A Semi-Explicit Surface Tracking Mechanism for Multi-Phase Immiscible Liquids

Meng Yang, Juntao Ye, Frank Ding, Yubo Zhang, and Dong-Ming Yan

Abstract—We introduce a new method to efficiently track complex interfaces among multi-phase immiscible fluids. Unlike existing techniques, we use a mesh-based representation for global liquid surfaces while selectively modeling some local surficial regions with *regional level sets* (RLS) to handle complex geometries that are difficult to resolve with explicit topology operations. Such a semi-explicit surface mechanism can preserve volume, fine features and foam-like thin films under a relatively low computational expenditure. Our method processes the surface evolution by sampling the fluid domain onto a *spectrally refined grid* (SRG) and performs efficient grid scanning, generalized interpolations and topology operations on the basis of this grid structure. For the RLS surface part, we propose an accurate advection scheme targeted at SRG. For the explicit mesh part, we develop a fast grid-scanning technique to voxelize the meshes and introduce novel strategies to detect grid cells that contain inconsistent mesh components. A robust algorithm is proposed to construct consistent local meshes to resolve mesh penetrations, and handle the coupling between explicit mesh and RLS surficial regions. We also provide further improvement on handling complicated topological variations, and strategies for remeshing mesh/RLS interconversions.

Index Terms—Surface tracking, explicit mesh, remeshing, regional level set, multi-material, spectrally refined grid

◆

1 INTRODUCTION

SURFACE tracking is an important topic in computer graphics, especially in liquid simulation. Implicit surface mechanisms, such as the level set and volume-of-fluid methods, are widely used to model dynamic liquid surfaces. However, implicit surfaces require high-resolution underlying grid to preserve fine details, thereby imposing a heavy computational overhead on the simulation. Explicit surface tracking, which advances dynamic interfaces modeled by Lagrangian meshes, has attracted attention in recent years because explicit surfaces show the ability to preserve liquid mass/volume and subgrid-scale details. On average, the majority of surface areas in a liquid simulation are either free and smooth regions or liquid-solid boundary regions. Sharp geometries or thin features often appear as local phenomena. If an implicit surface representation is used, the computation-heavy advection and reinitialization steps will treat the overall surface regions equally, thereby consuming a large amount of computing power in regions that have little contribution to visually interesting details. Explicit surface tracking [31] helps solve this problem by evolving an initial mesh based on the underlying simulation. In this method, calculations can be focused on regions where important topological changes occur. Therefore, if the topology operations are achieved in an efficient way, explicit surfaces will show superiority in capturing fine features even under a relatively low mesh resolution. Several robust techniques, which handle scenarios from two-phase fluids to multi-phase materials, have been proposed for tracking explicit surfaces.

In a multi-phase liquid simulation, two situations are most often encountered. First, contacting/colliding liquid bodies of different materials do not merge together, but are separated by phase interfaces with double-, triple- or high-order junctions. Second, drastic interactions form local liquid blocks or air pockets trapped inside a surrounding liquid. Therefore, several thorny points arise when tracking multi-phase triangle meshes: (I) If a manifold mesh representation is employed, in the case of air pockets, the surface of the surrounding liquid will have closed mesh subsets with vertex normals pointing to the interior, and these surface "pockets" may cause large amount of computations, leading to possible divergence problems. For instance, for collision resolution based methods, a large amount of collision detections will be triggered due to the mesh proximities between penetrated liquids. (II) If a non-manifold mesh structure is used, such as the framework of [9], surface "pockets" can be eliminated and the junction problem can also be well resolved. However, a non-manifold structure means that numerous complex geometric configurations are to be considered; thus several complicated topology surgeries are needed to handle non-manifold merging, splitting, and mesh improvements. Furthermore, modern ray-tracers

---

- *M. Yang is with Amazon.com Inc., Seattle, WA 98109, and also work was done while M. Yang was with Institute of Automation, Chinese Academy of Sciences. E-mail: meyang@amazon.com.*
- *J. Ye and D.-M. Yan are with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China.*
  E-mail: juntao.ye@ia.ac.cn, yandongming@gmail.com.
- *F. Ding is with the Light Chaser Animation Inc., Chaoyang District, Beijing 100103, China. E-mail: wss.frankding@gmail.com.*
- *Y. Zhang is with nVidia, Santa Clara, CA 95051.*
  E-mail: ybzhang@ucdavis.edu.

Manuscript received 9 Dec. 2017; revised 4 Aug. 2018; accepted 4 Aug. 2018.
Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Juntao Ye.)
Recommended for acceptance by T. Ju.
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.2018.2864283

generally cannot handle non-manifold surfaces. Therefore non-manifold structures usually need further manipulation before rendering, or the renderer needs customization as conducted by Ishida et al. [18].

On the basis of the preceding observations, a new technique that robustly and efficiently tracks multi-phase liquid surfaces is introduced in this paper. We use explicit triangle meshes to model the global or vast majority of liquid surfaces, and these meshes are closed manifolds. In some local regions, we selectively model the surfaces with *regional level sets* (RLS) [37] to eliminate the surfaces "pockets" and handle some high-order junctions of tangled interfaces. Our method is called "semi-explicit", due to two reasons. First, we utilize not only Lagrangian meshes to preserve fine features of large area surfaces, but also the RLS scheme to easily model arbitrary thin liquid films among locally contacting fluid regions. Second, our method performs topology operations by constructing local triangle meshes. The main contributions are as follows:

- a combination of novel techniques to decompose surface regions and detect topological events,
- a new method to reconstruct local meshes and robustly handle topological changes and the mesh/RLS coupling,
- several practical strategies for advanced remeshing requirements and mesh/RLS interconversion, and
- a modest improvement to the advection scheme for RLS.

2 RELATED WORK

2.1 Surface Tracking in Fluid Simulation

Particle-Based Surfacing. Particles are widely used for fluid simulation due to their property of mass-conservation and flexibility in handling topological changes [34]. With the rise of the fluid-implicit-particle (FLIP) method [38], particle simulation followed by surface reconstruction has become a popular solution. The FLIP method provides a smooth surface but is prone to generating erroneous volumes of fluids among splashes and in concave regions. Adams et al. [1] improved their method by storing a signed distance field at each particle's position, which allowed the generation of a smoother surface with an irregular particle distribution and with particles of varying radii. However, updating the signed distance field becomes the new computational bottleneck. Some particle-based methods [28] reconstruct surfaces by assigning material properties to particles, but these methods are unsuitable for simulating persisting foams or films, and the surface quality is sensitive to particle noises near interfaces. Yu and Turk [35] used anisotropic kernels that generate a smooth surface faster than that of [1]. However, unlike the method in [38], their method takes more computation time and suffers from volume shrinkage. Ando et al. [2], in their adaptive liquid simulation framework, used the union of convex hulls built from groups of three particles to generate a smooth surface, at the expense of long computation time. Yu et al. [36] adapted the technique originally designed for explicit surface tracking [32], for particle-based simulations. Because tracking the interior of fluids using particles often causes divergence between explicit surfaces and particles during mesh advection, their method projects the mesh onto an implicit surface representation in every frame. This projection does not preserve surface details, and the size of the features remains limited by the resolution of the underlying simulation. While also adopting the projection scheme, Dagenais et al. [11] extracts high-resolution details based on the distance between an initial surface mesh and a coarse implicit surface representation. Their method allows the tracking of a detailed explicit mesh surface by using a coarser particle simulation. All these particle-based methods easily preserve volume and capture turbulent wavelets. However, they need to wrap an implicit surface around particle clouds. Therefore, the effects of particle-based methods are closely tied to the resolution and distribution of particles.

Implicit Surface Tracking. In contrast to particle-based methods, a more common technique builds a representation of the surface based on an Eulerian simulation embedded in a grid-like structure. A triangle mesh is generated at each frame from the implicit representation (i.e., level sets [16]), using a method such as *marching cubes* [22]. Implicit surfacing naturally handles topological changes and yields smooth surfaces. Losasso et al. [23] proposed a level set projection technique to model multi-phase interfaces. This method needs to build a signed distance field for each phase individually, which consumes a large amount of memory when many phases exist in a simulation. For multi-material surface tracking, the seminal works of [20] and [37] efficiently simulate multi-phase fluids with the RLS method. In the RLS, the traditional ± signs for identifying "inside-outside" are replaced with a series of region codes, and the liquid film among contacting regions can be naturally modeled regardless of the grid resolution. Closely related to level sets are the so-called volume-of-fluid methods [25], that explicitly track a surface by computing mass fluxes, but they are not often used in computer graphics applications due to flotsam and jetsam artifacts. The semi-Lagrangian contouring (SLC) method [3] computes signed distances exactly from a triangle mesh, but the accuracy is limited by the resolution of the implicit surface representation.

Explicit Surface Tracking. In implicit surfacing, if the local feature size is below the resolution of the underlying grid, geometric features will be smeared out, leading to gradual volume losses. This limitation has spurred interest in explicit surface tracking methods. Brochu and Bridson [5] developed a framework for tracking explicit surfaces based on continuous collision detection (CCD) of triangles, and mesh surgery is performed directly on the triangles to handle the splitting and merging of the surface. This framework has been used to model liquid surfaces with thin features [14]. Müller [26] advected a velocity field of an Eulerian simulation, and then voxelized and rebuilt a mesh. The mesh allows the identification of cells that contain thin sheets of liquid, but other small-scale surface details, such as ripples, are not preserved. Wojtan et al. [32] developed a method that effectively preserves features of the explicit mesh in regions where no topological changes occur. The qualities of the topological change detection and the remeshing are only dependent on the voxelization grid resolution. Thus, the simulation resolution can be coarser while preserving a detailed surface. This work was improved in [33] by maintaining sheets of liquids thinner than the grid size. For mesh-based tracking, Da et al. [9] extended the two-phase work of [5] to handle multi-phase interfaces. They proposed strategies for non-manifold topological merging, splitting and foam-type operations, by assigning material labels to triangles and introducing non-manifold mesh structures. This proposal leads to the first collision-safe mesh-based tracking framework for non-manifold structures, and the extra large amount of computation, due to mesh proximities among contacting liquids, are avoided. Da et al. [10] proposed a novel vortex sheet model for surface-only film simulation, where a scalar circulation quantity is attached to surface meshes to drive the whole motion. In such a surface-based simulation, another challenge is handling the merging and splitting of film surfaces.

The moving mesh (MM) methods [8], [24] use watertight volumetric elements (typically a tetrahedral mesh) to discretize a physical field, and the surface tracking is automatically accomplished as the mesh evolves. This method is particularly powerful in simulating mixing viscoelastic/turbulent liquids. Material labels are assigned to each volume element, and the interface is the subset of boundary faces that border differently labeled elements. Similar to explicit surface methods, these methods have to robustly handle the topological changes and maintain the mesh quality. In our opinion, a triangle mesh is less complicated than a volumetric mesh for mesh surgeries, and is also easier to control.

Unlike collision resolution based methods, our mesh-based surface tracking algorithm handles topologically changing events on the basis of the local remeshing technique [13], [32]. This technique replaces invalid mesh parts with isosurface creation, rather than manipulating the original mesh with high complexity. It is therefore more efficient than a collision resolution scheme. Furthermore, topological operations are unnecessary to perform in every time step; thus it provides significant freedom on how frequently the topological operations are performed. Our method also extends the RLS technique [37] and uses it in local surficial regions to handle tangled geometries and model thin liquid films.

3 SURFACE MODELING AND SAMPLING

3.1 Overview

The proposed surface tracker depends on three main data structures, namely, an explicit triangle mesh, an RLS field, and an underlying adaptive grid that is used to sample the entire surface region. Throughout this paper, the triangle mesh is denoted as S, which can be specified as a list of vertex positions and a list of triangles (triples of vertex indices). We assign material identifiers to each triangle and each vertex of S to distinguish different liquid phases. As in [37], the RLS is specified as a list of pairs, and each pair consists of a region identifier and a distance value. The sampling grid is denoted as G, which is a spectrally refined grid (SRG) similar to the grid structure employed in [12], [17]. In our framework, the mesh sampling, the topology surgeries, and the RLS evolution are all performed on the sampling grid G. According to the Nyquist sampling theorem, to achieve a visually smooth inter-conversion between S and RLS, the grid resolution (at the finest level) for evolving the RLS, is at least twice the grid resolution for mesh voxelization (in our system, a ratio of 2 is uniformly taken). At a high level, our surface tracking algorithm consists of the following subtasks in a solving step:

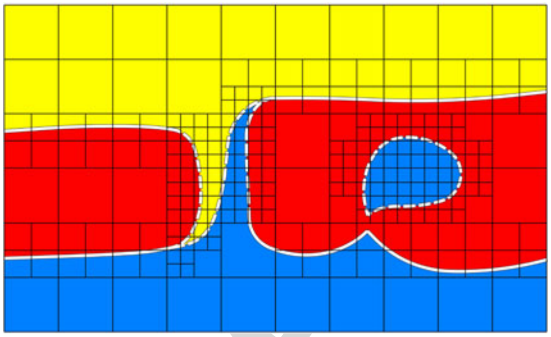


Fig. 1. 2D illustration of surface decomposition. Local surface regions to be modeled with RLS are marked with broken lines. The closed broken line on the right denotes an isolated local liquid block (or "pocket"), and the unclosed broken lines on the left denote interior tangled multi-phase sheets.

1) evolving the semi-explicit surface;
2) detecting mesh topology through voxelization (Section 5.1) and cell test of G (Section 5.3);
3) performing topology operations through local remeshing to handle mesh penetrations and the mesh/RLS coupling;
4) handling other topological events, such as mesh improvement and mesh/RLS inter-conversion.

3.2 Surface Modeling and Region Decomposition

As mentioned in Section 1, our framework uses explicit triangle meshes as the basic surface representation, and selectively models certain local regions with the RLS scheme. A sub-surface is modeled by RLS in two situations: namely, (1) some user-specified local liquids, such as a falling water ball or a rising bubble; (2) some subsets of S that are selected by our surface tracker.

To find the subsets of S that are to be modeled with RLS, we first identify isolated local liquid blocks trapped inside other liquids, such as oil drops or bubbles in water (see Fig. 1). For each subsurface mesh S_I enclosing a liquid block, we check its volume $V(S_I)$ by

$$V(S_I) = \frac{1}{6}\sum_{t\in S_I}(X_{t1}\times X_{t2})\cdot X_{t3}, \tag{1}$$

where X_{tj} is the position of vertex $j\in\{1,2,3\}$ in triangle t. If $|V(S_I)|$ is smaller than a user-specified volume threshold (e.g., $(10h)^3$, where h is the grid interval at the finest level of G), we further compute a weighted average of local feature size $F(S_I)$ by

$$F(S_I) = \frac{2}{\sum_i(1-w_i)}\sum_{X_i\in S_I}(1-w_i)D(X_i), \tag{2}$$

where $D(X_i)$ refers to the distance from X_i to the closest point on the medial axis of S_I, and can be estimated in a simple and fast way as proposed in [7]. The weight w_i is computed from a quartic spline function, i.e., $w_i = w(\frac{D(X_i)}{3h})$, as used in [27]. If the $F(S_I)$ of a liquid block is greater than a threshold (e.g., $4h$), which implies a relatively smooth geometry, S_I is replaced with RLS isosurfaces. That is to say, RLS is used for immersed liquid blocks that are relatively small and smooth. Note that $V(S_I) < 0$ indicates a surface "pocket" (with inward surface normal orientation) in the surrounding liquid, thus representation of these submeshes will be switched to RLS if $|V(S_I)|$ is small. Modeling immersed liquid blocks with RLS will reduce the complexity of S, as well as the number of mesh operations.

In addition to searching for isolated local liquids (i.e., "pocket"), we also search for local geometries of tangled interior sheets, as shown in Fig. 1. A similar local geometry can be rapidly detected by a "spreading" process. We first search regions with $F(S_I) < 4h$ for vertices that satisfy the following condition: if we shoot two opposite rays from such a vertex along its normal direction, they hit mesh triangles of different materials at least three times within a distance of $4h$. Once such vertices are found, we start to traverse their unvisited 1-ring neighbors and recruit those within $2h$-distance to the medial axis. This process stops when no new vertex is found or the number of vertices found for one region reaches a specified limit (e.g., 3000). Finally, the cells that contain such vertices are tagged for an RLS region, from which a local surface is reconstructed.

In addition to interior pocket elimination, other highlights exist for modeling certain local surfaces with RLS. Although the local remeshing technique is highly efficient, it only guarantees penetration-free at detail level of $2h$. Based on this observation, we find that tiny sharp geometries easily appear at the liquid-air interfaces, where slight mesh self-intersections are almost harmless to visual effects. Therefore we use explicit meshes for free surfaces as much as possible, to preserve sharp geometry. By contrast, at the interior liquid-liquid interfaces, if thin tangled sheets reach a penetration state that is difficult to resolve by a simple topological operation, the intersecting triangles with various material identifiers may cause visual flickers. A local high resolution RLS can be used to handle these complex situations and simultaneously guarantees the algorithm efficiency.

In some scenarios, other constraints, such as limiting the RLS region to visual fields sufficiently close to the viewpoint, can be imposed in designating which surface subsets are to be modeled with RLS. The local surface representation could also dynamically change during surface evolution, and we will discuss further details about the mesh/RLS inter-conversion in Section 7.

3.3 Surface Sampling

In our framework, the computational domain is sampled by a vector field $\mathbf{f}$ on the grid G, which contains three components, namely

$$\mathbf{f}(x) = (m(x), r(x), d(x)), \tag{3}$$

where x is the spatial position of any point in the space, m refers to the material identifier, r is the region code that indicates which RLS region x is inside, and d is the distance from x to the closest point on the nearby liquid surface. The values of either $m(x)$ or $r(x)$ are a series of consecutive non-negative integers. For convenience, we let $m(x) = 0$ denote solid, $m(x) = 1$ denote air, and $m(x) > 1$ denote liquids. We always use $r = 0$ as the region code of any spatial position outside RLS regions. Thus, region 0 can be considered the "ether" region for all RLS-modeled fluid parts. For immiscible materials, liquids inside the same local region must be of the same material, but not vice versa. The distance field d can be considered as the union of two distance functions

$$d(x) = min(d_S(x), d_R(x)), \quad d_S(x) \geq 0, d_R(x) \geq 0, \tag{4}$$

where d_S refers to the distance to the explicit mesh S and d_R refers to the distance to the RLS isosurfaces. In addition, we introduce two other functions $M_R(k)$ and $L_R(k)$ for RLS

$$\begin{aligned} &M_R(k) = m(x), r(x) = k, k > 0; \\ &L_R(k) = \{m(x_i) | r(x_i) = 0, r(x_i + \varepsilon e) = k, k > 0\}. \end{aligned} \tag{5}$$

$M_R(k)$ indicates the material of RLS region k, while $L_R(k)$ indicates the materials of all non-RLS fluids adjacent to region k, e refers to one of the six axis vectors $\{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$, and ε is a user-specified small value (e.g., h).

4 SURFACE EVOLUTION

Our fluid solver is defined on a hash grid, which can be regarded as an SRG with a coarser resolution. On this grid, the velocity field is advected by the back-and-forth error compensation and correction (BFECC) scheme, and the projection is solved by a second-order pressure discretization scheme. Our surface tracking starts with updating mesh vertex positions according to the velocity field. The mesh evolution and RLS evolution are performed independently. We advance S by moving triangle vertices without changing the original mesh connectivity. Vertex velocities are attained by interpolating the velocity field, and positions are computed by integrating the velocities using a third-order Runge-Kutta scheme [6]. Please note that a lower order integration method is also acceptable, since time accuracy is kind of useless when spatial accuracy is low. The RLS evolution needs to solve the advection equation discretized on the sampling grid G. As the surface evolves, G also needs an update. The SRG structure only contains a background lattice and a few fast hash tables that record the information of refined cells [12]. Unlike other spatially adaptive grids, such as the octree and tetrahedral mesh, this grid structure can be built much faster, because it only needs to mark and subdivide cells of the coarse lattice within a narrow band around the current surface.

The advection equation of RLS can be formulated as

$$\varphi_t(x) + U \cdot \nabla \varphi(x) = 0, \quad \varphi(x) = (r(x), d_R(x)), \tag{6}$$

where U refers to the fluid velocity field, and $\varphi_t(x)$ is the signed distance function. To save computation time, we only need to accurately compute φ values at grid nodes of small cells within a surficial narrow band. For nodes of coarse lattices that are far from interfaces, we can use an extrapolation technique to estimate the values of φ. To robustly solve the advection equation, a semi-Lagrangian scheme [29] can be applied. The 1st-order semi-Lagrangian operator $P(\varphi^{n+1}, \varphi^n, U, G_{new}, G_{old})$ can be formulated as

$$\varphi(x)^{n+1}_{G_{new}} = \varphi(x - U(x)\Delta t)^n_{G_{old}}, \tag{7}$$

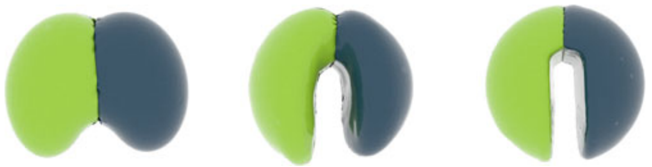


Fig. 2. Comparisons of different advection schemes with *Zalesak's sphere* test defined by Enright et al. [15], in which the grid resolution is 160^3. (left) First-order semi-Lagrangian with linear interpolation. (middle) First-order semi-Lagrangian with cubic polynomial interpolation. (right) Our BFECC scheme with linear interpolation.

where G_{new} and G_{old} refer to the SRG at the current and the previous time steps, respectively. Eq. (7) tells us that values of φ on G_{new} can be computed by interpolating back-traced positions on G_{old}. For the interpolation, definitions of the scalar multiplication and the addition operators for φ are identical to the those in [37]

$$\begin{aligned} a\cdot(r_1,d_1) &= (r_1, ad_1), \quad a\in\mathbb{R}^+;\\ (r_1,d_1)+(r_2,d_2) &= \begin{cases} (r_1,d_1+d_2), & r_1=r_2\\ (r_1,d_1-d_2), & r_1\neq r_2, d_1\geq d_2\\ (r_2,d_2-d_1), & r_1\neq r_2, d_1 < d_2.\end{cases}\end{aligned} \tag{8}$$

Though the semi-Lagrangian scheme is unconditionally stable, it produces excessive numerical diffusion and dissipation. To improve the computation accuracy, we extend the BFECC technique [21] for multi-phase liquids on SRG. The extension can be formulated as

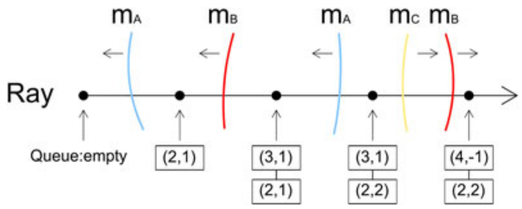
$$\begin{aligned} Forward \quad Advection &: P(\varphi^*, \varphi^n, U, G_{old}, G_{old})\\ Backward \quad Advection &: P(\bar{\varphi}, \varphi^*, -U, G_{old}, G_{old})\\ Error \quad Correction &: \varphi^* = \varphi^n - (\bar{\varphi} - \varphi^n)/2\\ Forward \quad Advection &: P(\varphi^{n+1}, \varphi^*, U, G_{new}, G_{old}).\end{aligned} \tag{9}$$

In our BFECC scheme, the first two semi-Lagrangian steps are applied on G_{old} to obtain a corrected φ on G_{old}. The error correction step uses the "$-$" operator. The "$-$" operator was given a definition in cyclic form in [37], which is impractical to implement. We define the "$-$" operator as follows:

$$(r_1,d_1)-(r_2,d_2) = \begin{cases} (r_1, |d_1-d_2|), & r_1=r_2\\ (r_1, |d_1+d_2|), & r_1\neq r_2.\end{cases} \tag{10}$$

In the RLS scheme, the values of $d_R(x)$ should theoretically stay non-negative, because the traditional "$\pm$" signs in the two-phase level set are replaced with regional codes. Negative distance values produce ambiguity in multi-phase cases. Our "$-$" operator only produces non-negative values; its physical significance is to reflect the distance change for a specified material at a spatial point, rather than a description of distance to the surface.

Both the "$-$" and the "$+$" operators defined in [37] are commutative, but none is associative. Therefore different computation orders lead to different results. This issue also occurs in the interpolation of $m(x)$ for material estimation, which we will discuss later. To guarantee a consistent calculation, we propose an order-independent interpolation operator, and details are provided in Section 5. Fig. 2 demonstrates that our BFECC scheme achieves a good advection accuracy. Furthermore, the BFECC scheme is only applied within the narrow band of G, and lattice nodes far away from the interface are updated with the first-order semi-Lagrangian scheme, which significantly reduces the total overhead.



5 TOPOLOGY DETECTION

The updated S may produce inconsistent geometry such as self-intersections, thus another task is to detect intersections and potential topological events. This process contains the following three substeps:

1) performing surface voxelization to generate the node states of G, and
2) determining materials for ambiguous node states, and
3) performing cell test to locate local grid cells where topological operations are needed.

5.1 Surface Voxelization

The surface voxelization step aims to determine the material properties of grid nodes. Müller [26] introduced a grid scanning technique to find the "inside-outside" state of a node in two-phase fluids, on the basis of a binary classification of space. For multi-phase liquids, we propose an advanced grid scanning technique that efficiently deals with general partition of space.

Scanning an SRG differs from scanning a regular grid. We perform the scanning along grid lines of coarse lattice, followed by local scanning of fine cells to visit grid nodes that are previously untouched. While scanning a ray along a grid line, we detect ray-triangle intersections and maintain a state queue Q which is initially set empty. Any member in Q is a pair consisting of a material identifier and a counter. Whenever a ray intersects a triangle t with material identifier m_t, we first construct a new pair (m_t, n), where the counter n is either 1 or -1, determined by the sign of the dot product of the ray vector with the triangle normal (A negative dot product means $n = 1$). If there exists a member in Q with material identifier equal to m_t, we add n, the counter of this new pair, to this member. Otherwise, we append the new pair (m_t, n) to Q. Whenever the counter in a member is zero, that member is deleted from Q. For each grid node the ray passes through, we store the current state of Q. The scanning process on a single grid line is illustrated in Fig. 3.

In real physics, as a ray goes through material A into material B, it should first hit the surface of A facing the outward normal directions and then hit the surface of B opposite to the normal directions. Therefore a valid state of list for a grid node G_X should be either an empty list or has only one member $(m_i, 1)$ in Q, which respectively indicates that G_X is outside any volume enclosed by S, or G_X is inside the liquid material m_i. Therefore two types of invalid state exist for G_X:

- *Self-intersecting state*. If only one member (m_i, n) is present in Q and $n \neq 1$, then G_X is in the region of multiple overlapping or inside-out surfaces of material m_i.
- *Intra-intersecting state*. If two or more members are present in Q, then G_X is in the region of intersecting surfaces with different materials.

Our mesh voxelization algorithm performs efficiently because it obtains information on all intersections and all node states, by scanning the grid only once (including all x, y, z directions); thus the time complexity and memory cost do not increase along with the increment in the number of phases. Furthermore, the SRG can be regarded as an accelerating structure for the voxelization process. Due to the narrow band refinement, the SRG has a memory cost of approximately one order of magnitude less than a global regular grid with the same effective resolution.

5.2 Material Estimation

As described in the preceding section, the mesh voxelization may leave grid nodes in intra-intersecting states, and we have to determine material identifiers for these nodes for later use. Since $m(x)$ is also evolved by the advection equation similar to Eq. (7), we compute $m(x)$ as follows:

$$m(x)^{n+1}_{G_{new}} = \begin{cases} M_R(r(\tilde{x})^{n+1}_{G_{new}}), & r(\tilde{x})^{n+1}_{G_{new}} > 0 \\ m(\tilde{x})^{n}_{G_{old}}, & r(\tilde{x})^{n+1}_{G_{new}} = 0, \end{cases} \quad (11)$$

where $\tilde{x}$ refers to the back-traced position: $\tilde{x} = x - U(x)\Delta t$. Eq. (11) manifests two points: (1) If $\tilde{x}$ is inside an RLS region, we directly assume the material of this region. (2) If $\tilde{x}$ is outside any RLS region, we derive the material through interpolation. Unlike Eq. (7), here the field to be advected is replaced with $\varphi(x) = (m(x), d_S(x))$, but the operators are identical. However, as mentioned in Section 4, an interpolation operator that guarantees consistent results (independent of computation orders) is required. We use letter c to denote generic identifier information and use d to denote the distance information. To get a linear combination of an arbitrary sequence $(\varphi_1, \varphi_2, \ldots \varphi_n)$, we follow the general RLS sum operation defined in [20]. First, elements with the same identifier are combined, forming a disjoint union set:

$$\{(c_{i1}, d_{i1}), (c_{i2}, d_{i2}), (c_{i3}, d_{i3}), \ldots\ldots(c_{ik}, d_{ik})\}, \quad (12)$$

where

$$d_{ik} = \sum_{m \in [1,n]} w(x_m) d_R(x_m), \quad \forall c(x_m) = c_{ik}, k \in [1, n], \quad (13)$$

and w is the interpolation weight. We then find the two largest distance values. For convenience, we assume:

$$d_{i1} \geq d_{i2} \geq d_{i3} \cdots \geq d_{ik}. \quad (14)$$

The final result of the linear combination is:

$$\varphi_{sum} = (c_{i1}, d_{i1} - d_{i2}). \quad (15)$$

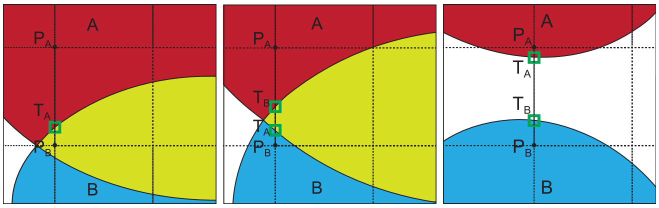


Fig. 4. Edge-wise cell tests that involve no RLS. (left) Ridging edge P_AP_B with a multi-phase node P_B. (middle) Ridging edge P_AP_B that crosses intersecting meshes. (right) Ditching edge P_AP_B where two end vertices are separated by a “ditch”.

The above steps produce consistent and convergent results in interpolation, and the distance values fall to zero at the interface, which also hold for two-phase cases. A first-order semi-Lagrangian scheme with a tri-linear interpolation is generally sufficient for material estimation. For RLS evolution (as shown in Eq. (9)), high-order interpolation, such as the WENO [30], is recommended to further improve the spatial accuracy. In this case, we temporarily allow negative distance values because some interpolation weights may be negative. However, if the final result introduces a negative value or a new extremum, which may cause numerical instability, we perform the tri-linear interpolation which guarantees a monotonic solution. This strategy was also employed in [17].

5.3 Cell Test

Cell test for G aims to search grid cells that involves invalid mesh components. This test is only necessary for refined cells within the narrow band of G, which surrounds the main surface. Through the test, certain cells are marked as *“complex cells”*, and mesh surgeries are only applied to regions that consist of *complex cells*. The cell test is divided into two types according to the interference of RLS in this step. They are the test for a cell with edge length $2h$, which does not intersect any RLS isosurface; and the test for a cell with edge length h, which is crossed by RLS isosurfaces. We can easily learn the test type and then select the proper test type by examining the material and region property of a cell. The two test types are elucidated as follows:

Cell Test without RLS. We analyze a cell in an edge-wise manner. A cell edge e_C, with its two endpoints denoted as P_A and P_B, is supposed to intersect the surface mesh S at point T. We then define two types of invalid edges:

- *Ridging edge*. If at least one endpoint is in a *self-intersecting state* or in an *intra-intersecting state*, e_C is marked as a *ridging edge*. For example in Fig. 4 (left), P_B is in an *intra-intersecting state*because it is in both materials A and B. In another case, as shown in Fig. 4 (middle), if $(|P_AT_A| + |P_BT_B|) > 2h$, then e_C is also marked as a *ridging edge*. In both cases, an intersecting region (in yellow) exists, which is much like a ridge formed by two moving patches of “lands”.
- *Ditching edge*. As shown in Fig. 4 (right), if two intersection points T_A and T_B exist and $(|P_AT_A| + |P_BT_B|) < 2h$, then e_C is marked as a *ditching edge*. In this case, P_A and P_B are located in two different materials which are separated by a “ditch”.

We mark any cell that is incident to a *ridging edge* as a *complex cell*. If a cell is only incident to *ditching edges*, where

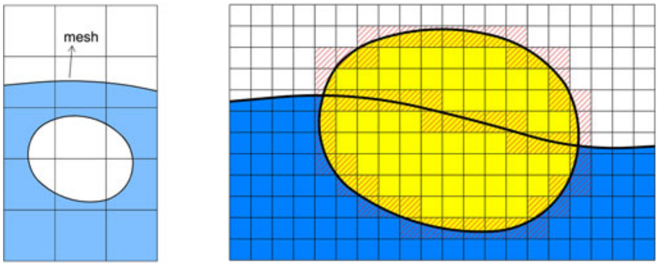


Fig. 5. Cell test for mesh/RLS coupling. (left) Isolated RLS regions far away from meshes do not trigger remeshing behaviours. (right) As an RLS region touches or intersects the mesh, cells that are crossed by the corresponding isosurfaces are marked.

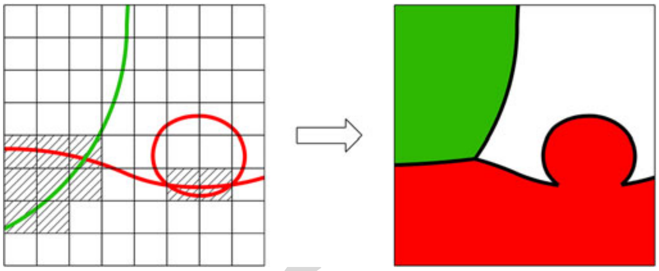


Fig. 6. 2D illustration of our general implementation of topology operations. According to materials involved in complex cells (colored squares), new interfaces are formed.

$|P_AT_A| + |P_BT_B|$ is close to $2h$, we never mark the cell because this case most likely indicates that the cell is crossed by the contact interface between two liquid materials. Therefore, if we mark this type of cell as *complex*, the surface tracker may remesh a large area of contacting interface, thereby leading to unnecessary mesh surgeries and smearing of the surficial geometry.

Cell Test with RLS. We perform two examinations in this step. The first is to find all RLS regions that touch or intersect the explicit meshes, and the second is to determine *complex cells* involved in these RLS regions. Finding RLS regions that are topologically coupling with S can be achieved by checking the material and the region codes of involved cell nodes for each RLS region. As shown in Fig. 5, if we find any cell where proximity between mesh S and RLS isosurfaces occurs, we mark all RLS regions overlapping with this cell, and then check other unmarked regions. We also update the $L_R(k)$ of Eq. (5) in this examination to find adjacent mesh materials for RLS regions. The second examination is relatively simple. We only mark the cells that are crossed by the isosurfaces of marked RLS regions. An extreme case also occurs in which cells completely inside an RLS region exist and are simultaneously crossed by S. These cells are marked as well, as shown in Fig. 5 (right). However, in practice, this case seldom happens because a liquid surface always evolves gradually and smoothly.

To this end, we have a set of marked cells. However, for the later topology surgery process, we must guarantee that each cell on the boundary of a marked cell region contains neither *ridging edges* nor edges that intersect triangles with the same material more than once. Thus similar to [32], we execute a flood-fill algorithm, starting with the initially marked cells and marching outward. When an aforementioned invalid edge is encountered, we go beyond it and mark the neighboring cells. As a result the marked region is bounded by topologically simple cube faces. Finally, we obtain a group of local grid regions, each bounded by marked cells.

6 TOPOLOGICAL OPERATIONS

6.1 General Implementation

Our scheme for topological operations is based on the local remeshing technique [13], which was further explored in [32]. Our local remeshing algorithm reconstructs triangle meshes in marked local grid cells. This process differs from that in [32] in two points. First, for complex cells involving multiple liquid materials, our algorithm reconstructs individual local meshes for each phase; Second, involved RLS regions are considered in our algorithm but not in [32]. This algorithm is summarized in Algorithm 1. To subdivide S along the boundary cell faces of a local grid region, the subdivision proposed in [32] accurately clips S along the boundary of the grid region (no triangle penetrates boundary cell faces). Similar to the topology detection process, the topological operations are classified into two types: namely, remeshing for cells that have no coupling with RLS regions (with edge length $2h$), and remeshing for cells that are involved in RLS regions (with edge length h). Type 1 and 2 remeshing aim to generate new local meshes to replace the original meshes inside the grid cells. The mesh stitching step aims to sew these newly generated meshes with the original meshes outside the grid cells, which can be robustly achieved by applying the subdivision stitching method in [32]. Fig. 6 presents a 2D illustration.

Algorithm 1. Topology Operation

```
Require: $S$; a group of local grid regions of $G$
for all local grid cells do
    $S \leftarrow$ subdivide $S$ along the boundary of the grid region
    for all complex cells $C_R$ in the grid region do
        for all mesh materials $m_i$ involved in $C_R$ do
            if the edge length of $C_R$ is $2h$ then
                $S \leftarrow$ type-1 remeshing
            else
                $S \leftarrow$ type-2 remeshing
            end if
        end for
    end for
    $S \leftarrow$ delete original explicit faces inside the cell
    $S \leftarrow$ mesh stitching
end for
$S \leftarrow$ extract faces from RLS isosurfaces and trim off those
    faces with open edges and append the rest to $S$
return $S$
```

Type-1 Remeshing. This process constructs local meshes for each material in a cell that has no coupling with RLS. Since *marching cubes* templates are used to generate new triangles, we only need to find one stencil point on each cell edge as a vertex of the newly formed triangle. The first step is to check the material identifiers of the eight cell nodes to match a *marching cube* template. The two-phase *marching cubes* algorithm can easily be extended into a multi-phase version. For a specific material m_i, cell nodes with material identifiers m_i are considered "inside" the material. The matched template indicates on which cell edges to find

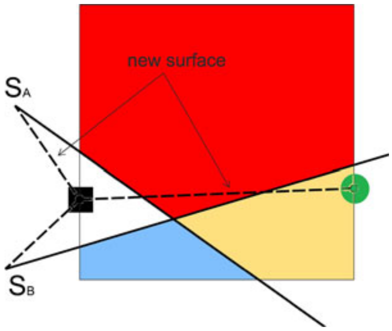


Fig. 7. Intersection adjustment in a Type-1 remeshing. A cell (the square) is crossed by two surface meshes S_A and S_B (the two solid bold lines), and the newly reconstructed interface is indicated as dashed lines. For the ditching edge (the left edge of the cell), we adjust the intersection points to a new position on the interface (the solid square). For the ridging edge (the right edge of the cell), we generate new stencil points (the solid circle).

stencil points. In [32], all stencil points are intersections between S and cell edges. However for multi-phase cases, things are quite different, e.g., a *ridging edge* may contain no intersection with the surface of material m_i, but we need to generate new stencil points to construct the contact interface (see Fig. 7). This step is summarized in Algorithm 2. To find the interface position on a cell edge, we interpolate the positions of two end points denoted as X_A and X_B, i.e,

$$X_{interface} = \frac{d_{S_B}(X_B)X_A + d_{S_A}(X_A)X_B}{d_{S_A}(X_A) + d_{S_B}(X_B)}. \tag{16}$$

Along the interface between two materials, duplicated faces and vertices are usually formed, with normals pointing to the inside of their respective materials.

Algorithm 2. Type-1 Remeshing for Material m_i

```
Require: $S$; $G$; a specified cell $C_R$ and material $m_i$
$C_R \leftarrow$ Identify node material w.r.t. $m_i$ to find a matching marching cube template
if no triangle is to be generated (the case of null template) then
    exit
end if
for all cell edges of $C_R$ do
    if no stencil point is required then
        go to check the next cell edge
    end if
    if the edge is a ridging edge or has no intersections with $S$ of $m_i$ then
        generate a new vertex at the interface position
    else if the edge is a ditching edge then
        adjust the intersection point to the interface position
    end if
end for
$S \leftarrow$ generate new triangles according to the matched template
return $S$
```

Type-2 Remeshing. The type-2 remeshing is similar to the type-1 remeshing. The only difference is that we need to check cell edges crossed by RLS isosurfaces and find stencil points on these edges, as shown in Algorithm 3. Our method uses RLS information to correct mesh information, and the isosurface position can be computed by replacing the distance function d_S in Eq. (16) with d_R. A 2D illustration for type-2 remeshing is shown in Fig. 8.

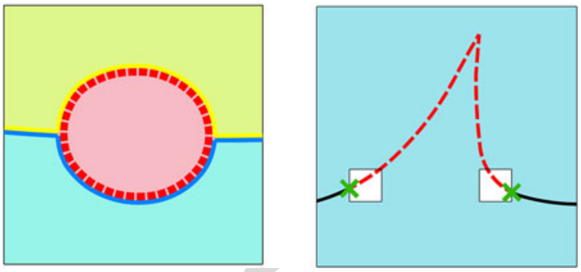


Fig. 8. Illustration of mesh/RLS coupling. (left) For RLS regions with closed isosurfaces, we directly construct local meshes (solid bold lines) along the triangulated contour of isosurfaces (broken lines). (right) For RLS isosurfaces with open boundaries, we need to stitch the constructed meshes with S at the corresponding boundary faces (green crosses).

Algorithm 3. Find a Stencil Point on a Cell Edge Crossed by RLS in the Type-2 Remeshing

```
Require: $G$; a specified cell edge $E$ and material $m_i$
if only one intersection point exists between E and the mesh of $m_i$ then
    adjust the intersection point to the isosurface position
else
    generate a new point at the isosurface position
end if
return the stencil point
```

Appending Contoured Mesh. After accomplishing topological operations in all local grid cells, we apply the *marching cubes* algorithm to RLS regions to extract explicit faces and append them to S. During the mesh extraction process, we find triangle vertices on cell edges whose two end points have not only distinct region codes but also distinct material identifiers. The extracted mesh (denoted as S^+) may contain unclosed meshes that have open boundary edges (see Section 3.2 and Fig. 1). Therefore, we stitch these meshes with the original meshes along the corresponding boundary cell faces.

The mesh appending process is crucial because it guarantees a closed manifold mesh, thus the mesh voxelization in the next solving step can be guaranteed safe. We perform topological surgeries per frame or even less frequently, rather than per time step. Extracting S^+ during the surgeries is not redundant as it is needed for rendering. Triangles of S^+ carry special marks and thus can be deleted immediately after the final S is output for rendering. These subsets of closed meshes indicate isolated liquid blocks enclosed by RLS isosurfaces, thus they are not needed for the voxelization process in the next time step because $M_R(k)$ functions always provide correct material information. Deleting these subsets will consequently reduce the computational cost without sacrificing robustness. For the subsets of unclosed meshes in S^+, the majority of their triangles will be deleted in future remeshing operations because the RLS cell test will mark most of their bounding cells as *complex cells* in the next frame.

6.2 Discussion and Optional Topology Strategies

From the preceding description, our method reconstructs meshes from grid-sampled data, rather than manipulating

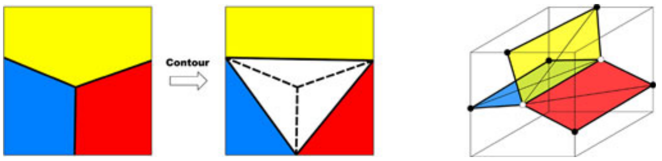


Fig. 9. Applying marching cubes to triple junctions (left) of isosurfaces will produce triangle meshes with volume gaps (middle). Construction templates with face stencil points (hollow circles) can fix these gaps (right).

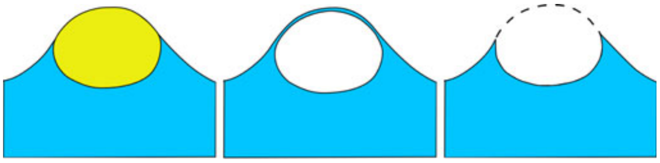


Fig. 10. Extended reconstructing modes. (left) Local meshes produced by conventional remeshing, which is incapable of modeling persisting bubbles. (middle) Meshes generated in wrapping mode. (right) Bubble rupture can be modeled by disabling the wrapping mode.

the original topology with high complexity. As contouring algorithms such as *marching cubes* always achieve a simple and valid geometry, the robustness of remeshing largely depends on the boundary stitching operations. Handling the coupling between RLS and S is also straightforward, as long as we can properly mark surficial cells of coupled RLS region and construct new local meshes. These newly constructed meshes are consistent with the triangle meshes extracted from RLS isosurfaces, thus they are intersection free.

Marking Two-Phase Liquid Cells. For the type-2 remeshing, a useful fact is that if a *complex cell* only involves two liquid phases (no air nor solid nodes), each phase will have coincident interfacial triangles relative to the other phase. Therefore, we can mark these cells and postpone the generation of new triangles. When extracting S^+, we apply *marching cubes* for any marked cell to generate triangles for one phase and directly make a triangle copy for the other phase.

Extended Triangulation Templates. If a cell involves more than two liquid phases, applying a common *marching cubes* will form volume gaps at the sub-grid scale, as shown in Fig. 9. This gap formation is an intrinsic problem when applying two-phase contouring algorithms to extract explicit meshes for multi-phase liquids, although the implicit isosurfaces have seamless junctions [20], [23]. This problem can be fixed by applying specially designed contouring templates, as shown in Fig. 9. Similar to [26], these templates need stencil points on cell faces to form the triple or quadruple junctions. The volume gap problem can also be fixed as a post-processing, in which the gap volumes are further triangulated. In general the volume gaps are visually negligible even if the grid is very coarse. Therefore we still use the regular *marching cubes* templates for most of our simulation examples.

The "Wrapping" Mode. When an isolated RLS region hits the surficial meshes of its surrounding liquid, our type-2 remeshing ruptures the mesh wrapping the RLS region and reconstructs new meshes for the surrounding liquid. This phenomenon is exactly what happens to purely implicit surfaces when the topology changes [20], [37]. However, our meshing mechanism supports another mode. This mode, called the "wrapping mode", can be used to model arbitrary thin films that wraps closed RLS isosurfaces. This mode can be achieved through a trivial modification to the type-2 remeshing step. Only the rule of material marking needs to modify. The rule is that for a cell node x, if $r(x) = k (k > 0)$ and m_i is the wrapping material of region k, then node x is marked as inside m_i. Here we can learn the wrapping material by checking the vector $L_R(k)$ (see Section 3.3) and selecting the material with the maximum contact area. By identifying node material in this manner, our algorithm can directly reconstruct the wrapping films. In practice, this mode can be used to model persisting bubbles in foam. The bubble rupture can be easily achieved if the wrapping mode is disabled, as illustrated in Fig. 10. The RLS in [20] also allows rupture control, by storing values at the edges of the RLS graph. Our mesh based approach is more powerful, considering that mesh vertices can keep thin film thickness, or any varying field on the film.

7 OTHER TOPOLOGICAL EVENTS

To improve simulation quality, other topological events, including mesh edge operations, mesh/RLS interconversion, RLS region rearrangement, and particle generation, are performed. They are not necessarily to be performed in every time step. Instead, we process these events once the topology operations listed in Algorithm 1 are accomplished.

Mesh Quality Improvement. These operations keep a relatively nice aspect ratio and an appropriate area for each triangle of S. The main operations include edge flip, edge split, edge collapse, and singular vertex/triangle pinch, as described in [5]. For multi-phase manifold meshes, the typical vertex/triangle pinch operations can be directly used to manage the mesh splitting process without extra effort, because triangles in a closed mesh subset do have the same material. However, the edge flip and edge collapse may cause slight mesh penetrations at liquid-liquid interfaces. This issue can be simply fixed by the traditional signed distance field correction technique [19], which perturbs the corresponding vertex positions to guarantee that vertices are inside correct phases.

Mesh/RLS Interconversion. For the transition from mesh to RLS, we use the technique described in Section 3.2 to detect additional mesh regions which should turn to an RLS representation. Once a new region is found, we pick a new region code and assign it to all interior nodes of this new RLS region. The corresponding $d_R(x)$ values can be initialized with the original $d_S(x)$ values. The transition from RLS to mesh is simultaneously performed. A single RLS region may break into smaller pieces due to liquid motion. Accordingly, we should first rearrange the RLS regions through a flood fill algorithm. We then examine each RLS region and convert it into mesh representation if one of the following two conditions holds: (1) The surface area of an RLS region becomes larger than a user-specified value; (2) The wrapping material (see Section 6.2) of an RLS region is gas or solid. We can compute the surface area by integrating on the subsets of the extracted mesh S^+ with triangles that are specially marked. For closed subsets of S^+, we simply reset the region codes of the interior nodes of the region to be 0 and unmark the corresponding triangles. For unclosed subsets of S^+, we first reset the corresponding region codes and triangle marks, and then we select connected triangles with

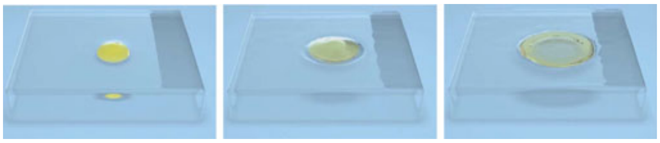


Fig. 11. As the contact area between the floating oil drop and the atmosphere becomes sufficiently large, our method converts the RLS-modeled liquid into mesh-modeled liquid.

a distance to the medial axis of S is less than $2h$ and use these selected triangles to reinitialize a "reduced" new RLS region. Furthermore, in some simulation cases, the transition for the second condition is useful in avoiding many type-2 remeshing operations for large surface areas, such as an oil drop spreading on the water surface, as shown in Fig. 11.

Particle Generation. To simulate the splashing droplets of a ruptured film, we can generate particles at the positions of deleted mesh vertices. In this step, our method ignores candidate particle positions that are significantly close to mesh triangles. Tiny mesh pieces or isolated RLS regions with volumes smaller than a threshold (e.g., $(2.5h)^3$) are converted into particles. We move particles by applying external forces, such as gravity, drag force and buoyancy force as described in [20].

8 RESULTS AND DISCUSSION

We ran several single-thread simulations on a 3.6 GHz Intel-i7 4790 CPU with 8 GB memory. On average, the surface-tracking algorithm takes approximately one third of the total computational time, while the remaining time is mostly spent on velocity projection and advection.

Fig. 12 shows a simulation of four oil balls rising up to the water surface. After a while, another two semispheres drop and splash. The four rising oil balls are modeled with RLS, whereas other liquids are modeled with meshes. The effective resolution of SRG is 120^3. We compare the results of our method with the typical RLS method (implemented with our BFECC scheme on SRG). As shown in Fig. 12, the pure RLS method demonstrates obvious volume diffusion during the splashing process. By contrast, our method changes the surface representation of oil balls from RLS to mesh when they spread on the water surface. Due to the detail preservation capability of Lagrangian meshes, the volume of each phase is better preserved and the feature loss is significantly reduced.

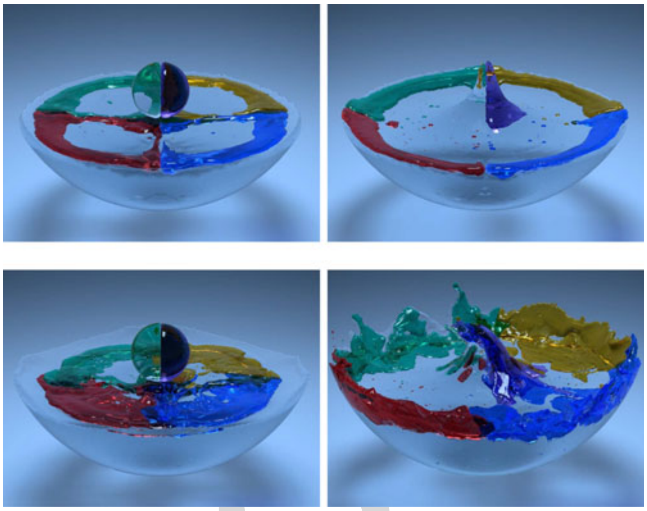


Fig. 12. Simulation of rising and falling liquids: (top) Results of the RLS method, and (bottom) results of our method.

Fig. 13 shows the simulation of the Rayleigh-Taylor instability phenomenon through a two-phase cocktail. All liquids are initially modeled with explicit meshes, with a total of 166 K triangle faces. During the liquid layer switching, our method continuously converts appropriate mesh pieces into RLS isosurfaces. At the moment of the most drastic liquid intertwining, the number of faces reaches 991 K, and the area of RLS isosurfaces is approximately 14 percent of the overall surface areas. We compare our method with the *moving mesh* (MM) method [24] by running a lower-resolution counterpart of this experiment (please refer to the accompanying video). In two methods, we adopt the same resolution for the velocity field, and the mesh resolution is adapted to be consistent with the corresponding velocity field. As a result, the SRG resolution for our method is $80 * 64 * 64$, and the number of effective tetrahedra for MM is $96 * 64 * 80$. In terms of computational efficiency (see the timing statistics in Table 1), our method is almost $4\times$ faster. In terms of visual appearance, our method introduces less damping to the motion of liquids. Although we are not well-grounded to explain that, the MM method has several disadvantages compared with ours. First, the MM method requires the resolution of the velocity field to be consistent with that of the mesh, and a low-resolution grid is not possible to combine with a high-resolution mesh. Second, the finite element based fluid solver is tightly coupled with the tetrahedral mesh, whereas our surface tracking mechanism is relatively independent and can be coupled with various types of fluid solvers. Third, the triangle faces constituting the interface are actually faces from tetrahedra, and two coincident faces have to be displaced by a tiny distance in opposite directions along the face normal, in order to make a renderer happy. Although a similar offsetting operation is required for RLS isosurfaces in our method, it can be more easily conducted along cell edges. Fig. 14 shows the simulation of a four-phase cocktail with an SRG resolution of 150^3.

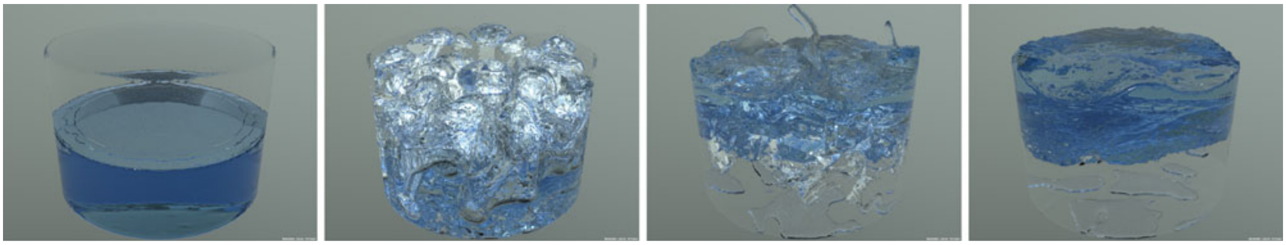


Fig. 13. Simulation of two-phase cocktail, the SRG resolution: 128^3.

TABLE 1
Timing for Simulating 720 Frames of the Two-Phase Cocktail with Our Method and the MM Method

	Resolution	Total	Proj.	Advec.	Track.	Others
Ours	128^3	5h20m	35%	25%	35%	5%
	$80 * 64 * 64$	2h	40%	20%	33%	7%
MM	$96 * 64 * 80$	7h40m	N/A			

The time percentage breakdown of our method is given in terms of projection, advection, tracking and other stuff.

Fig. 15 shows nine phases of liquids interacting inside a container of two connected spherical glasses. The SRG resolution is 256^3. We first disable gravity but apply a rotational force field to make these liquids twist and mix, then we resume gravity and let liquids drop. We intend to compare our method against collision detection based techniques. The El Topo method [5] is designed for manifold meshes that represent two phases, and the Los Topos method [9] is extended to handle non-manifold meshes that represent multiple phrases. We prefer manifold meshes as they are easily supported by renderers, whereas non-manifold meshes often confuse a renderer on the orientation of certain triangle faces. Therefore, we modify the El Topo method to make it support multi-phase cases, by assigning material tags to triangles and prohibiting merge operations among triangles of different materials. When using the modified El Topo to run the same experiment, the liquids soon gets locked during the rotation. When the gravity is resumed, multiple liquids are stuck at the bottleneck. The reason is that the CCD based technique only guarantees all collisions to be caught in the detection process, but the response process does not guarantee a safe post-collision state. If an intersection-free state is not achieved, the simulation will not advance to the next step, and the surface evolution may stagnate. This issue is also a threat to the Los Topos method, as acknowledged in [9]. By contrast, our method finds potential topological events through fast grid scanning and only reconstructs meshes in local regions where necessary. Our method permits slight self-intersections during the substeps in a frame interval, therefore the topology operations can be performed less frequently. These treatments drastically accelerate the simulation (about 5× to 8× faster than the El Topo).

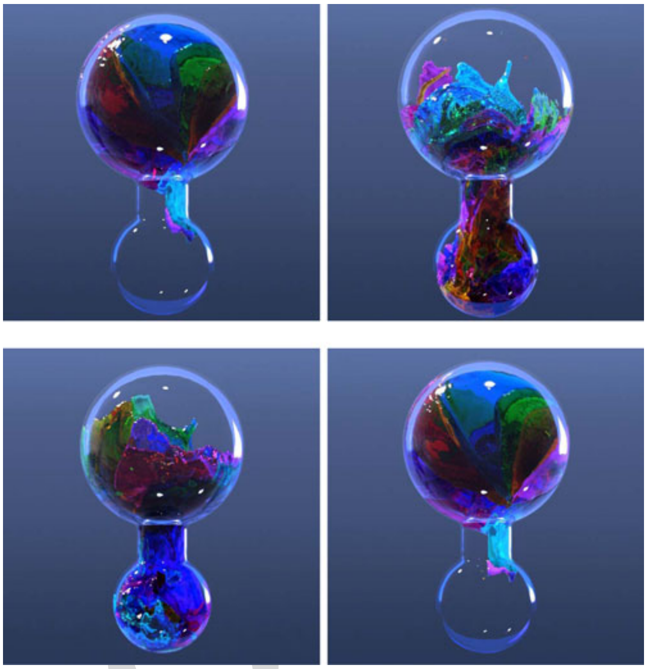


Fig. 15. Simulation of multiple interacting nine-phase liquids in a narrow container: (top left) Initial setup. (top right) Our method runs smoothly. (bottom left) Running El Topo encounters a halt in the middle of the simulation, and after a few seconds the simulation resumes (bottom right).

Fig. 16 shows a simulation of multiple splashing liquids on a 200^3 SRG. In this scene, the RLS-modeled liquid bunnies fall onto a background water surface, and produce drastically changing waves. The fine details of the free surface are well preserved.

9 CONCLUSION

We present a practical surface tracking approach for multi-phase immiscible liquids. Our method handles various types of multi-phase liquids. It utilizes both mesh-based Lagrangian surface tracking methods and the RLS method, and performs efficiently and robustly. However our method suffers issues similar to those in [32], due to mesh voxelization. First, the extracted isosurfaces may not match the surface mesh at ambiguous *marching cubes* faces. Subsequent mesh surgeries may consequently produce polygonal holes on the mesh. We address this issue by triangulating these holes to form a watertight surface mesh, as performed in [32]. Second, extremely thin sheets and strands of liquid may not be preserved because marching cubes cannot represent features smaller than a grid cell. The local convex hull method [33], which connects surface features during topological changes, preserves thin features efficiently. We

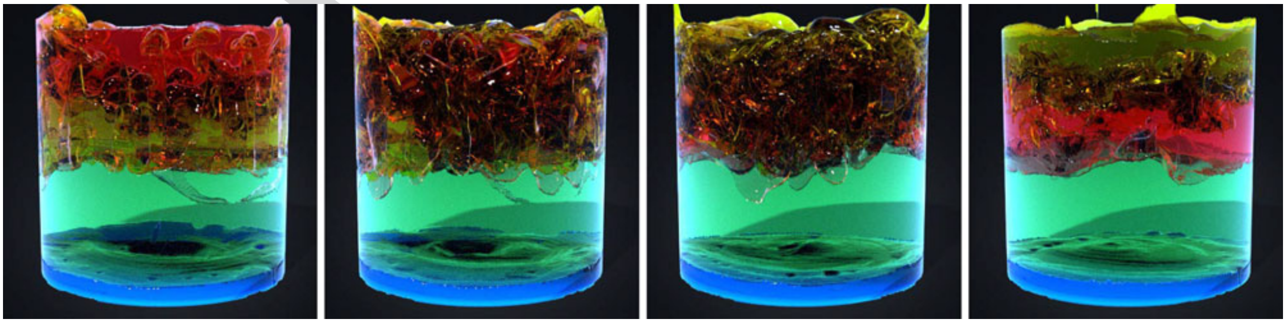


Fig. 14. Simulation of four-phase cocktail, the SRG resolution: 150^3.

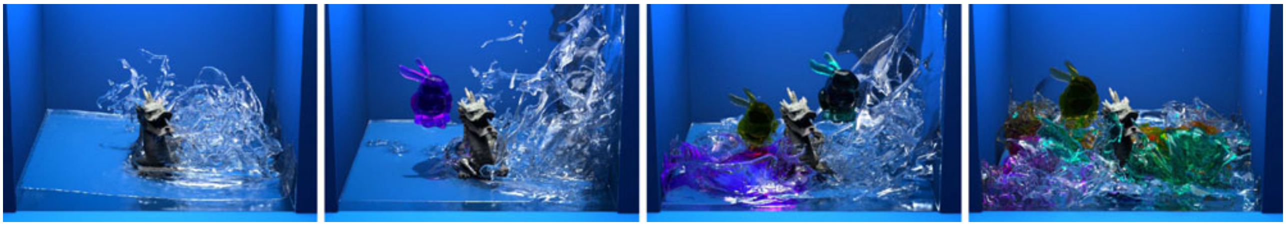


Fig. 16. Simulation of drastic splashing.

believe this method can also be adapted for multi-phase cases, i.e., we can identify *complex cells* that contain only one type of liquid material and then construct the convex hull of vertices. We aim to improve the topological operations of the "wrapping mode" for rupturing liquid films. In some cases slightly serrated geometries are formed at the film boundary, and we hope this formation can be alleviated by refining the underlying grid. The mesh error compensation technique [4] can be integrated into our framework to handle the minimal surface noise produced due to the mismatched resolution between the surface tracker and the physics solver. We further plan to add viscoelasticity and plasticity into our simulation in the future.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their valuable comments and suggestions. This work is partially funded by the National Natural Science Foundation of China (61772523 and 61620106003) and Chinese Guangdong's S&T project #2017B090912001.

REFERENCES

[1] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 48:1–48:8, 2007.

[2] R. Ando, N. Thürey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 103:1–103:10, 2013.

[3] 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, 2006.

[4] M. Bojsen-Hansen and C. Wojtan, "Liquid surface tracking with error compensation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 68:1–68:13, 2013.

[5] T. Brochu and R. Bridson, "Robust topological operations for dynamic explicit surfaces," *SIAM J. Sci. Comput.*, vol. 31, no. 4, pp. 2472–2493, 2009.

[6] J. W. Carr III, "Error bounds for the Runge-Kutta single-step integration process," *J. ACM*, vol. 5, no. 1, pp. 39–44, 1958.

[7] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in *Proc. SIGGRAPH/Eurographics Symp. Comput. Animation*, 2007, pp. 219–228.

[8] P. Clausen, M. Wicke, J. R. Shewchuk, and J. F. O'Brien, "Simulating liquids and solid-liquid interactions with Lagrangian meshes," *ACM Trans. Graph.*, vol. 32, no. 2, pp. 17:1–17:15, 2013.

[9] F. Da, C. Batty, and E. Grinspun, "Multimaterial mesh-based surface tracking," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 8:1–8:11, 2014.

[10] F. Da, C. Batty, C. Wojtan, and E. Grinspun, "Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 149:1–149:9, 2015.

[11] F. Dagenais, J. Gagnon, and E. Paquette, "Detail-preserving explicit mesh projection and topology matching for particle-based fluids," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 444–457, 2017.

[12] O. Desjardins and H. Pitsch, "A spectrally refined interface approach for simulating multiphase flows," *J. Comput. Physics*, vol. 228, no. 5, pp. 1658–1677, 2009.

[13] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, and Y. Li, "A simple package for front tracking," *J. Comput. Physics*, vol. 213, no. 2, pp. 613–628, 2006.

[14] E. Edwards and R. Bridson, "Detailed water with coarse grids: Combining surface meshes and adaptive discontinuous galerkin," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 136:1–136:9, 2014.

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

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

[17] N. Heo and H.-S. Ko, "Detail-preserving fully-Eulerian interface tracking framework," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 176:1–176:8, 2010.

[18] S. Ishida, M. Yamamoto, R. Ando, and T. Hachisuka, "A hyperbolic geometric flow for evolving films and foams," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 199:1–199:11, 2017.

[19] M. W. Jones, J. A. Bærentzen, and M. Sramek, "3D distance fields: A survey of techniques and applications," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 4, pp. 581–599, Jul./Aug. 2006.

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

[21] B. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Advections with significantly reduced dissipation and diffusion," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 1, pp. 135–144, Jan./Feb. 2007.

[22] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proc. 14th Annu. Conf. Comput. Graph. Interactive Tech.*, 1987, pp. 163–169.

[23] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 812–819, 2006.

[24] M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. B. Christensen, J. A. Bærentzen, and R. Bridson, "Multiphase flow of immiscible fluids on unstructured moving meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 1, pp. 4–16, Jan. 2014.

[25] P. Mullen, A. McKenzie, Y. Tong, and M. Desbrun, "A variational approach to Eulerian geometry processing," *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 66.

[26] M. Müller, "Fast and robust tracking of fluid surfaces," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2009, pp. 237–245.

[27] F. Sin, A. W. Bargteil, and J. K. Hodgins, "A point-based method for animating incompressible flow," in *Proc. SIGGRAPH/Eurographics Symp. Comput. Animation*, 2009, pp. 247–255.

[28] B. Solenthaler and R. Pajarola, "Density contrast SPH interfaces," in *Proc. SIGGRAPH/Eurographics Symp. Comput. Animation*, 2008, pp. 211–218.

[29] J. Stam, "Stable fluids," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Tech.*, 1999, pp. 121–128.

[30] Z. Wang, "High-order methods for the Euler and NavierStokes equations on unstructured grids," *Progress Aerosp. Sci.*, vol. 43, no. 1–3, pp. 1–41, 2007.

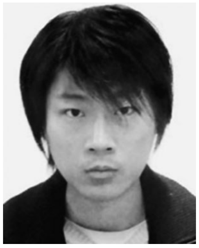
[31] C. Wojtan, M. Müller-Fischer, and T. Brochu, "Liquid simulation with mesh-based surface tracking," in *Proc. ACM SIGGRAPH Course Notes*, 2011, pp. 8:1–8:84.

[32] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming meshes that split and merge," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 76:1–76:10, 2009.

[33] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Physics-inspired topology changes for thin fluid features," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 50:1–50:8, 2010.

[34] T. Yang, J. Chang, M. C. Lin, R. R. Martin, J. J. Zhang, and S.-M. Hu, "A unified particle system framework for multi-phase, multi-material visual simulations," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 224:1–224:13, 2017.

[35] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 5:1–5:12, 2013.
[36] J. Yu, C. Wojtan, G. Turk, and C. Yap, "Explicit mesh surfaces for particle based fluids," *Comput. Graph. Forum*, vol. 31, no. 2, pp. 815–824, 2012.
[37] W. Zheng, J.-H. Yong, and J.-C. Paul, "Simulation of bubbles," in *Proc. SIGGRAPH/Eurographics Symp. Comput. Animation*, 2006, pp. 325–333.
[38] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 965–972, 2005.

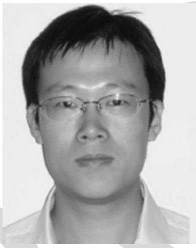


Meng Yang received the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2011. From 2012 to 2016, he was an assistant professor with the Institute of Automation, Chinese Academy of Sciences. He is now a software engineer with Amazon. His research interests include computer graphics, particular physically based simulation.

Juntao Ye received the BEng degree from Harbin Engineering University, in 1994, the MSc degree from the Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, in 2000, and the PhD degree in computer science from the University of Western Ontario, Canada, in 2005. He is currently an associate professor with the National Laboratory of Pattern Recognition of the Institute of Automation, Chinese Academy of Sciences. His research interests include graphics, particularly physically-based simulation of cloth and fluid, as well as image/video processing.

Frank Ding is a visual effect specialist with Light Chaser Animation Inc. His research interests include real-time rendering, physically based fluid/rigid body simulation, virtual reality, and machine learning.

Yubo Zhang received the PhD degree from the University of California, Davis, in 2014. From 2014, he has been a GPU architect with NVIDIA. His research interests include volume rendering, real-time global illumination, physically-based animation, moving mesh methods, and next generation GPU architecture.



Dong-Ming Yan received the bachelor's and master's degrees from Tsinghua University, in 2002 and 2005, respectively, and the PhD degree from the Hong Kong University, in 2010. He is currently an associate professor with the National Laboratory of Pattern Recognition of the Institute of Automation, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.