The location of the intersection is iteratively adjusted until $|\phi_A - \phi_B|$ is smaller than some tolerance ϵ (i.e., $|\phi_A - \phi_B| < \epsilon$) or the maximum iteration number is reached.

To compute the triple point inside a triangle. We first compute three intersections on its edges, and assume that the triple point is inside the triangle formed by these three intersection points (see upper right of Fig. 5). Then, we iteratively confine this triangle to a smaller one until it converges to a point, which is exactly the triple point. Let A, B, C be the three vertices of a triangle, ϕ_A, ϕ_B, ϕ_C be the corresponding distances, and χ_A, χ_B, χ_C be the indicators. The initial guess of the triple point is computed as: $O = (O_{AB} + O_{BC} + O_{AC})/3$, where O_{AB},O_{BC},O_{AC} are the intersection points on the edges $\overrightarrow{AB}$, $\overrightarrow{BC}$ and $\overrightarrow{AC}$. Once O is computed, one of A, B, C will be replaced when the indicators match (e.g., if $\phi_A = \phi_O$, A will be replaced by O). This procedure is iterated and the stopping criterion is similar as the edge bisection except that we test against the tolerance with the area of the triangle ABC (i.e., Area$(ABC) < \epsilon$). The intersections on the edges can be computed using bisection, but it can be quite time-consuming. In practice, linear estimation is used for computing edge intersections as it works well enough here. The left of Fig. 6 shows an example of the iteration progress inside a triangle.

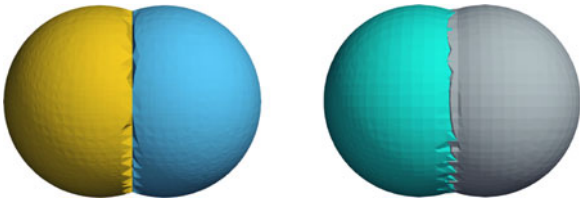

Fig. 7. A 3-phase example showing the comparison between bisection (left) and linear interpolation (right). Bisection produces smoother meshes than linear interpolation even with a fairly low resolution of 32^3.

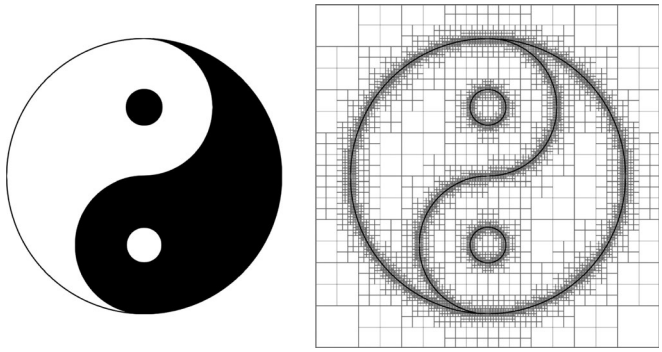

Fig. 8. A 2D multiphase distance tree (five phases). The multiphase distance tree is refined to the finest level near the interface and rapidly coarsened away from the interface.

As for computing junction points inside a tetrahedron, the bisection strategy works just in the same way. We first compute intersections on four triangles of the tetrahedron, and take the average of these intersections as the guess of the junction point, which then replaces one of the vertices of the tetrahedron with a same indicator. The tetrahedron is thus confined to a smaller one and finally converges to the junction point.

Figs. 6 and 7 gives two examples to illustrate the advantage of using a bisection method over a linear interpolation. In these examples, we assume there are no multiple intersections on the edge.

There might exist edges with multiple intersections anyway. In such cases, our method will find one of the multiple intersections. These effects can be mitigated by increasing the resolution (see Section 6).

6 Adaptive Refinement

To provide more details near the multiphase interfaces and guarantee the efficiency of the simulation, we introduce an adaptive structure based on the octree tree (quadtree in 2D) to store the discretized distance function ϕ and the indicator function χ. In the following discussion, we will refer the adaptive structure as a *multiphase distance* (MD) tree. The major difference between the MD tree and a standard distance tree [7] is an extra indicator χ on the tree. Both ϕ and χ are sampled on the corner vertices and the center of the cell of the MD tree. All cells also contain a list of triangles of the explicit mesh.

6.1 Computing ϕ and χ from the Tree

To compute $\phi(\mathbf{x})$ and $\chi(\mathbf{x})$, we first identify the leaf cell $\mathbf{C}$ in which the point $\mathbf{x}$ is located. If $\mathbf{C}$ is the finest cell, it means $\mathbf{x}$ is near the interface. We compute the exact distance and indicator by taking the point-triangle tests for all the triangles stored in $\mathbf{C}$ and its neighbors. Otherwise, all corners of the cell $\mathbf{C}$ will belong to the same material, in which case we only compute an approximate value of the distance by taking an interpolation from the corners and assign $\chi(\mathbf{x})$ with the value stored on center of $\mathbf{C}$.

Evaluation on a cell typically costs $O(1)$ work. For interpolation, this is obvious. For the exact point-triangle test, as there are a bounded number of triangles to be tested for any nonempty cell, the cost is $O(1)$ as well. Therefore, the cost of the evaluation of ϕ and χ is dominated by the $O(\log N_T)$ cost of finding the cell $\mathbf{C}$ containing $\mathbf{x}$ when $N_T \to \infty$, where N_T is the total number of cells of the tree.

6.2 Building the Tree

Starting with the new distance function $\phi_{n+1}(\mathbf{x})$ and the indicator function $\chi_{n+1}(\mathbf{x})$, we build the tree from top to bottom by recursively splitting the cells. If the distance at a cell center is less than the edge length, we will split the cell. As ϕ_{n+1} may not be the minimum distance to the interface in this cell, the resulting tree may be unbalanced, which is difficult to contour. Strain [32] suggested to balance the tree by brute force or estimate a gradient bound γ for $||\nabla\phi||$ and multiply the edge length by γ. The latter approach was used in [7] with a constant of 3. However, we have found out that it is difficult to estimate the gradient bound properly in a multiphase system. Increasing the constant may be a

TABLE 1
Detailed Statistic of the First Step of Zalesak Disk Test in Fig. 16 and Enright Sphere Test in Fig. 17 under Different Resolutions

	Zalesak Disk					Enright Sphere				
Res(n^d)	32^3	64^3	128^3	256^3	512^3	32^3	64^3	128^3	256^3	512^3
Cells(N_T)	2,969	10,897	48,497	200,673	816,849	2,873	10,849	46,457	194,505	796,121
Intersect Cells(N)	502	2,074	8,556	34,868	140,168	535	2,276	9,401	38,494	155,011
N/N_T	0.169	0.190	0.176	0.174	0.172	0.186	0.210	0.202	0.200	0.195
Vertices	3,455	11,619	49,655	207,303	847,428	3,304	11,517	48,580	204,971	845,379
Building Tree	0.0298	0.1201	0.2905	1.1519	5.0307	0.0497	0.1819	0.4070	1.3957	6.0386
Contouring	0.0944	0.1632	0.5724	2.2793	9.1415	0.2596	0.4607	1.2177	4.9097	19.4096
Redistancing	0.0143	0.0320	0.2519	1.1170	5.7564	0.0245	0.0345	0.2094	1.0323	6.2804
Total Time	0.1386	0.3153	1.1148	4.5482	19.9286	0.3338	0.6771	1.8341	7.3377	31.7286

The numbers of cells and vertices and the timings (in second) for building/contouring/redistancing an MD tree are displayed. The numbers of cells and vertices match the predicted $O(N)$ estimation while the timings verify the predicted $O(N \log N)$ (or $O(n^{d-1}\log n)$) cost of our method.

TABLE 2
Accuracy Evaluation for Indicator Computation

		Zalesak Disk		Enright Sphere	
Res	h	accuracy	e_i	accuracy	e_i
32^3	$3.13e^{-2}$	99.9517%	1.30	99.9664%	1.00
64^3	$1.56e^{-2}$	99.9886%	1.19	99.9934%	0.86
128^3	$7.81e^{-}3$	99.9977%	0.73	99.9983%	0.65
256^3	$3.91e^{-3}$	99.9993%	0.27	99.9996%	0.48
512^3	$1.95e^{-3}$	99.9999%	0.09	99.9999%	0.14

h is the smallest grid size and ϕ_e is the distance value of the point with mismatched indicator. $e_i = |\phi_e/h|_{max}$ measures the maximum distance of the point with error indicator over grid size. We can see that the accuracy rates are all above 99.9 percent and all errors occurred near the interfaces.

solution while leading to too many unnecessary splits away from the interface.

As some indicator values on the corners of a cell are different if the interface intersects this cell, we proposed another split criterion: *split every cell that has at least two different indicator values on the corners*.

To sum up, we split every cell that fulfills any one of the following conditions:

$$\phi_{n+1}(\mathbf{v}_c) < 3h_c$$
$$\exists \chi_{n+1}(\mathbf{v}_i) \neq \chi_{n+1}(\mathbf{v}_j), i \neq j,$$

where $\mathbf{v}_c$ is the center, $\mathbf{v}_i$ the corner vertex and h_c the edge length of the cell. Though it is not guaranteed to produce a balanced tree for all cases, we have found it works well in practice as it always splits the cells near the interface. We did not apply any techniques to balance the tree in all our examples, though it can be done quickly with little extra expense.

In general, our criteria produce a tree whose cells coarsen very rapidly away from the interface so as to resolve the interface accurately at minimal cost. Fig. 8 shows a 2D multiphase distance tree built with our criteria.

6.3 Contouring the Tree

After the MD tree is built, the interface can only intersect with the finest cells according to our reconstruction rule. Therefore, we run the contouring on the leaf cells only. The reconstructed surface meshes are guaranteed to be watertight and topologically consistent, having no gaps or overlaps. During the contouring process, each triangle generated is assigned with two different material labels (the neighboring indicators, as described in Section 4) and attached to the cells that they intersect.

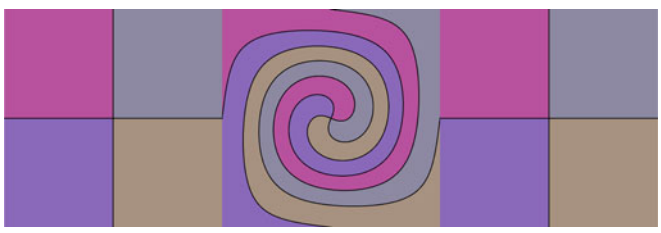

Fig. 9. The four phases were advected with a periodic velocity field and the interface recovered the initial position after a period.

TABLE 3
Here $H(t) = d(\Gamma_R, \Gamma_t)$ Measures the Hausdorff Difference between the Reference Interface and the Evolved Interface for Tests in Fig. 9

Res	$H(T)$	$\frac{H(T)}{h}$	rate	$H(2T)$	$\frac{H(2T)}{h}$	rate
32^2	$3.40e^{-2}$	1.09	-	$2.25e^{-2}$	0.72	-
64^2	$1.25e^{-2}$	0.80	1.44	$1.48e^{-2}$	0.94	0.60
128^2	$4.48e^{-3}$	0.57	1.48	$5.50e^{-3}$	0.70	1.42
256^2	$2.33e^{-3}$	0.60	0.94	$2.82e^{-3}$	0.72	0.96
512^2	$1.12e^{-3}$	0.57	1.05	$1.21e^{-3}$	0.62	1.23
$1{,}024^2$	$4.67e^{-4}$	0.48	1.26	$6.15e^{-4}$	0.63	0.97

6.4 Redistancing the Tree

Once the contouring is finished, we perform a fast marching method [39] to get the exact distances for the vertices of each cell. We first compute the distances for the vertices of the finest cells that intersect with the surface meshes and extend the values to the outer cells. Indicator values remain unchanged in this process since they are accurate already.

7 Results and Discussions

All examples were taken on an Intel 2.93 GHz PC with 12 GB memory in a single thread. The method ran pretty fast as it's fully adaptive. The interface tracking time ranged from a few milliseconds to several seconds for 2D examples when the resolutions ranged from 32^2 to $2{,}048^2$. For 3D examples, it took a few seconds to several minutes with an effective resolution up to 512^3. The renderings were performed with POV-Ray [40] and Mitsuba [41].

7.1 Efficiency and Stability

Here we analyze the complexity of our method to show its efficiency. As noted in Section 6.2, our splitting criteria produce a tree which coarsens so rapidly away from the interface that if there are N cells intersecting the interface, the entire tree contains only $O(N)$ cells and locating a cell **C** for

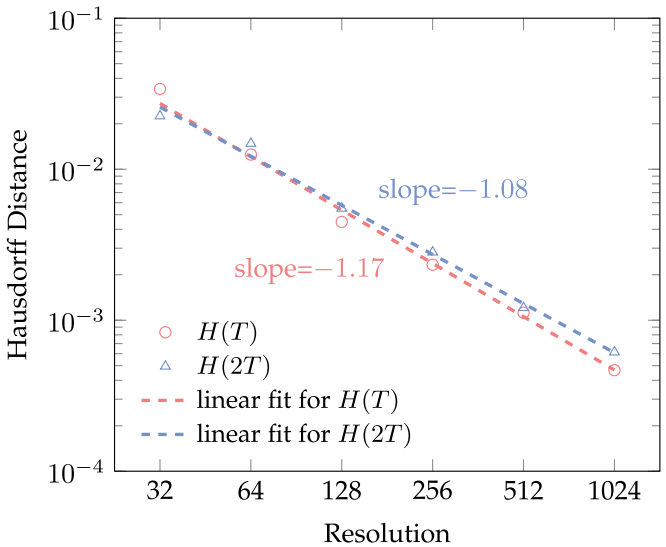

Fig. 10. Hausdorff distance versus resolution in Table 3. The results show a first-order convergence.

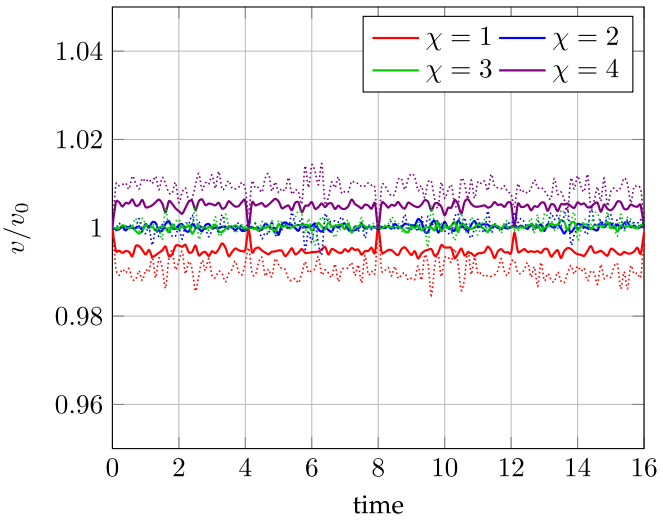

Fig. 11. Volume Percents of each phase relative to the initial volumes for the test of Fig. 9 in four periods with an effective resolution of 256^2 (solid lines) and 128^2 (dotted lines).

a point $\mathbf{x}$ costs $O(\log N)$ work. We first exam the cost of building an MD tree. When a new cell is added, we need to evaluate the distance and indicator functions for its vertices, which cost $O(\log N)$ as indicated in Section 6.1. Therefore, building an MD tree containing $O(N)$ cells costs $O(N \log N)$.

To contour an MD tree, we only need to process cells intersecting the interface, which costs $O(N)$ work. However, as we use bisection to locate the intersections, we need to evaluate ϕ and χ several times for each intersection computation. Each query on the tree costs $O(\log N)$ work, thus the total cost for contouring is $O(N \log N)$.

To redistance an MD tree, we first compute exact distance and indicator values for the finest cells intersecting the interface in $O(N)$ time. As the tree has $O(N)$ cells, the number of vertices is $O(N)$. For the remaining vertices, we run a fast-marching method, which costs $O(N \log N)$ [39].

In summary, our method runs in $O(N \log N)$ time, which mainly depends on the size of the interface. Table 1 verifies the predicted $O(N \log N)$ cost by taking the detailed statistic of the first step of Zalesak Disk test in Fig. 16 and Enright Sphere test in Fig. 17 under different resolutions. The last column of Table 4 also demonstrates the complexity of $O(N \log N)$ in 2D case.

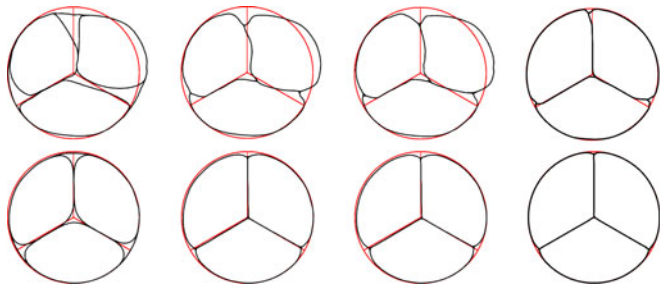

Fig. 13. Comparison of final interfaces (black) to analytic interfaces (red) for test of Fig. 12. From left to right: MLS, MLS-MBO, MLS-LSSF, and ours. (128^2).

Consider a uniform mesh in $\mathbf{R}^d$ with n^d points, a typical interface would intersect with $O(n^{d-1})$ cells, leading to $O(n^{d-1})$ interface elements. Direct evaluation of ϕ and χ on a uniform mesh costs $O(n^{2d-1})$ work while the complexity of our method is $O(n^{d-1} \log n)$. Thus our method is highly efficient in practice.

The stability of semi-Lagrangian method is excellent. Since new values are interpolated values from last time step, unconditional stability is guaranteed in any norm where the interpolation does not increase norms [7]. In our framework, we change the interpolation to exact distance and indicator computation. These changes only improve the accuracy and do not affect the unconditional stability. It should be noted that this is an inherent intrinsic characteristic in semi-Lagrangian method. Therefore, our method does not restrict the time step and arbitrarily large time step can be taken given a perfect path-tracer. In practice, we require the time step to satisfy the CFL condition and consider the errors in path tracing. We used a second-order Runge-Kutta scheme for path tracing and a CFL number of 1 for our fluid simulations while much larger time steps can be used for the passive advection tests.

7.2 Accuracy Evaluation

As the distance computation is accurate and was used in two-phase problem in [7], we just evaluate the accuracy of our method for indicator computation. We ran two tests on the Zalesk Disk in Fig. 16 and Enright Sphere in the top row of Fig. 17. In each test, the tree was initially built with exact functions to evaluate distance and indicator. Then we

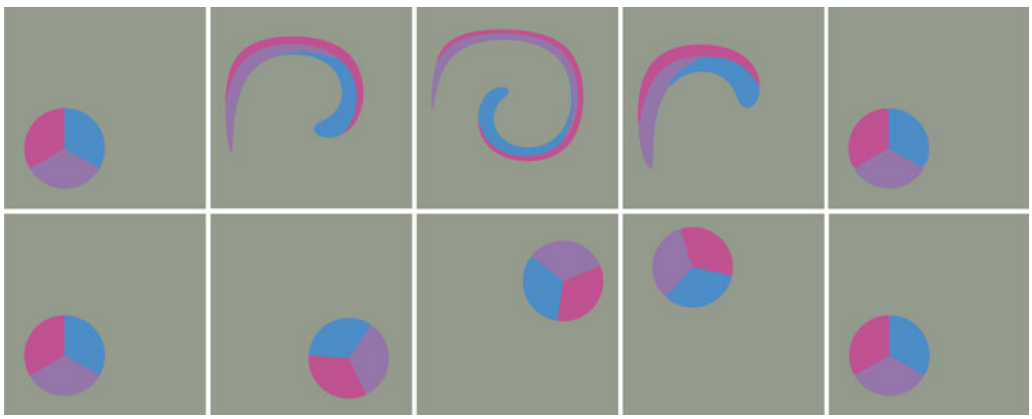

Fig. 12. The four phases were advected with a periodic stretching velocity field (top) and a rigid rotating velocity field (bottom) with our method. (256^2).

TABLE 4
Average Timings (sec/step) for the Stretching Tests of Fig. 12

Res(n^d)	MLS	MLS-MBO	MLS-LSSF	Ours
128^2	0.038	0.045	0.046	0.1
256^2	0.139	0.185	0.189	0.208
512^2	0.579	0.742	0.722	0.468
$1{,}024^2$	2.243	2.840	2.782	0.936
$2{,}048^2$	9.304	11.93	11.944	2.057

The Complexity of MLS is $O(n^d)$ while Ours is $O(n^{d-1}\log n)$.

randomly selected 10,000,000 points inside the domain. For each point, the indicator value was computed from the tree and compared to the exact value. Each test was run three times and the average data were taken. Table 2 summarizes the accuracy rate of these tests on different resolutions. At least 99.9 percent accuracy rates were achieved for all the tests and the rate was even higher as resolution increased, which was believed to be more accurate than methods based on interpolation. We tried to locate the source of the errors by comparing the distance value to the smallest grid size and found that they almost all occurred near the interface (less than a grid size). When the point lies on the interface, by our computation, either indicators of the face will be returned. This would not cause any problems as the distance value is zero. Another source of errors might be the numerical errors due to the floating point precision which can be addressed using robust floating point techniques. In practice, this may introduce very small artifacts close to the sharp corners of the interface, which is unnoticeable in sufficient resolution. It did not cause any stability issues in all our experiments.

To evaluate the capacity of conserving the interface surface details and volume of individual phase, we ran a test of four-phase advect problem in 2D. The setup of the test was quite simple, that we equally divided the domain $[0,1]^2$ into four regions, each corresponding to a phase (see Fig. 9). We built the multiphase interface and advected it with a periodic velocity field using $T = 4$:

$$\begin{aligned} v_0(\mathbf{x},t) &= -cos(\pi t/T)sin^2(\pi x_0)cos(2\pi x_1) \\ v_1(\mathbf{x},t) &= cos(\pi t/T)sin^2(\pi x_1)cos(2\pi x_0). \end{aligned} \quad (2)$$

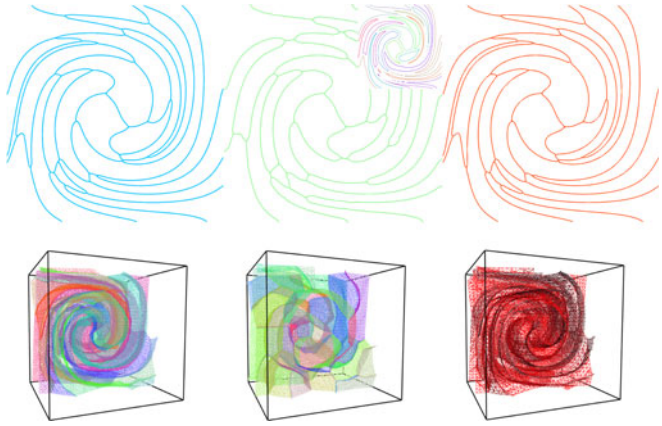

Fig. 14. Comparison of our method with the VIIM. From left to right: Interfaces created by our method, the VIIM (and the ϵ-surface, $\epsilon = 2\,\mathrm{h}$), and the reference interfaces.

After a whole period of evolution, the interface recovered its initial position. We computed the Hausdorff distance [42] between the interface with the exact reference interface to measure how well the method maintained the shapes of the interface. The data were collected in Table 3 and plotted in Fig. 10. We can see that most of the differences are less than one grid size and the results show a first-order convergence. We also measured the volume changes of each phase by computing the exact volume percentages relative to the initial volumes during the evolution (see Fig. 11). The maximum volume error relative to the initial volume was about 1 percent for a resolution of 128^2 and 0.5 percent for 256^2. The errors did not appear to accumulate in time.

7.3 Comparison

Inspired by [14], we first designed a three-phase circle, and advected it with a stretching velocity field in Equation (2) and a rigid rotating field. Fig. 12 shows the results after a period of stretching and rotation. The results were compared to the analytic interfaces for our method and several previous methods (in the semi-Lagrangian framework), including the multiple level set methods without reinitialization (MLS), with MBO reinitialization (MLS-MBO) [11], and with LSSF reinitialization (MLS-LSSF) [12] in Fig. 13. Timings for each method were summarized in Table 4.

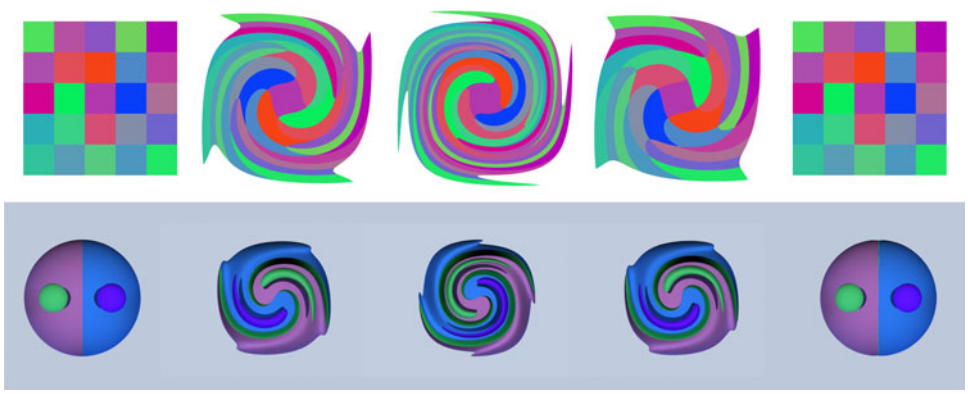

Fig. 15. Advection of a 5×5 color checker (top, 512^2) and a scene with two semi-spheres and two spheres inside (bottom, cutaway view, 256^3). Our method faithfully tracked the thin features and highly conserved the volumes and shapes, even under extreme stretching.

Fig. 16. Rotation test of a 3-phase Zalesak Disk. (5 s/step, 256^3).

We can see that our method outperforms the MLS methods and is very good at conserving the volumes and shapes. It is easy to verify that the complexity of MLS is $O(n^d)$ while ours is $O(n^{d-1}\log n)$, so our method also runs faster than MLS. It should be noted that MLS might be made spatially adaptive or improved with the narrow band method, though the projection step requires some special efforts. In such case, a similar complexity can be expected for the modified MLS. However, as we use only a distance function and an indicator function to represent the multiphase system, our method requires much less memory.

We also compared our method to the VIIM. One big advantage of our method is that we can capture thin details easily while VIIM quickly losses the details and tends to round off the sharp corners as it depends on the ϵ-surface to construct the interface. Fig. 14 shows some examples of random Voronoi regions moving under an external agitator. When the ϵ-surface of any phase vanished, the VIIM produced incorrect interfaces immediately, while our method preserved the thin regions well and produced interfaces which highly matched the reference interfaces.

It should be noted that our method deals with topological changes implicitly by recontouring every step, thus its implementation is much simpler to than the explicit method, such as [18], [43].

7.4 Passive Advection

We will show more examples of passive advection in this section. Fig. 15 shows two challenging cases of advection. We designed a color checker and a scene with nested spheres. In both cases, the interfaces were driven under extreme periodic stretching. The thin features were captured well during the evolution and the shapes and volumes were both faithfully conserved after several periods. Note that some accumulated wrinklings appear near the triple lines in the latter case since this is a purely passive flow. Fig. 16 shows the results of a 3-phase Zalesak Disk test and Fig. 17 shows the Enright tests on two spheres, subdivided into three and six phases, respectively. For more advection tests, please refer to the supplementary video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2015.2476788<.

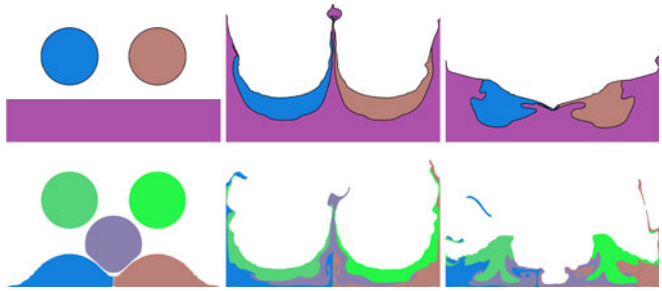

Fig. 18. 2D inviscid fluid simulations. Our interface tracker was able to capture extremely thin details easily. (256^2).

For all these tests, our method exhibited little smoothing on sharp features and was able to capture thin surface details faithfully and conserve the volume of each individual phase well. It should be noted that slight offsets near the triple line (or triple point in 2D) may develop during the evolution as the accurate computation of intersections also rely on an exact evaluation of the velocity field. While further research into remeshing strategies may reduce these effects, reducing the time step is also an alternative.

7.5 Fluid Simulation

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

We first consider the inviscid fluid simulation. Since the inviscid flow can be turbulent, the capture of multiphase interfaces is highly challenging. Fig. 18 shows two

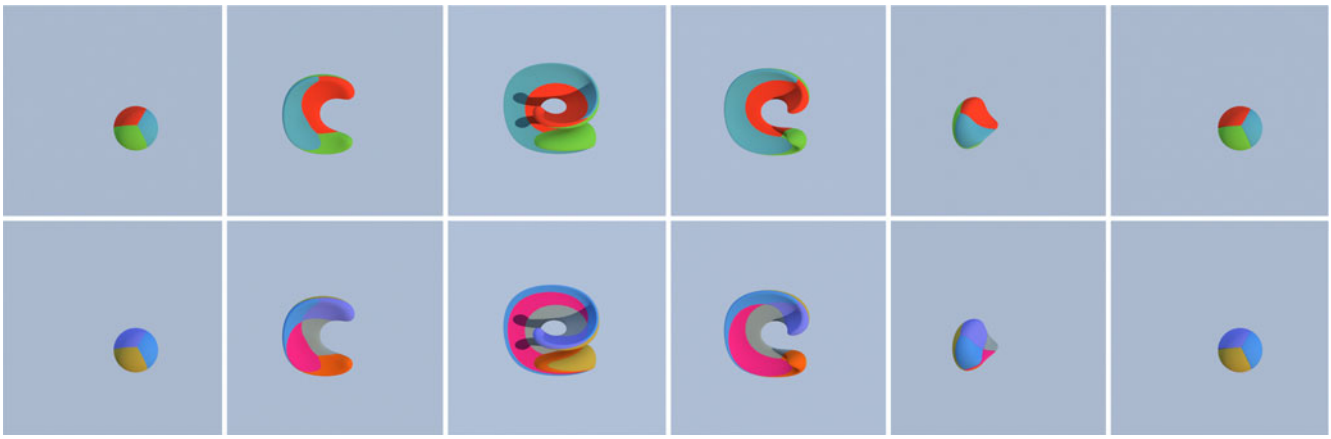

Fig. 17. Enright tests of multiphase spheres. Our method captured the thin regions well in spite of the extreme stretching. (50 and 90 s/step, 512^3).

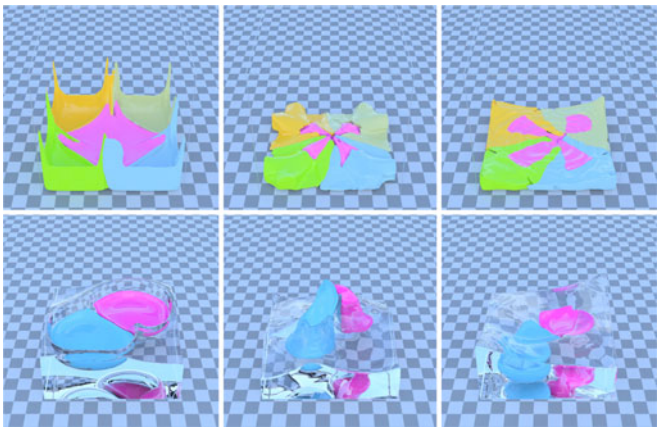

Fig. 19. Inviscid fluid simulations of five balls colliding to form complex interfaces (top) and two balls fell into a pool (bottom). Splashing details were captured and the complex interfaces were tracked well. (256^3).

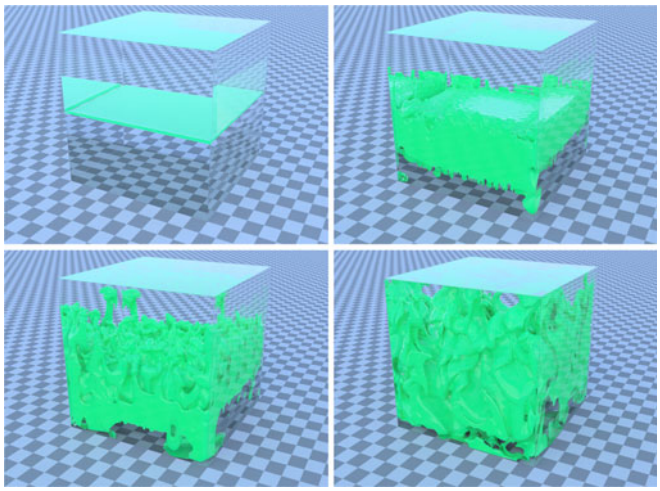

Fig. 21. Rayleigh-Taylor instability simulation. Our interface tracker was able to capture the complex interfaces. (128^3).

examples on 2D inviscid fluid simulation, from which we can see that our tracker captured highly detailed thin features. The top row of Fig. 19 shows the results of falling and colliding of five balls, which created high splashes and formed complex interfaces while the bottom row shows the results of dropping two balls into a pool. Fig. 20 shows a simulation of a dambreak scene. Two dams of different phases were released to collapse with each other and form detailed thin sheets. Our method needs no special treatment for these simulations, though the velocity fields were quite turbulent.

Fig. 21 shows two layers of fluid where the bottom layer fluid was lighter than the top layer fluid, causing the Rayleigh-Taylor instability when two layers of fluid switched places. Though it was an extremely challenging case, our tracker was able to track many thin features with a resolution of only 128^3.

We also coupled our interface tracker with a viscous fluid simulator [44] in Fig. 22 and a viscoelastic fluid simulator [45] in Fig. 23. As our interface tracker computes the indicator function accurately, we can easily assign different physical parameters to each phase efficiently.

In most of these tests, the interface tracker (256^3) took significantly less time than the fluid solver, although the latter was run with a lower resolution. The interface tracker usually took less than a minute. In the viscous fluid simulation, the interface tracker could take up to 20 percent of the total computational time. For examples with a small number of phases or less complicated interfaces, the interface tracker only took a small fraction of the total computational time.

It should be noted that when the velocity field is tangentially discontinuous across the material interfaces, the triple lines may not be smooth due to the failure of bisection of locating a triple point. This is noticeable in Fig. 19. Increasing the resolution or reducing the time step might alleviate these effects, but cannot remove them completely. Since this is an inherent issue of coupling the interface tracker with a low resolution fluid simulator, which might be solved with strong coupling strategy and we will leave it to future research. Further smoothing strategies [36] may be applied as a post-processing step if necessary.

8 Conclusions and Future Work

We provide an elegant method for multiphase interface tracking, which is capable of handling topological changes automatically while conserving the surface details very

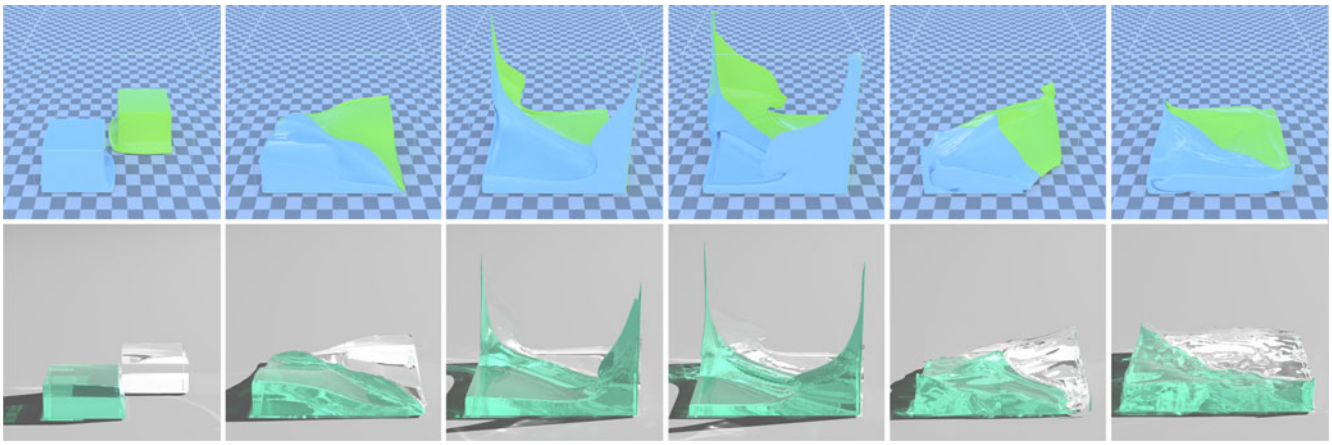

Fig. 20. Multiphase dambreak simulations. Two dams of different phases were released to collapse with each other to form detailed thin sheets. The interfaces were clearly captured by our method. (256^3).

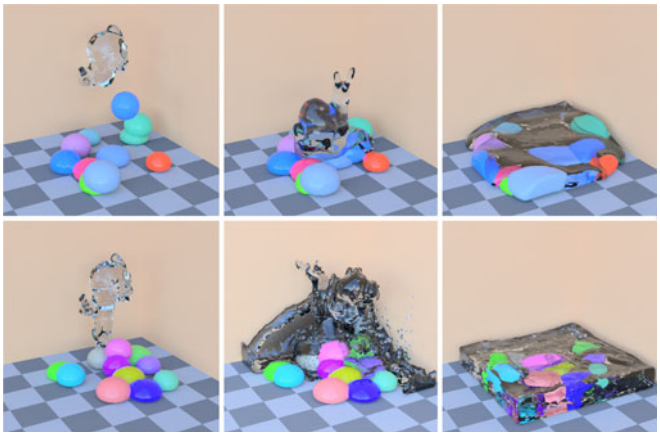

Fig. 22. Coupling of our interface tracker with a viscous fluid simulator. Viscous balls and bunnies with different viscosities are dropped together into a pile under the gravity. The bottom one is simulated with smaller viscosities, which created some splashes and droplets. (256^3).

well. Our experiments show that it is effective, efficient, and robust. It is also very flexible and easy to be applied in multiphase simulations.

There are a few limitations of our current implementation, which need to be addressed in our future. Multiple intersections along a cell edge cannot be triangulated exactly by our stencils currently. Increasing the resolution may help as the error is limited within a cell. Our contouring could create poor shape triangles, which might be improved with further remeshing. When the velocity field is not smooth, the bisection may fail and lead to some artifacts. Volume losses can still happen as surface features can easily shrink below the resolution of the solver. These effects were caused by the weak coupling between the solver and the interface tracker. Further research into a strong coupling scheme will help to remove the artifacts and conserve the volumes. Techniques, such as thickening scheme [46], might also be adopted to overcome the volume loss though modifications regarding multiphase are required.

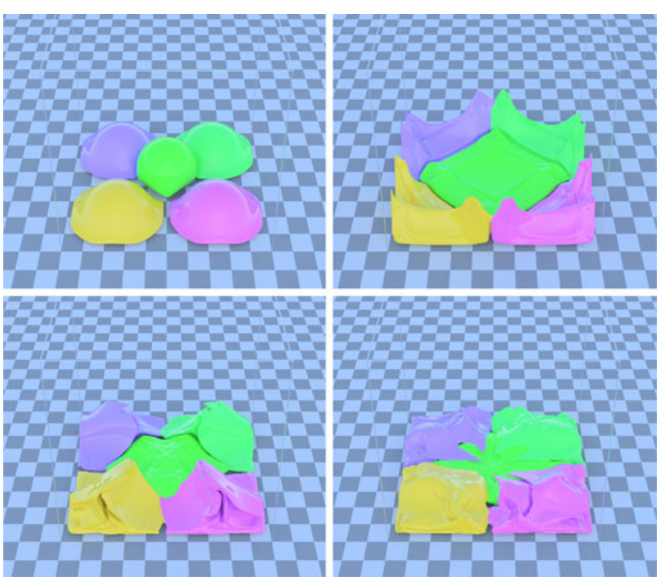

Fig. 23. Coupling of our interface tracker with a viscoelastic fluid simulator. Note the interesting surface details that were captured. (256^3).

Acknowledgments

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 the National Natural Science Foundation (NSF) of China under Grant No. 61272326, No. 51475394 and No. 6140051239, and by Grants of University of Macau ((MYRG202(Y1-L4)-FST11-WEH), (MYRG2014-00139-FST)), the Project of Macao Science and Technology Development Fund (FDCT) "Adaptive Approach in Fluid Dynamic Simulation", and EU FP7 under grant agreement [612627-"AniNex"], BIS of the British Government and Ministry of Education of China (Sino-UK Higher Education Research Partnership for PhD Studies Project).

References

[1] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan, "A front-tracking method for the computations of multiphase flow," *J. Comput. Phys.*, vol. 169, no. 2, pp. 708–759, 2001.

[2] C. Hirt and B. Nichols, "Volume of fluid (vof) method for the dynamics of free boundaries," *J. Comput. Phys.*, vol. 39, no. 1, pp. 201–225, 1981.

[3] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, Nov. 1988.

[4] R. I. Saye and J. A. Sethian, "The Voronoi implicit interface method for computing multiphase physics," in *Proc. Nat. Acad. Sci.*, vol. 108, no. 49, pp. 19 498–19 503, 2011.

[5] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces* (Series Applied Mathematical Sciences), 2003rd ed. New York, NY, USA: Springer, Nov. 2002.

[6] R. Saye and J. Sethian, "Analysis and applications of the Voronoi implicit interface method," *J. Comput. Phys.*, vol. 231, no. 18, pp. 6051–6085, 2012.

[7] A. W. Bargteil, T. G. Goktekin, J. F. O'brien, and J. A. Strain, "A semi-lagrangian contouring method for fluid simulation," *ACM Trans. Graph.*, vol. 25, no. 1, pp. 19–38, Jan. 2006.

[8] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.* 2001, pp. 23–30.

[9] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 736–744, Jul. 2002.

[10] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A hybrid particle level set method for improved interface capturing," *J. Comput. Phys.*, vol. 183, no. 1, pp. 83–116, Nov. 2002.

[11] B. Merriman, J. K. Bence, and S. J. Osher, "Motion of multiple junctions: A level set approach," *J. Comput. Phys.*, vol. 112, no. 2, pp. 334–363, Jun. 1994.

[12] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," in *Proc. ACM SIGGRAPH 2006 Papers*, 2006, pp. 812–819.

[13] X. Zhang, J.-S. Chen, and S. Osher, " A multiple level set method for modeling grain boundary evolution of polycrystalline materials," *Interaction Multiscale Mech.*, vol. 1, no. 2, pp. 191–209, Jun. 2008.

[14] D. P. Starinshak, S. Karni, and P. L. Roe, "A new level set model for multimaterial flows," *J. Comput. Phys.*, vol. 262, pp. 1–16, Apr. 2014.

[15] W. Zheng, J.-H. Yong, and J.-C. Paul, "Simulation of bubbles," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2006, pp. 325–333.

[16] 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, pp. 98:1–98:10, Jul. 2007.

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

[18] F. Da, C. Batty, and E. Grinspun, "Multimaterial mesh-based surface tracking," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 112:1–112:11, Jul. 2014.

[19] M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. B. Christensen, J. A. Baerentzen, and R. Bridson, "Multiphase flow of immiscible fluids on unstructured moving meshes," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2012, pp. 97–106.
[20] J. Bloomenthal and K. Ferguson, "Polygonization of non-manifold implicit surfaces," in *Proc. 22nd Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 309–316.
[21] R. J. Balsys and K. G. Suffern, "Adaptive Polygonisation of non-manifold implicit Surfaces," in *Proc. Int. Conf. Comput. Graph., Imaging Vis.*, 2005, pp. 257–263.
[22] H.-C. Hege, M. Seebass, D. Stalling, and M. Zöckler, " A generalized marching cubes algorithm based on non-binary classifications," ZIB, Takustr.7, 14195 Berlin, Germany, Tech. Rep. SC-97-05, 1997.
[23] M. Bertram, G. Reis, R. H. V. Lengen, S. Köhn, and H. Hagen, "Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart," in *Proc. Joint Eurographics / IEEE VGTC Conf. Vis.*, 2005, pp. 1–10.
[24] B. Reitinger, A. Bornik, and R. Beichel, "Constructing smooth non-manifold meshes of multi-labeled volumetric datasets," in *Proc. Int. Conf. Central Eur. Comput. Graph. Vis.*, 2005, pp. 227–234.
[25] Z. Wu and J. M. Sullivan, "Multiple material marching cubes algorithm," *Int. J. Numerical Methods Eng.*, vol. 58, no. 2, pp. 189–207, 2003.
[26] T. K. Dey, F. Janoos, and J. A. Levine, "Meshing interfaces of multi-label data with delaunay refinement," *Eng. Comput.*, vol. 28, no. 1, pp. 71–82, Jan. 2012.
[27] J. Bronson, J. Levine, and R. Whitaker, "Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality," in *Proc. 21st Int. Meshing Roundtable*, 2013, pp. 191–209.
[28] R. I. Saye, " An algorithm to mesh interconnected surfaces via the Voronoi interface," *Eng. Comput.*, vol. 31, no. 1, pp. 123–139, 2015.
[29] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in *Proc. 29th Annu. Conf. Comput. Graph. Interactive Techn.*, 2002, pp. 339–346.
[30] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy, "Discrete multi-material interface reconstruction for volume fraction data," *Comput. Graph. Forum*, vol. 27, no. 3, pp. 1015–1022, 2008.
[31] J. Anderson, C. Garth, M. Duchaineau, and K. Joy, "Smooth, volume-accurate material interface reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 5, pp. 802–814, Sep. 2010.
[32] J. Strain, "A fast modular semi-lagrangian method for moving interfaces," *J. Comput. Phys.*, vol. 161, no. 2, pp. 512–536, Jul. 2000.
[33] J. Stam, "Stable fluids," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 121–128.
[34] F. Da, C. Batty, and E. Grinspun, "Multimaterial front tracking," *CoRR*, vol. abs/1306.3113, 2013.
[35] J. A. Barentzen and H. Aanas, "Computing discrete signed distance fields from triangle meshes," Tech. Rep. Informatics Math. Modelling, Technical Univ. Denmark, Lyngby, Denmark, Tech. Rep. IMM-TR-2002-21, 2002.
[36] X. Li, X. He, X. Liu, B. Liu, and E. Wu, "Multiphase surface tracking with explicit contouring," in *Proc. 20th ACM Symp. Virtual Reality Softw. Technol.*, 2014, pp. 31–40.
[37] J. Bloomenthal, "An implicit surface polygonizer," in *Graphics Gems IV*. New York, NY, USA: Academic, 1994, pp. 324–349.
[38] A. Gueziec and R. Hummel, "Exploiting triangulated surface extraction using tetrahedral decomposition," *IEEE Trans. Vis. Comput. Graph.*, vol. 1, no. 4, pp. 328–342, Dec. 1995.
[39] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," in *Proc. Nat. Acad. Sci.*, vol. 93, no. 4, pp. 1591–1595, 1996.
[40] Persistence of Vision Pty. Ltd., Persistence of Vision Raytracer [Online]. Available: http://www.povray.org/, 2004.
[41] W. Jakob. (2010). Mitsuba renderer [Online]. Available: http://www.mitsuba-renderer.org
[42] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2002, vol. 1, pp. 705–708.
[43] M. K. Misztal and J. A. Baerentzen, "Topology-adaptive interface tracking using the deformable simplicial complex," *ACM Trans. Graph.*, vol. 31, no. 3, pp. 24:1–24:12, Jun. 2012.
[44] C. Batty and R. Bridson, "Accurate viscous free surfaces for buckling, coiling, and rotating liquids," in *Proc. ACM/Eurographics Symp. Comput. Animation*, Jul. 2008, pp. 219–228.
[45] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien, "A method for animating viscoelastic fluids," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 463–468, 2004.
[46] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2007, pp. 219–228.

Xiaosheng Li received the BSc degree from Sun Yat-Sen University, Guangzhou, China, in 2010. He is currently working toward the PhD degree at the State Key Laboratory of Computer Science, the Institute of Software, Chinese Academy of Sciences. His research interests include physically based simulation and rendering.

Xiaowei He received the BSc degree in information management and information system and the MSc degrees in biomedical engineering from Peking University, P.R. China, in 2008 and 2011, respectively. He is currently working toward the PhD degree in computer science at the Institute of Software, Chinese Academy of Sciences. His research interests include computer graphics and physical-based simulation.

Xuehui Liu received the BSc and MSc degrees from Xiang Tan University, Hunan, and received the PhD degree in 1998 from the Institute of Software, Chinese Academy of Sciences. Since then she has been working at the Institute of Software, Chinese Academy of Sciences, and is now an associate professor. Her research interests include realistic image synthesis, physically based modeling, and simulation.

Jian J. Zhang is a professor of computer graphics at the National Centre for Computer Animation, Bournemouth University, United Kingdom and leads the Computer Animation Research Centre. He is also a co-founder of the UK's Centre for Digital Entertainment, funded by the Engineering and Physical Sciences Research Council. His research focuses on a number of topics relating to 3D virtual human modelling, animation and simulation, including geometric modelling, rigging and skinning, motion synthesis, deformation and physics-based simulation. He is also interested in virtual reality and medical visualisation.

Baoquan Liu received the PhD degree in computer graphics in 2007 from the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He is a senior research fellow of computer graphics in the Department of Computer Science and Technology, University of Bedfordshire. His research interests are computer graphics, GPU computing, and visualization.

Enhua Wu recievd the BSc degree in 1970 from Tsinghua University, Beijing and the PhD degree from the University of Manchester, United Kingdom in 1984. He has been working at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences since 1985, and at the same time invited as a full professor of the University of Macau since 1997. He has been invited to chair or co-chair various conferences or program committees including Pacific Graphics 2005, CASA2006, ACM VRST2010, 2013, 2015, WSCG2012 and as a keynote speaker for CyberWorlds2006, ACM VRST2010, ACM VRCAI2011, WSCG2012 etc.. He is an associate editor-in-chief of the *Journal of Computer Science and Technology* since 1995, and the editorial board member of TVC, CAVW, IJIG, IJVR. His research interests include realistic image synthesis, physically based simulation, and virtual reality. He is a member of the IEEE & ACM, and a fellow of the China Computer Federation (CCF).