

Multimaterial Front Tracking

Fang Da, Christopher Batty, and Eitan Grinspun
Columbia University

We present the first triangle mesh-based technique for tracking the evolution of general three-dimensional *multimaterial interfaces* undergoing complex topology changes induced by deformations and collisions. Our core representation is a non-manifold triangle surface mesh with material labels assigned to each half-face to distinguish volumetric regions. We advect the vertices of the mesh in a Lagrangian manner, and employ a complete set of collision-safe mesh improvement and topological operations that track and update material labels. In particular, we develop a unified, collision-safe strategy for handling complex topological operations acting on evolving triple- and higher-valence junctions, and a flexible method to merge colliding multimaterial meshes. We demonstrate our approach with a number of challenging geometric flows, including passive advection, normal flow, and mean curvature flow.

Categories and Subject Descriptors: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Additional Key Words and Phrases: front tracking, multimaterial flows, triangle meshes, remeshing, topology change

1. INTRODUCTION

Representing and tracking the evolution of dynamic surfaces is a challenging problem with a host of applications spanning animation, geometric modeling, scientific computing, processing of image and volume data, and more. The most familiar setting is a domain consisting of *two* materials or phases, one interior and one exterior, separated by an interface. This is the case for the liquid-air surface in a simple water animation.

More challenging is the case of general *multimaterial* interfaces, that divide space into a set of two *or more* distinct materials; for example, a multiphase flow featuring three immiscible fluids, such as air, oil, and water. Numerous additional physical and mathematical applications have this same form: soap bubbles and dry foams, crystal grain growth, geometric flows, minimal surface problems, and various kinds of multi-cellular structures [Saye and Sethian 2012b].

Recently, Saye and Sethian presented a powerful and flexible *Voronoi Implicit Interface Method* (VIIM) to track multimaterial interfaces [Saye and Sethian 2012a; 2012b]. The approach represents the interfaces *implicitly* via an unsigned distance function sampled on a regular grid. Indeed, the dominant paradigms for multimaterial interface tracking have thus far built on classic implicit surface approaches: level sets [Osher and Fedkiw 2002], volume-of-fluid (VOF) [Hirt and Nichols 1981], and particles [Harlow and Welch 1965].

On the other hand, *explicit* triangle mesh-based surface tracking, or *front tracking*, has long been suggested as an alternative in the computational fluid dynamics community [Glimm et al. 1988; Unverdi and Tryggvason 1992; Glimm et al. 1998], at least in the simpler two-material case. Such methods can better preserve details and thin geometries, and a number of authors [Du et al. 2006; Brochu and Bridson 2009; Müller 2009; Wojtan et al. 2009; Campen and Kobbelt 2010; Zaharescu et al. 2011] have recently demonstrated implementations of this concept that overcome

its well-known Achilles heel: geometric robustness problems in resolving complex topology changes caused by collisions. However, to our knowledge no existing three-dimensional front tracking method has addressed these challenges when three or more materials are involved.

Contributions: The purpose of the present work is therefore to develop the first fully general three-dimensional *multimaterial front tracking* algorithm, and demonstrate its efficacy, flexibility, and robustness. Achieving this goal at first appears quite daunting: allowing non-manifold, multimaterial geometries dramatically expands the space of possible entangled mesh configurations that can arise, and topological transformations that must be supported. Our central contribution is to show how this complexity can be reduced to handling essentially two fundamental operations: *multimaterial merging* and *T1 processes*.

Merging occurs when closed material regions collide. While the two-material case has been considered by existing front-tracking methods, the multimaterial case has not. As an example, when a droplet of water and a droplet of oil collide in air, they do not merge into one; instead, a new oil-water interface maintains their separation. We design a multimaterial merging strategy for this scenario, based on locally remeshing and snapping together nearby mesh elements. To ensure that our geometry remains a valid, non-overlapping partition of space throughout this and all other mesh transformations, we maintain an *intersection-free invariant*: following Brochu and Bridson [2009], we check each individual mesh operation for potential intersections, canceling any that violate this invariant.

T1 Processes occur when the local connectivity of volumetric regions changes due to deformations [Weaire and Hutzler 2001]. A familiar example arises for bubbles in a foam structure: as bubbles push past each other they rapidly rearrange, and their local neighbor relationships adjust accordingly. Non-manifold mesh-based algorithms for simulating T1 (and the simpler T2) processes have been developed by authors studying minimal surfaces, foams, and grain growth [Brakke 1992; Lazar 2011]. However, these methods do not consider collisions, and therefore cannot be directly applied to the general front tracking problem. To respect our intersection-free invariant, we must carry out *collision-safe* T1 processes. We break down a (potentially quite complicated) T1 process into a sequence of fine-grained *atomic* mesh operations, each treated in an all-or-nothing fashion. Each operation affects only a small mesh neighborhood, making it more likely to succeed without introducing mesh interference. To construct the fine-grain operation sequence, we introduce and exploit a *region graph*, a duality mapping between the local neighborhood of a multimaterial mesh vertex and the graph of material incidence relations.

The resulting multimaterial front tracking algorithm is robust in the face of complex topological transitions and large geometric deformations, as demonstrated by a number of multimaterial geometric flows, including prescribed velocity fields, mean curvature flow, and multimaterial normal flow. In these experiments the algorithm successfully tracks complex configurations of multiple moving fronts, preserves sharp geometric features and mesh quality, and introduces new interfaces between distinct colliding materials.

2. RELATED WORK

There have been a host of surface tracking approaches presented in the computer graphics, computational physics, and scientific computing literatures, including level sets [Osher and Fedkiw 2002], volume-of-fluid (VOF) [Hirt and Nichols 1981], particles [Harlow and Welch 1965], and both surface- and volume-based mesh techniques [Wojtan et al. 2011; Misztal and Baerentzen 2012], along with various hybrids. Below we briefly review both general multimaterial approaches and existing two-material front tracking approaches. We then discuss non-manifold front tracking methods, as exemplified by *Surface Evolver* [Brakke 1992], and highlight some relevant literature on surface remeshing.

2.1 Multimaterial Surface Tracking

2.1.1 Level Set Methods. Level set methods are usually extended to multimaterial settings by replacing the binary sign of the distance field with an integer material label [Losasso et al. 2006; Zheng et al. 2006; Kim et al. 2007; Kim 2010; Saye and Sethian 2012a]. In some methods a single *unsigned* distance field is used, while in other cases each material requires a *signed* distance field to distinguish its interior from all others. Reconstructing quality multimaterial surface or volume meshes from this labeled implicit data is also an interesting problem [Wu and Sullivan 2003; Bronson et al. 2012].

2.1.2 Particle Methods. For particle-based surface representations, such as in SPH or FLIP methods, each particle can be augmented with a material label or *color* [Müller et al. 2005; Solenthaler and Pajarola 2008], or just directly assigned the appropriate material properties.

2.1.3 Volume-Of-Fluid Methods. The volume-of-fluid method uses a volume fraction per cell to implicitly represent the interface. Its multimaterial generalization takes a similar approach, where each cell stores one volume fraction *per material*, summing to one. As in standard VOF, a major challenge is reconstructing a reasonable, continuous interface geometry from volume fraction data (e.g., [Dyadechko and Shashkov 2008; Anderson et al. 2010]). The difficulty of this task is strongly exacerbated by the presence of greater than two materials.

2.1.4 Tetrahedral Methods. Volume-based labeling can be useful for meshes as well [Pons and Boissonnat 2007a; 2007b; Misztal et al. 2012; Misztal and Baerentzen 2012]. Using a tetrahedralization of space, each tetrahedron can be labeled, and the interface is the subset of triangles that border two tetrahedra with different labels. However, evolving a tetrahedral mesh while maintaining element quality is a complex and potentially expensive endeavor [Wicke et al. 2010; Clausen et al. 2013], and is wasteful if the particular application doesn't otherwise require a conforming volumetric mesh. Bronson et al. [2012] also briefly demonstrated an application of their Lattice-Cleaving tetrahedral mesher to multiphase flows.

2.2 Two-Material Front Tracking

We consider evolving a triangle mesh representing only the interface itself [Wojtan et al. 2011]. To our knowledge no existing Lagrangian triangle mesh-based surface tracking approach supports multiple materials.

2.2.1 Mesh Surgery / Boolean Approaches. Traditionally, the main difficulty with surface mesh approaches has been topology

changes. Colliding surfaces become entangled and must be stitched together through a mesh surgery process, similar to a mesh-based Boolean operation. In the past this approach has been prone to robustness issues [Glimm et al. 1998; Glimm et al. 2000], but advances in efficient and robust mesh Booleans have made it more practical [Bernstein and Fussell 2009; Campen and Kobbelt 2010; Zaharescu et al. 2011; Zhang et al. 2012]. Still, this approach cannot be directly mapped to the multimaterial setting where topological changes involve more than Boolean-like operations. For example, the collision and overlap of two fluid regions of different materials immersed in a third requires that a separating interface be established, and this interface is not a component of any existing surface. Glimm et al. [2000] allude in passing to this three-material case, but it does not appear that such a scheme has ever been developed.

2.2.2 Hybrid Implicit Approaches. To sidestep numerical robustness issues, several methods convert colliding regions into implicit surfaces, either globally or locally, apply a surface reconstruction method to build a new mesh or patch, and if necessary stitch a final mesh back together [Glimm et al. 2000; Du et al. 2006; Bargeil et al. 2006; Müller 2009; Wojtan et al. 2009; 2010]. Unfortunately, using basic implicit surfaces to manage topology changes means they rely crucially on the binary inside/outside classification; extension to the multimaterial setting appears non-trivial. Bo [2009] provides a potential starting point: he proposes a method that, under some simplifying assumptions, handles two regions bordering a static solid wall using a three-component marching cubes method.

The SpringLS method [Lucas et al. 2012] is another hybrid approach which combines a set of disconnected triangles with a distance field, and has been applied to multi-object segmentation tasks.

2.2.3 A Proximity-Based, Collision-Safe Approach. A third strategy, which we adopt, is that used by the *El Topo* code of Brochu and Bridson [2009]. They apply local collision-safe topological operations whenever meshes come in close proximity, and otherwise use fully robust cloth collision-processing techniques to maintain an intersection-free guarantee across both time evolution and remeshing [Bridson et al. 2002; Harmon et al. 2008; Brochu et al. 2012]. We will see that because this approach is entirely “grid-free” and relies only on simple, local mesh operations, it is extensible to the multimaterial setting. Stanculescu et al. [2011] proposed a related approach using a minimum separating distance bound to compute safe time step restrictions, rather than employing collision resolution.

2.3 Non-Manifold Front Tracking

We are inspired by *Surface Evolver* [Brakke 1992; 2012], which uses an evolving non-manifold triangle surface mesh to find equilibrium configurations for myriad variational problems. The method supports the various foam and film-like topological transformations arising in many multimaterial applications, during which the local connectivity of material regions changes. Our method expands on the applications of *Surface Evolver* in two ways: (a) we take into account collisions, which induce merging-type topological changes; (b) we advect the surface along arbitrary vector fields, not just gradient fields, thereby enabling general front tracking. Achieving these new goals requires several advances. For example, we must reframe surface evolution as a (quasi-)dynamical simulation respecting a no-interpenetration invariant. To respect this invariant becomes exceedingly difficult when topological tran-

sitions have many special cases or involve large simultaneous modifications to the mesh. Therefore we develop a unified topology resolution mechanism for the class of topology changes known as T1 processes (§5.1). The new approach avoids special cases and, crucially, iterates only on individual vertex operations. This eliminates the need for the “edge popping” used by *Surface Evolver*, and enables robust treatment of collisions.

We also point out recent work by Lazar et al. [2011; 2012] on normal grain growth. This method is likewise oblivious to collisions, and discretizes cell faces using only a single central vertex and a triangle fan; however, these simplifications enabled large-scale simulations of up to 100,000 regions in a grain growth problem. Lazar et al. also describe a set of five grain-level topological changes required for such problems; we instead describe a lower-level set of discrete local mesh operations whose collision-safety can be verified more readily.

2.4 Surface Remeshing

Remeshing of surfaces is a broad topic [Alliez et al. 2008]; we focus on incremental local remeshing, since it allows each remeshing operation to be robustly checked for collision-safety [Brochu and Bridson 2009]. Several authors have specifically considered remeshing of surfaces undergoing gradual but large deformations (sometimes called *mesh adaptation*), as opposed to one-time remeshing. High quality results have been achieved using remeshing criteria and operations that depend on anisotropy, curvature, and feature-detection [de Goes et al. 2008; Jiao et al. 2010; Clark et al. 2012; Narain et al. 2012; Wicke et al. 2010; Clausen et al. 2013]. We employ uniform resolution isotropic meshes with one particular choice of feature detection; combining our method with more complex remeshing criteria such as these is an interesting future extension. A few authors have also studied remeshing, subdivision, and fairing targeted at non-manifold surfaces [Hubeli and Gross 2000; Ying and Zorin 2001; Zilske et al. 2008; Pellerin et al. 2011].

3. MULTIMATERIAL TRIANGLE MESHES

We begin by describing a triangle mesh-based representation of multimaterial interfaces which forms the basic infrastructure upon which our method is built.

Existing front tracking approaches generally rely on a binary inside/outside classification of space. One common classification approach is parity counting of ray casts, however this does not readily extend to multiple materials.

The second common approach to binary classification is the use of a *consistent orientation* of triangular mesh faces: each triangle’s vertex ordering and the right-hand rule define its normal direction, and by convention the normal points to the interior (for example). If each triangle is oriented consistently with its neighbors and the mesh is both watertight and non-self-intersecting, this information defines a strict interior/exterior classification that can be exploited to perform safe topological stitching operations.

The non-manifold setting requires n -ary material classification; therefore, materials cannot be distinguished by the normal alone. Let each material be identified by an integer label. We assign to each triangle two material labels, associating one to the front (normal) and one to the back (anti-normal) sides; equivalently, each *half-face* stores a label. The *Surface Evolver* code uses this approach to specify different surface tension forces between pairs of materials [Brakke 2012], and Yuan et al. [2012] applied it in the context of constructing multimaterial implicit functions from non-manifold meshes. Figure 1 illustrates the analogous 2D case.

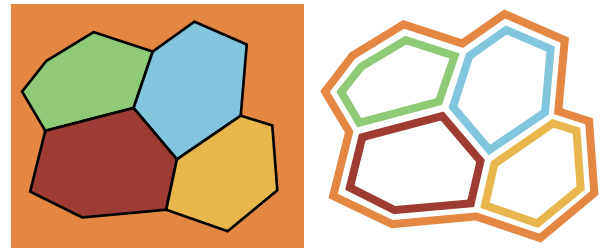


Fig. 1. Left: A 2D multimaterial object, with colors indicating material labels. Right: Its representation as a polygon mesh with material labels assigned to half-edges. The outer region is also assigned a label, shown in orange.

This approach no longer requires consistent triangle orientation, even for triangle pairs sharing a manifold edge; instead, it requires *consistent material labels*, in the sense that all half-faces incident to a closed volumetric region are labeled identically (Figure 1, right). We will exploit and continually update this extended notion of mesh orientation to ensure our regions remain consistently watertight.

Throughout, we will use the term *region* to refer to a closed volume of space, and the term *material* to refer to the type of each region, indicated by its incident triangle labels. Thus, two different regions may be composed of the same material, but a single region cannot consist of more than one material.

4. METHOD OVERVIEW

Algorithm 1 lays out the structure of our method, which relies on the core geometric representation described above.

Algorithm 1 Multimaterial Front Tracking

```

while simulating do
  Compute predicted vertex positions (§4.1)
  Eliminate interpenetrations via collision resolution (§4.2)
  Perform topological merging (§6)
  Perform mesh improvement (§7)
  Perform topological separation (§5.1 and §5.2)
end while

```

4.1 Computing Vertex Positions

The first step in our algorithm takes the current intersection-free mesh and computes for each vertex a proposed new position. Our method is agnostic to the source of these vertex motions, which will depend on the particular application. Typically, this data will be produced by integrating some physical system or geometric flow through time. The resulting mesh trajectories are allowed to temporarily leave the geometry in an intersecting state, as this will be resolved in the next step of the algorithm.

4.2 Collision Resolution and the Intersection-Free Invariant

Given the set of potentially colliding vertex trajectories, we now seek to minimally perturb these trajectories such that the new trajectories leave the mesh in a *non-intersecting state*. We resolve collisions using a standard cloth collision response scheme [Bridson et al. 2002; Harmon et al. 2008]. Within this scheme, we used the

exact geometric continuous collision detection method of Brochu et al. [2012], because it eliminated missed collisions in certain degenerate scenarios we encountered.

This *intersection-free invariant* is crucial to our algorithm: subsequent steps fundamentally rely on it to guarantee that remeshing and topology change can be performed safely and successfully, following the strategy of Brochu and Bridson [2009]. Beginning from the intersection-free state produced by collision-resolution, every proposed edit to the mesh is checked for intersections and canceled if any would be introduced. That is, individual mesh edits are treated as *atomic operations* that either complete in full or are canceled, and at all other times the mesh is intersection-free. The need to treat these operations atomically implies that we should prefer smaller, more localized operations where possible, since these can be more cheaply verified and are more likely to succeed in the presence of complex or tangled geometries.

The remaining three steps comprise our core contributions: handling of multimaterial topology change (§5 and §6) and mesh improvement (§7).

5. FOAM-LIKE TOPOLOGY CHANGES: T1 AND T2

Multimaterial scenarios undergo new types of topological transitions that do not arise in the two-material case. Two-dimensional versions of these transitions are shown in Figure 2 to provide intuition, which will be helpful as we approach the three-dimensional equivalents.

The first case represents two distinct material regions colliding while immersed in a third material; the result is that the middle region is divided in two, and a new interface maintains the separation between the originally disjoint outer regions (Figure 2, top). This contrasts with the usual two-material case in which two colliding regions will always have the same material and therefore merge into a single region. We defer discussion of merging to §6.

In the case of *four or more* material regions additional types of topology change can occur, referred to as *T1 and T2 processes* in the literature on soap films and bubbles [Weaire and Hutzler 2001]. In that application, the interior physical material is always air, but since a thin film keeps bubbles from combining into one, our method views them as distinct pressure regions or “materials”. In a *T1 process* two material regions that originally share a border become separated from one another, while two other material regions become connected to form a new interface; the number of regions remains unchanged (Figure 2, middle). In a *T2 process* a region collapses to a point and disappears, reducing the number of regions by one (Figure 2, bottom). This occurs commonly in convergent velocity fields. We collectively refer to T1 and T2 processes as “foam-like” operations, since this is the most familiar setting in which they arise, however they are necessary for any application involving greater than three materials.

5.1 T1 Processes

5.1.1 The Two-Dimensional Picture. We consider the behavior of a T1 process in two dimensions by first studying the mesh configurations involved. Most common are mesh vertices with edge valences of two or three, which we refer to as *regular* configurations. In most practical applications, vertices with valences greater than three exist only transiently, so we refer to these as *irregular* configurations. Two dimensional regular and irregular vertex configurations are illustrated in Figure 3. For example, Plateau’s laws describing bubbles and foams only allow stable junctions with va-

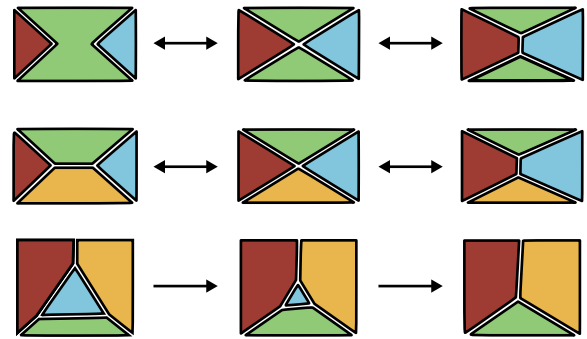


Fig. 2. **Multimaterial topology changes in 2D.** Top: A collision between two material regions separated by a third leads to a new shared interface. The reverse operation may also occur, leading to two regions disconnecting. Middle: A *T1 process* involving four regions; two regions that share a border separate and the remaining two regions become connected. This operation is reversible, depending on the velocity field. Bottom: A *T2 process* in which one region (cyan) collapses.



Fig. 3. **Regular and irregular vertex configurations in 2D:** Left: An interface separating two regions is regular, as is a triple-point vertex separating three regions. Right: Vertices with edge valence of four or higher are irregular.

lences of three or less [Weaire and Hutzler 2001] in two dimensions.

For a T1 process to occur, however, we cannot avoid irregular configurations altogether. We must enter such a configuration, and then *resolve* it back into appropriate nearby regular configurations with lower valence vertices. Without an explicit mechanism to do so, the single-valued nature of the input velocity field means that once created, such vertices are artificially preserved even as the surface continues to evolve, regardless of the underlying physics; this can be viewed as a kind of locking due to insufficient degrees of freedom.

A discrete two-dimensional T1 process can therefore be interpreted as a two step operation. Considering again Figure 2, middle, we see that as the two approaching regions (red and cyan) come close to one another, the horizontal interface in the middle shrinks to a point via an *edge collapse* operation, creating a new irregular configuration (vertex with valence four). This vertex is then resolved or separated back into two regular, valence three vertices, with the red and cyan regions sharing a new interface.

Geometrically, this process is perfectly reversible, so the choice of separation direction must be made by considering the underlying physics. *Surface Evolver* does this by considering surface tension forces on the vertex in question [Brakke 1992]. We will instead analyze the local velocity field provided by the user, as illustrated in Figure 4.

5.1.2 The Three-Dimensional Picture. Our philosophy in the three-dimensional setting follows the discussion above: we allow irregular configurations to arise naturally through collapsing of short edges during mesh improvement. We then seek to separate

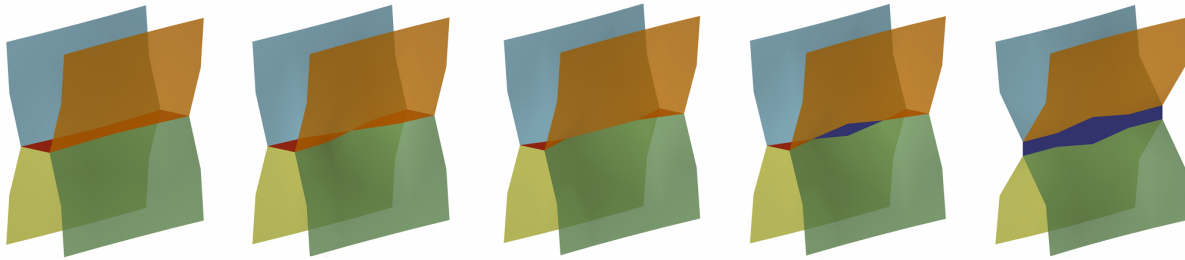


Fig. 5. **T1 process in 3D:** In this mean curvature flow example, a typical T1 process begins when a short edge between vertices on distinct triple-junction curves collapses, producing a single vertex incident on four material regions. Nearby edges on the original interface (red) continue to collapse yielding a sequence of edges on a quadruple-junction curve. Next, a vertex on the curve separates along a perpendicular axis leading to the creation of a new interface (dark blue). Lastly, nearby vertices along the curve also separate, completing the process.



Fig. 4. **Choice of separation direction:** The same irregular vertex configuration will separate in different directions depending on the velocity field driving the motion.

them back into regular configurations in a direction dictated by the velocity field.

However, whereas the mesh operations required to perform a T1 process in two-dimensions are fairly straightforward, the three-dimensional equivalents are much less obvious. Figure 5 shows a complete three-dimensional T1 process involving many individual mesh operations. Rather than attempt to resolve T1 processes through high-level, special-case code, we will show that **all irregular configurations can be treated in a simple and unified manner** by analyzing the local mesh topology and iterating on a single new type of mesh operation. The local incremental nature of this approach further enables complete collision safety, by allowing us to cancel any operations that would yield a self-intersecting mesh. Consistent with the *El Topo* method [Brochu and Bridson 2009], this can temporarily delay the completion of a topological event in order to preserve the intersection-free invariant; in practice, this did not lead to any observable artifacts even for problems involving hundreds of topological operations.

5.1.3 Region Graphs. We first consider the space of possible configurations about a vertex in three dimensions. The regular configurations again mirror the stable states described by Plateau’s laws (Figure 6). That is, a manifold surface or film by itself is regular; three surfaces may meet along a triple curve; and four triple curves may meet at a point [Weaire and Hutzler 2001]. However, unlike the two-dimensional case, both vertices *and edges* may have high valences, which leads to a tremendous variety of possible irregular configurations. Figure 7 illustrates a few of these.

To simplify the discussion, we adopt the following incidence definitions. A vertex and an edge (resp. triangle) are incident to each other if the vertex is one of the two (resp. three) vertices composing the edge (resp. triangle). A region and another simplex (i.e., vertex, edge, or triangle) are incident if any triangle bordering that region contains the simplex in question. Two regions are incident on one another *only if they share a triangle*; we will not consider two regions joined only by a vertex or edge to be incident.

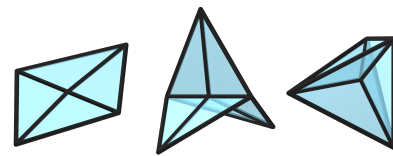


Fig. 6. **Regular configurations in 3D:** Left: A manifold surface separating two volumetric regions (in this case, the top and bottom half-spaces). Middle: A triple-curve separating three regions, at which three surfaces meet. Right: A quadruple-point at which four regions, four triple-curves, and six surfaces meet.

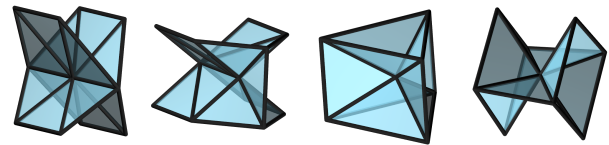


Fig. 7. **Irregular configurations in 3D:** Four of the infinitely many possible irregular non-manifold configurations that must be resolved by vertex separation.

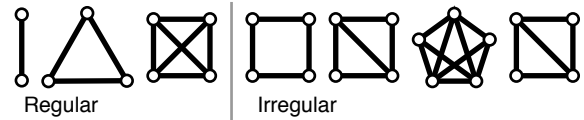


Fig. 8. **Region graphs:** Left: The region graphs for vertices at: (1) a two-region surface, (2) a 3-region curve, and (3) a 4-region junction. These regular cases are shown in Figure 6, and their graphs are *complete*. Right: The region graphs for the central vertices in the irregular configurations of Figure 7 are *incomplete*.

Because the space of possible irregular configurations is difficult to visualize and understand in three dimensions, we need a better tool with which to explore their topology and determine how to resolve them. We therefore define the concept of a *region graph* of a vertex. For each mesh vertex v , its region graph is an undirected graph in which each graph *node* corresponds to a region incident on v . Two nodes in the graph are joined by an *arc* if the two corresponding regions are incident. For clarity, we use the terms *node* and *arc* when referring to graph elements, and *vertex* and *edge* when referring to the three-dimensional mesh. Figure 8 shows the

region graphs for the regular and irregular configurations in Figures 6 and 7.

Next, we make the key observation that for a regular vertex configuration, the corresponding region graph will be a *completely connected* graph (Figure 8, left); all other graphs correspond to irregular vertex configurations (Figure 8, right). This is because two nodes that do not share an arc in the graph correspond to two regions incident on v but not each other. Therefore one way to characterize a regular vertex v is to say that every pair of regions incident on v are incident on each other. A missing arc in a graph indicates a pair of regions that are not incident, an irregularity that must be corrected. As suggested by Figures 9 and 10, this will be done by duplicating and separating the irregular vertex in a carefully chosen manner, and filling the gap that this creates with new geometry. We will now describe this *vertex separation* process in detail.

5.1.4 Vertex Separation. To ground our discussion in a concrete example, consider the central vertex v of the mesh in Figure 10, left. Its region graph has a missing arc corresponding to the top and bottom (non-incident) regions **A** and **B**. We duplicate and pull the mesh vertex v apart into new vertices v_a and v_b . Triangles originally incident to region **A** are updated to use vertex v_a rather than v , which simply requires relabeling one of its vertex indices. Similarly, triangles incident to region **B** are updated to use v_b . All of the remaining triangles incident on v , but not on region **A** or **B**, are updated to use v_b . Figure 10, middle, shows the mesh after it has been pulled apart.

Separating the vertex in this way leaves a disconnected gap in the mesh that we *may* need to fill, depending on the material types of the regions involved. The shape of this gap can be seen to arise from the set of existing edges incident on v “opening up” into triangular holes, as the original vertex v separates in two. Therefore, for each edge vw incident on region **A**, vertex separation creates a hole that may be filled by a new triangular face $v_a v_b w$, as in Figure 10, right.

However, we do not necessarily instantiate all such triangles, because some of these would erroneously separate regions consisting of the *same material*; i.e., the new triangle would have the same label on both sides making it unnecessary. This can be detected in advance by examining the two triangles incident on the edge vw and the region **A**. If the labels of these triangles on the *outside* of region **A** differ from one another, then the triangle is created to maintain the separation of these regions.

Returning to the topological view, Figure 10, bottom, shows the effect of vertex separation on the region graph. The original graph for vertex v is replaced by two distinct region graphs for the vertices v_a and v_b , both of which are complete. Therefore the new geometry is in a regular configuration, as desired.

Vertex separation supercedes the splitting or pinching of “singular” non-manifold vertices discussed by Brochu and Bridson [2009] in the two-material case.

5.1.5 Resolving Irregular Configurations. Vertex separation can be performed in this manner on any irregular vertex. However, when the original region graph of a vertex contains more than one missing arc (i.e., multiple pairs of regions are not incident on each other), a single vertex separation will not fully resolve the irregular configuration; v_a or v_b may still be irregular. However, instead of attempting to exhaustively deal with all possible irregular configurations in a single operation, we take an iterative approach by repeating the above process. This requires two additional ingredients: a strategy to pick which missing arc to process at each step (§5.1.6), and a guarantee that all irregular vertex configurations will ultimately be resolved.

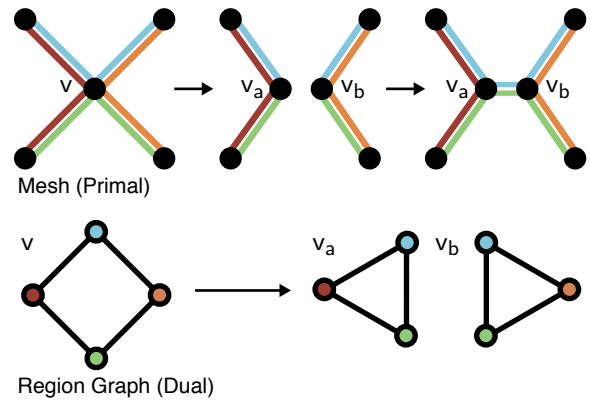


Fig. 9. **Vertex separation in 2D:** Top: An irregular vertex v in 2D (left) is duplicated and separated into two regular vertices, v_a and v_b (middle); the resulting gap is filled by a new edge with appropriate labels (right). Bottom: The incomplete region graph of v is correspondingly converted into two complete region graphs for v_a and v_b .

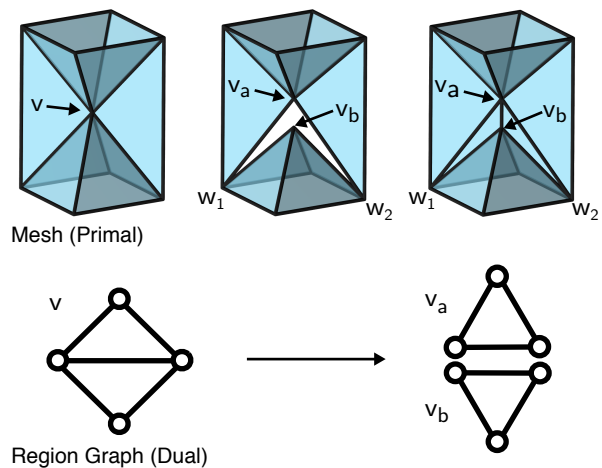


Fig. 10. **Vertex separation in 3D:** Top: An irregular vertex in 3D (left) is duplicated and separated (center), and the resulting gap is filled by new triangles with appropriate labels (right). Bottom: The incomplete region graph is replaced by two complete region graphs.

For now, consider arbitrarily picking one missing arc and applying the corresponding vertex separation: each of the two resulting vertices may still have missing arcs in their region graphs. However, their region graphs will have strictly fewer nodes, because the graph for mesh vertex v_a does not contain node **B**, and *vice versa*. Therefore, we simply put any resulting irregular vertex back into a queue to be processed, and we are assured that this iterative process terminates in finitely many steps with a set of complete region graphs, or equivalently, a set of regular vertex configurations. Figure 11 shows a vertex initially incident on six regions, which requires three vertex separations to resolve. The corresponding progression of region graphs is shown in Figure 12.

5.1.6 Choosing Separation Directions. In many cases there are multiple candidate region pairs $\{\mathbf{A}, \mathbf{B}\}$ at a vertex; the correct choice of which pair to separate is not arbitrary, but is instead de-

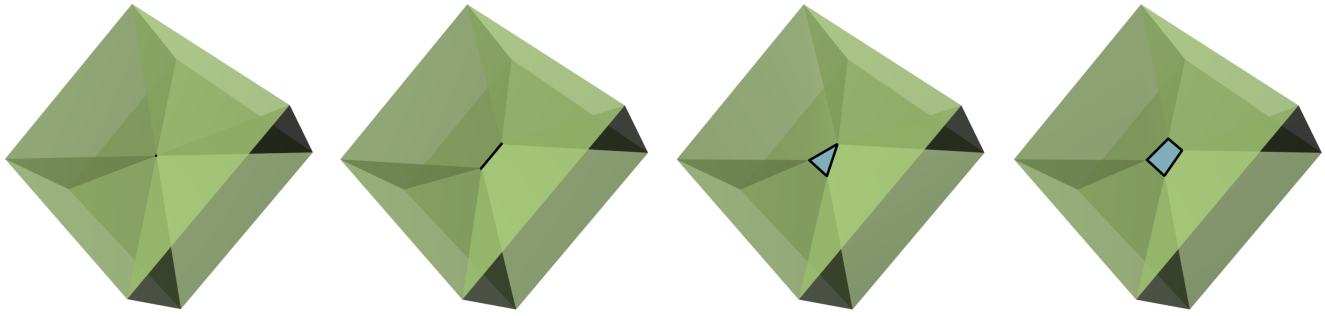


Fig. 11. **Resolution of a more complex vertex:** This geometry requires multiple vertex separations to return to a regular configuration. The center vertex is initially incident on six regions, and its region graph has multiple disconnected node pairs. A sequence of three vertex separations resolves this case and creates a new interface (green) between the front and back regions. The resulting four vertices are regular. The region graphs are shown in Figure 12.

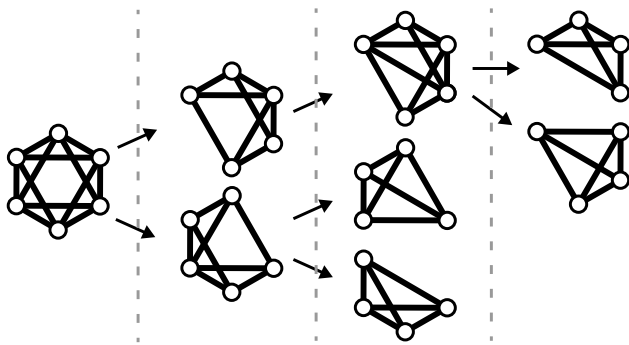


Fig. 12. **Complex vertex region graphs:** The sequence of region graphs corresponding to the topology changes in Figure 11. The original irregular vertex is split into two irregular vertices incident on five regions each. Each of these regions is subsequently split into two regular vertices, each incident on four regions. In the second step, while the top vertex is not directly involved in the mesh edits, its region graph is modified because the newly created interface connects the front and back regions.

terminated by the underlying velocity field. For each candidate pair $\{\mathbf{A}, \mathbf{B}\}$, we compute the vector between the centroids of the opening vertices of v that are incident to either region, and use it as the direction of separation. The dot product between the directional derivative of the velocity with the direction of separation, which we call *separation strength*, indicates how strongly the velocity field is converging along that direction, i.e., how strongly this pair desires separation. We compute this separation strength for each candidate pair, selecting the one with the highest value for processing. Algorithm 2 summarizes the complete process.

For settings without a prescribed 3D velocity field, the separation strength measure is problem-dependent and must be defined by the user. For example, mean curvature flow seeks to minimize surface area, so we examine the change in surface area for each proposed vertex separation. A decrease in the surface area indicates that separation should be performed. T1 processes do not occur for the normal flow scenarios considered, so we did not specify a separation strategy.

Vertex separation is performed in descending order of maximum separation strength over all irregular vertices. Because edits to the mesh can slightly change its resulting behavior, complex T1 processes may depend on the outcome of prior nearby T1 processes

at the discrete level. This global sort guarantees that irregular junctions are processed in a consistent order, ensuring that the algorithm is physics-dependent rather than machine-dependent. While this is more costly than greedily separating in the optimal direction for each vertex individually, the number of irregular vertices typically comprises a small and sparse subset of the complete mesh.

Algorithm 2 Vertex Separation

```

while T1 process has occurred in the last iteration do
  Candidate list  $C = \{ \}$ 
  for all vertex  $v$  in the mesh do
    Construct  $v$ 's region graph  $G = V, E$ 
    If the graph  $G$  is already complete, skip this vertex
    for all  $\{\mathbf{A}, \mathbf{B}\} \notin E$  where  $\mathbf{A} \in V, \mathbf{B} \in V$  do
      Compute separation direction  $\mathbf{d}_{AB}$ 
       $t_{AB} \leftarrow \text{separation\_strength}(\mathbf{d}_{AB})$ 
      if  $t_{AB} > 0$  then
        Add  $\{\mathbf{A}, \mathbf{B}, t_{AB}, \mathbf{d}_{AB}\}$  into the candidate list  $C$ 
      end if
    end for
  end for
  Sort  $C$  with descending  $t$  (separation strength)
  for all candidate  $\{\mathbf{A}, \mathbf{B}, t_{AB}, \mathbf{d}_{AB}\}$  in  $C$  do
     $t_{AB} \leftarrow \text{separation\_strength}(\mathbf{d}_{AB})$ 
    If  $t_{AB} < 0$ , skip this candidate
    Create vertices  $v_a$  and  $v_b$ 
     $v_a \leftarrow v + \epsilon \mathbf{d}_{AB}$ 
     $v_b \leftarrow v - \epsilon \mathbf{d}_{AB}$ 
    for all face  $f$  incident to  $v$  do
      if  $f$  is incident to region  $\mathbf{A}$  then
        Change  $f$ 's vertex  $v$  to  $v_a$ , keeping its labels
      else
        Change  $f$ 's vertex  $v$  to  $v_b$ , keeping its labels
      end if
    end for
    for all vertex  $w$  adjacent to  $v$  and incident to  $\mathbf{A}$  do
      Add face  $v_a v_b w$  with proper labels
    end for
  end for
end while

```

5.1.7 Consistency with Edge Collapses. Since edge collapses create irregular vertices, and vertex separations eliminate them, it is important that these operations be consistent. If the criterion for a collapse to occur disagrees with that for vertex separation, temporal incoherence artifacts can arise in which the same irregular vertex is repeatedly created and destroyed without the actual topology changing (i.e., a collapse initiates a T1 process but a vertex separation immediately reverts it). Therefore, when performing an edge collapse that would create an irregular vertex, we check whether the edge length is decreasing due to the velocity field. If not, we cancel the collapse operation, since this implies that the underlying physics is driving the geometry away from the potential T1 process, rather than towards it.

5.1.8 Collision Safety. It remains to ensure collision safety of vertex separation. To do so, we exploit the concept of *pseudo-motions* [Brochu and Bridson 2009], which approximates certain instantaneous mesh operations as a continuous deformation over a step of fictitious time. This allows continuous collision detection (CCD) to identify any collisions that arise between the relevant geometry and the rest of the mesh. Vertex separation can be viewed as a pseudo-motion transporting vertex v to the position of v_b , followed by a second pseudo-motion that separates v_a from v_b , bringing the triangles of region A along with it. The newly created gap-filling triangles $v_a v_b w$ are just the sweeping trajectories of the edges $v_b w$ due to the motion from v_b to v_a .

However in addition to colliding with the rest of the mesh, triangles and edges moved in the second pseudo-motion may also end up intersecting the geometry they were pulled apart from. This cannot be tested by CCD since these elements are all initially incident to vertex v , i.e., trivially colliding. We instead detect such intersections through instantaneous (static) collision detection on the final configuration. Specifically, the final positions of triangles incident on region A are tested against all mesh edges, and the final positions of edges incident on A are tested against all mesh triangles. This will guarantee the intersection-free invariant, but could potentially allow tunneling of small components. We rule this out by further testing for instantaneous collisions between all mesh vertices and the *volume* swept out by each moving triangle's pseudo-motion. Since only a single vertex of each triangle moves during the pseudo-motion, these volumes are tetrahedra, and a standard point-in-tetrahedron test suffices.

5.1.9 Ordering of Operations. In Algorithm 1, we order merging and mesh improvement passes prior to vertex separation. This choice ensures that potential T1 processes initiated by edge collapses are usually completed by vertex separation operations before the algorithm returns to the next time integration step. This in turn allows the underlying physics to continue evolving without locking four or more regions together.

5.2 T2 Processes

Having described a strategy to handle arbitrarily complex T1 processes, we now consider the simpler T2 process in which a region shrinks until it disappears, as may occur in convergent velocity fields (Figure 13). In the discrete case, a closed tetrahedron shaped region may undergo an edge collapse or edge flip during remeshing (§7) that yields a specific degenerate configuration: a pair of triangles sharing the same three vertices. Their labels nevertheless remain consistent; i.e., the conceptual zero-volume region “between” the two triangles is closed and consistently labeled.

To resolve this degeneracy, we detect if the outer regions of the two triangles have different material labels; if so, we delete *one*

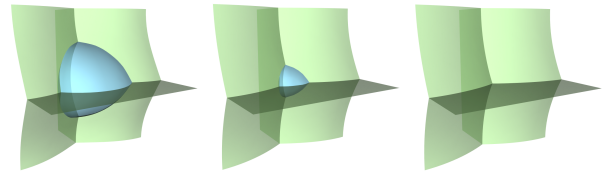


Fig. 13. **T2 process in 3D:** A T2 process occurs when a region shrinks to a point and disappears, such as this blue tetrahedron-shaped region collapsing under mean curvature flow.

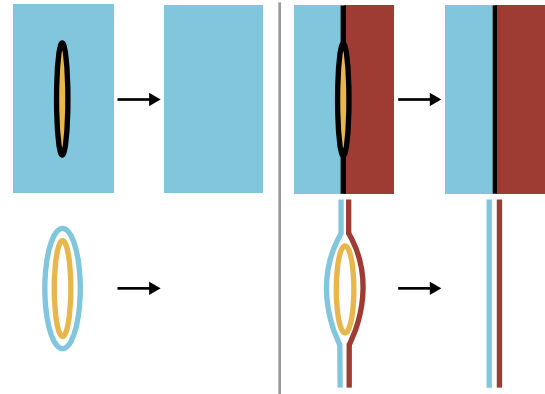


Fig. 14. **Removing degenerate regions in 2D.** Due to mesh improvement operations, shrinking regions eventually lead to degenerate zero-volume geometry (i.e., coincident but consistently labeled triangle-pairs, shown here in yellow). These must be deleted and/or relabeled. Left: If the exterior labels are the same, the enclosing elements are deleted. Right: If the exterior labels are different, one element is deleted, and the other is relabeled to eliminate the enclosed region. (Top: Volumetric regions, with interfaces in black. Bottom: Corresponding labels, where each interface becomes two labeled layers.)

of the two triangles and relabel the other to effectively remove the degenerate region, while leaving in place the necessary separating interface between materials. Otherwise, the outer regions have the same material and should be connected, so we simply delete both triangles. This latter case is consistent with the two-material situation described by Brochu and Bridson [2009]. Figure 14 illustrates this relabeling operation in two-dimensions.

This degeneracy removal can also lead to splitting apart of a single region, for example, if a thread-like geometry becomes sufficiently slender that its tetrahedra collapse and are deleted.

6. COLLISION-INDUCED TOPOLOGY CHANGES: MERGING

T1 and T2 processes consider topology changes involving regions that are already locally connected by the non-manifold mesh. The second key requirement of a multimaterial front tracking scheme is the ability to handle collision-induced topology changes between regions that are disjoint, either merging them into one or establishing an appropriate separating interface as in Figure 15. In our framework, this is done by applying local merge operations when geometry comes in close proximity, as measured by a proximity threshold chosen by the user. We first describe a direct multimaterial extension of the zippering approach of Brochu and Bridson [2009], and discuss its shortcomings. We then propose a new and

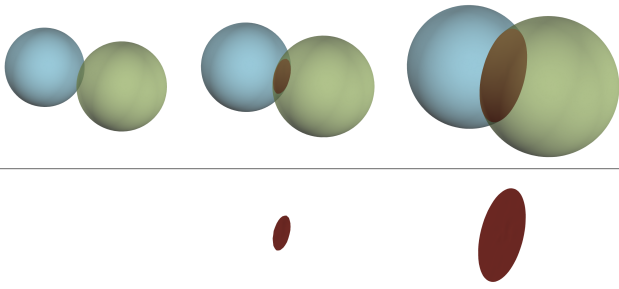


Fig. 15. **Merging in 3D:** Two spheres of different materials undergoing outward normal flow, from left to right. As they collide and merge, a new separating interface is created. Top: All interfaces visible with transparency. Bottom: The new interior interface (red) shown in isolation.



Fig. 16. **Zipper-based merging:** Left: Two nearby edges are identified, with incident triangles shown in blue. Deleting these triangles produces a quad-shaped hole in each surface. Middle: If the materials on their interiors are the same, the holes are zippered together to produce a connecting tube. Right: If the materials differ, one of the triangle-pairs is left in place but with its labels modified, to maintain separation of the materials.

more effective merging approach based on snapping vertices together.

6.1 Zipper-Based Merging

To handle merging of nearly colliding geometry in the two-material case, Brochu and Bridson [2009] proposed a zippering approach. First, proximate edge-edge pairs are identified, and for each edge the incident two triangles are deleted. This creates two quadrilateral holes which are then zippered together using eight new triangles to create a seamless tube connecting the previously disjoint regions. If the resulting geometry would induce interpenetrations, the operation is canceled. Stanculescu et al. [2011] proposed a similar approach in which proximate vertices are detected and their complete one-rings are zippered together, though without ensuring collision safety.

This approach can be extended to multiple materials by zippering in the same manner, but leaving in place and relabeling one of the two-triangle patches to create the separating interface (Figure 16). However, we experimentally observed that since zippering relies on colliding geometries being in relatively ideal alignment, many potential merges are canceled because they would induce collisions. Furthermore, in complex collision scenarios there may not be sufficient space to safely zipper at all, which can permanently prohibit the topology change. This may account for the lingering surface noise observed in some of the liquid animations of Brochu et al. [2010]. We conclude that zippering is overly restrictive, and ineffective in certain cases.

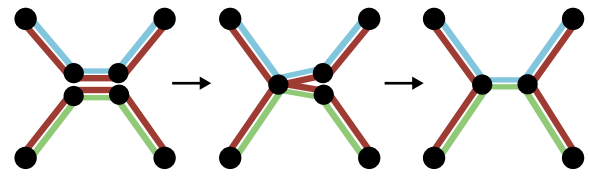


Fig. 17. **Ideal snapping in 2D:** Two edges approach one another in perfect alignment. First, one pair of vertices is snapped together, yielding a non-manifold state. Then a second pair is snapped together, the duplicate edge is removed, and its partner edge is relabeled to complete the merge.

6.2 Snap-Based Merging

We propose a new merging strategy based on *snapping* together of nearby vertices. The snapping operation is nearly identical to an edge collapse, in that it merges two existing vertices into one. The key difference is that the original vertices do not share an edge. We will see that this simple operation can lead to more effective merging.

Consider first an idealized scenario in which two perfectly aligned triangles approach one another head on. As they come within the merge proximity threshold, each pair of corresponding vertices is snapped together, so that the original two triangles become perfectly coincident. The degeneracy removal routine discussed in §5.2 handles this by removing one or both triangles, in the case where the two materials are different or the same, respectively. The result is a successful merge. The analogous 2D process is diagrammed in Figure 17.

Since meshes rarely collide in ideal alignment, we generalize this approach by adapting notions from compatible remeshing [Kanai et al. 2000; Alexa 2000]. Concretely, we attempt to locally modify the nearby meshes such that there *are* vertex pairs that can be easily snapped together. To do so, we identify nearly colliding edge-edge and triangle-vertex pairs, and subdivide them as necessary: each edge in an edge-edge pair is split at the closest point on the edge, and the triangle in a triangle-vertex pair is split into three triangles at the closest point on the face. Each vertex now has a counterpart on the opposing mesh, and snapping can proceed as in the idealized setting. Figure 18 illustrates the analogous process in two dimensions.

When snapping a particular vertex pair, we place the new snapped vertex at the average position of the original vertices. Splitting and snapping operations are checked for collision safety in the same manner as standard edge splits and collapses, respectively, and canceled if necessary. We also check these operations for the introduction of poor quality or degenerate geometry just as we do during mesh improvement (§7). To minimize poor quality geometry and unnecessary refinement, we also check if an edge or triangle would be subdivided close to one of its existing vertices, and instead directly snap to that vertex. Likewise, if a triangle would be subdivided close to one of its edges, we instead split the edge, and use the new edge midpoint for snapping. We do not snap vertices that already share an edge since this is the role of the edge collapse operation. We allow snapping between components of adjacent triangles only if the angle between the triangles is less than 90° . This allows snapping of folded geometry, while preventing snapping in the plane of the triangles.

Because this split-and-snap strategy involves smaller incremental edits to the non-manifold mesh rather than the simultaneous creation of a complete manifold tube, each individual operation is less likely to be halted by a collision. Furthermore, the splitting strategy

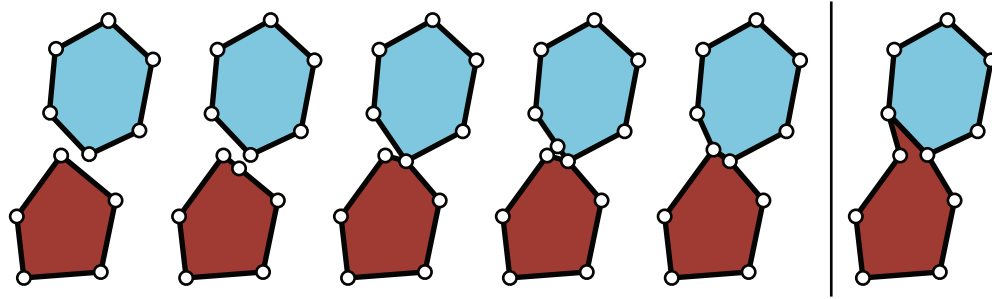


Fig. 18. **Merging in 2D:** Left: Snap-based merging begins as two regions come within the merge proximity. The edge in a proximate edge-vertex pair is split at the closest point, and the resulting vertex pair is snapped together. A second edge-vertex pair is processed in the same way to complete the merge. If the regions had the same material, the separating edge would then be deleted. Right: A zipper-based merge of the same initial mesh produces skewed geometry and a larger volume change.

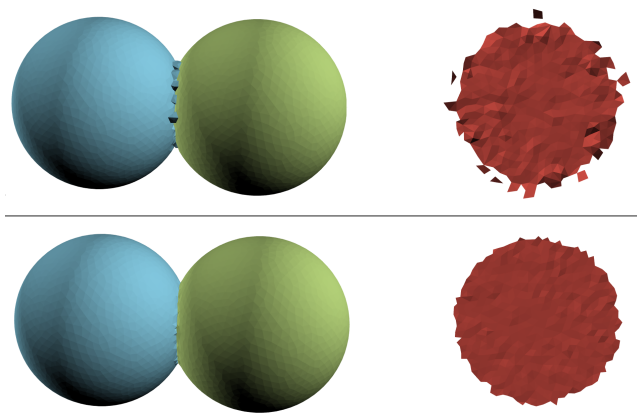


Fig. 19. **Zippering vs. Snapping:** Top: Zipper-based merging between two expanding spheres leads to noisy exterior geometry (left) in the colliding region and a discontinuous interior interface (right). Bottom: Our snap-based merging yields higher quality geometry (left) and a single gradually expanding interface (right).

actively works to modify the mesh so that merging becomes feasible; by contrast, zippering passively relies on the physics to put the mesh into a feasible state, which is problem-dependent and unlikely for complex deformations. As a result of these factors, we observe that our approach leads to more successful and smooth merging, as illustrated in Figure 19.

7. MULTIMATERIAL MESH IMPROVEMENT

As our mesh is advected by the flow and modified by topology changes, its constituent triangles can become misshapen, necessitating remeshing to improve triangle quality and maintain uniform triangle sizing. We apply an incremental feature-preserving remeshing strategy relying on four standard local operations: edge splitting, edge collapsing, edge flipping, and vertex smoothing. We largely follow the collision-safe remeshing strategy of Brochu and Bridson [2009] to preserve our intersection-free invariant. However, we have also made a number of modifications to specifically address non-manifoldness and multimaterial labeling. For completeness, we provide the details of our remeshing approach in Appendix A.

8. RESULTS

We will now consider a variety of geometric flows to illustrate the properties of the multimaterial front tracking scheme we have proposed. These properties include robustness and collision-safety under large, complex, and arbitrary multimaterial deformations, support for general merging, splitting, and foam-type topological changes, and good preservation of thin features and details. Simulation times ranged from a few minutes for low resolution examples up to 50 hours for the largest mean curvature flow example involving 2000 initial regions and up to 220K triangles simulated over 44K frames.

8.1 Prescribed Velocity Flows

We begin by performing multimaterial variants of two traditional surface tracking tests: the Enright and Zalesak disk tests. In the Zalesak disk test a disk with a notch cut out is rotated through 360° about an external point. The intent is to verify that sharp features are well-preserved under rigid body motions. In the Enright test a sphere is advected through a highly deforming velocity field which causes it to develop very thin features. Enright et al. showed that purely Eulerian level set methods (of which VIIM is one example) exhibit severe volume loss and over-smoothing on these tests, even when using higher order discretizations. On the other hand, hybrid particle level set methods perform better, at the cost of 32-64 Lagrangian particles per near-surface cell, but cannot handle geometric flows with merging characteristics such as inward normal flow [Enright et al. 2002]. By contrast, our Zalesak disk test (included in our supplementary video) exhibits no perceptible smoothing of sharp features. Similarly, our multimaterial Enright test, shown in Figure 20, illustrates that we can preserve thin features even under extreme stretching in multi-layered regions. Some accumulated wrinkling remains on the final sphere, since this is a purely passive flow. While further research into remeshing strategies may reduce these effects, surface tension or similar mechanisms should typically eliminate them in physical scenarios.

We constructed a more challenging robustness test featuring both substantial stretching and merging by alternating steps of outward normal flow and the curl noise velocity field of Brochu and Bridson [2009]. We applied this process to four spheres of different materials, which resulted in the complex deformations shown in Figure 21. For this test, we performed normal flow by computing area-weighted vertex normals and applying a constant offset.

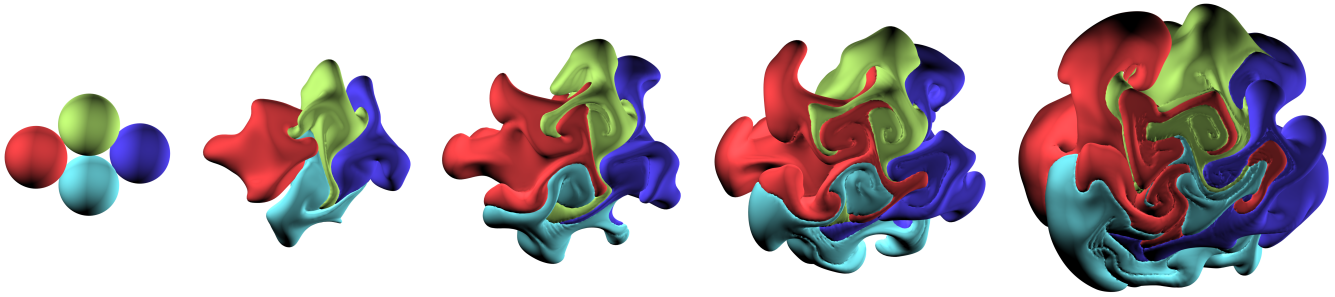


Fig. 21. **Normal Flow + Curl Noise:** Alternating steps of normal flow and curl noise applied to four spheres yields complex deformations and topology changes.

8.2 Constant Velocity Normal Flows

To examine merging in more detail, we consider multimaterial normal flow applied to two special cases of interfacial speed functions which were considered by Saye and Sethian [2012a]. (Fully general multimaterial normal flow is highly non-trivial even for level set methods [Reitich and Sonner 1996; Zhao et al. 1996], and to our knowledge has not yet been addressed for triangle meshes; this is an intriguing question for future work.)

First, we consider material surfaces moving (either inward or outward) at a constant speed; when they collide, a new interface is created and assigned a velocity of zero. Figure 15 shows an example in which two expanding spheres collide, leading to the formation of a circular internal interface. This non-manifold curve remains sharp, and the surfaces on either side remain smooth, due to our non-manifold remeshing strategy. Our supplementary video shows that when the direction of flow is reversed, the spheres seamlessly shrink until they pinch off and ultimately disappear.

Figure 22 shows a larger collection of spheres of various materials, indicated by their colors, which expand until they collide and merge forming complex intersection surfaces. Several of these spheres are assigned matching material labels, and therefore these pairs merge without a separating interface. Reversing the direction of flow causes the interfaces to smoothly shrink and disappear. Our supplementary video contains an additional example in which two Stanford bunnies undergo normal flow and collisions, while maintaining considerable detail in their expanding outer surfaces. These details would be rapidly smoothed away by Eulerian schemes.

The second special case of normal flow we consider follows an example by Saye and Sethian [2012a]. A set of three interfaces move at constant speed in the normal direction, with the signs of the velocities chosen to satisfy a *cyclical ordering*; that is, A flows normally into B, B flows normally into C, and C flows normally into A. The initial configuration is two overlapping spheres sharing a flat circular interior interface where they meet. This geometry produces a fascinating rotational or curling motion around the original non-manifold intersection curve, which remains stationary. The complex evolution of this geometry is shown in Figure 23, with the front half of the geometry cut away.

For these tests, we performed normal flow using the *face offsetting method* (FOM) developed by Jiao [2007]. This approach alleviates certain artifacts in naïve discretizations based on average vertex normals, at the cost of a more stringent time step restriction. Brochu and Bridson [2009] previously applied this method to the two-material case.

8.3 Comparing Snapping and Zippering

To illustrate the advantages of snap-based merging over zippering we consider the merging of two spheres under normal flow, shown in Figure 19. With zippering, the growing intersection curve and internal interface develop in a discontinuous and noisy fashion. By contrast, snapping yields smoother growth of the new interface, while the outer geometry exhibits greater overall symmetry and smoothness.

For these tests, we applied normal flow based on area-weighted vertex normals. This choice induces merging velocities reflective of typical physical simulations which undergo large numbers of topological operations during collisions. (Normal flow based on face-offsetting generates vertex velocities that, combined with a stiff time step restriction, avoid merge operations after the initial moments of contact; this yields better normal flow results, as in §8.2, but is less useful as a general test of merging behavior.)

8.4 Mean Curvature Flows

Figures 5 and 13 demonstrate mean curvature flow applied to two simple interface configurations inside a cubic domain, illustrating that we correctly handle T1 and T2 processes, respectively. Figure 24 illustrates mean curvature flow on a large arrangement of 2000 distinct material regions inside a cubic domain. This example demonstrates that our method can robustly handle large numbers of T1 and T2 topology changes, which is crucial for applications such as foam coarsening and grain growth [Weaire and Hutzler 2001; Lazar 2011].

For these tests, we applied the mean curvature flow discretization of Desbrun and collaborators [Desbrun et al. 1999; Meyer et al. 2002]. Since this approach is a gradient flow applied to the surface area of the interfaces, it straightforwardly extends to the non-manifold case by considering all triangles incident on each vertex.

9. DISCUSSION AND CONCLUSIONS

We have presented a novel and robust front tracking approach to the evolution of multimaterial interfaces, and applied it to a range of geometric flows. Our method includes a new more flexible and effective snapping approach for merging surfaces, and a unified vertex-based method to characterize and resolve complex topologies in multimaterial meshes.

A number of challenges and potential extensions remain. Certain applications would benefit from pervasive support for spatially adaptive and possibly anisotropic elements [Jiao et al. 2010; Narain et al. 2012], as well as other advances in basic two-material front

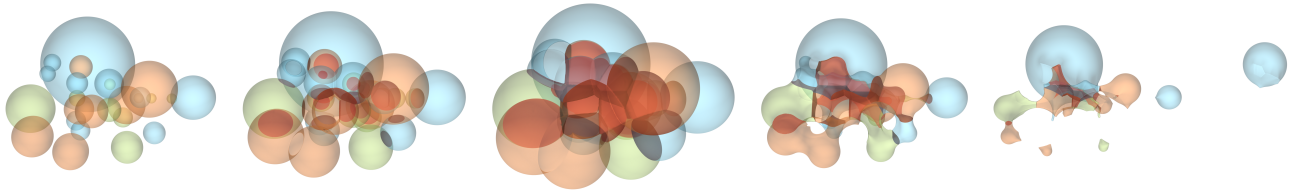


Fig. 22. **Multimaterial normal flow:** Twenty-five spheres of different materials and sizes initially undergo normal flow in the outward direction. As they collide regions of the same material merge, while regions of differing materials form separating interfaces (red) that steadily expand. The direction of normal flow is then reversed, causing the spheres to shrink and ultimately disappear.

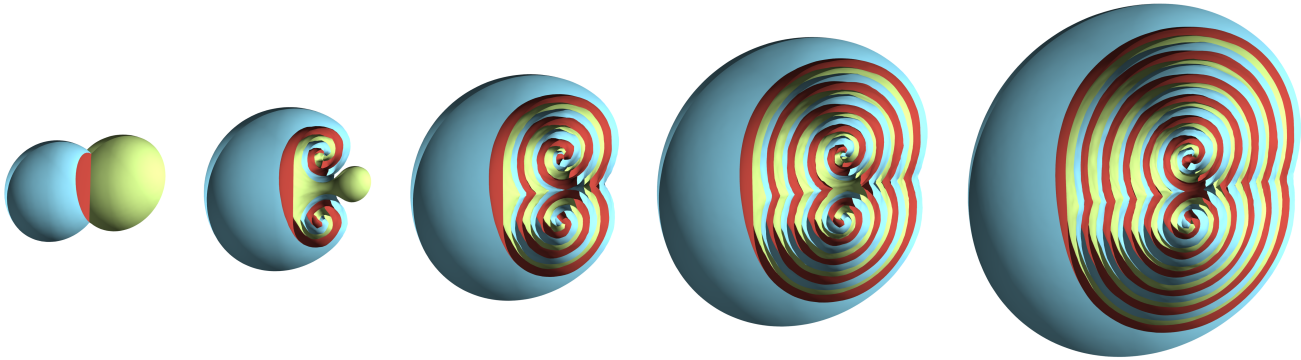


Fig. 23. **Cyclical normal flow:** Two overlapping spheres sharing a circular internal interface undergo normal flow with a cyclical ordering. The front half is cut away for visualization. The result is a curling effect around the original triple curve, which remains stationary.

tracking (e.g., [Clark et al. 2012]). In place of our independent split, flip, collapse, and smoothing passes, a unified local hill-climbing strategy could also be effective in preserving quality with minimum modification to the mesh [Wicke et al. 2010].

Collision detection is a key bottleneck. Our broad-phase uses a simple regular grid, and alternate approaches may provide speed-ups, especially for scenarios in which the geometry is unevenly distributed or the mesh resolution varies.

Visible popping can occur when collapse operations modify vertices lying on separate nearby features or non-manifold geometry. For example, a collapse occurring on a near-planar region is typically imperceptible, whereas a collapse that initiates a T1 process involves a discontinuous motion on the order of the minimum mesh edge length; this will be apparent at low resolutions. We experimented with using a smaller minimum edge length threshold in these scenarios, however this sacrificed mesh quality and lead to the need for smaller time steps. Moreover, level set-type methods also exhibit small instantaneous “popping” effects at the length scale dictated by the grid resolution, such as when two droplets collide. Spatial adaptivity may help to alleviate this issue.

Our algorithm currently relies on closed, consistently labeled volumes. Modeling of soap films and bubble blowing will require extending our method to open surfaces, which raises a number of interesting questions.

The proposed method has the potential to make feasible a unified triangle-mesh based approach to a very large class of interface evolution problems, including a superset of those for which the widely adopted *Surface Evolver* and *FronTier* software packages were designed [Brakke 1992; Du et al. 2006]. Examples include multimaterial elastoplastic deformations, complex multiphase and foam flows, PDEs on deforming non-manifold interfaces, segmentation and shape reconstruction problems, multimaterial sculpting

and modeling, and grain boundary evolution. We hope to explore many of these applications in future work. To encourage adoption and extension of our method, we will make the code publicly available.

REFERENCES

- ALEXA, M. 2000. Merging polyhedral shapes with scattered features. *The Visual Computer* 16, 1, 26–37.
- ALLIEZ, P., UCELLI, G., GOTSCHMAN, C., AND ATTENE, M. 2008. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*, L. Floriani and M. Spagnuolo, Eds. Springer, Berlin, 53–82.
- ANDERSON, J. C., GARTH, C., DUCHAINEAU, M. A., AND JOY, K. I. 2010. Smooth, volume-accurate material interface reconstruction. *IEEE TVCG* 16, 5, 802–814.
- BARGTEIL, A. W., O’BRIEN, J. F., GOKTEKIN, T. G., AND STRAIN, J. A. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph.* 25, 1 (Jan.), 19–38.
- BERNSTEIN, G. AND FUSSELL, D. 2009. Fast, exact, linear Booleans. *Computer Graphics Forum* 28, 5, 1269–1278.
- BO, W. 2009. Applications of 3D front tracking to multi phase fluid. Ph.D. thesis, Stony Brook University.
- BOTSCH, M. AND KOBELT, L. 2004. A remeshing approach to multi-resolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, New York, 185–192.
- BRAKKE, K. 1992. The surface evolver. *Experimental Mathematics* 1, 2, 141–165.
- BRAKKE, K. A. 2012. *Surface Evolver Manual*.
- BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH)* 21, 3, 594–603.

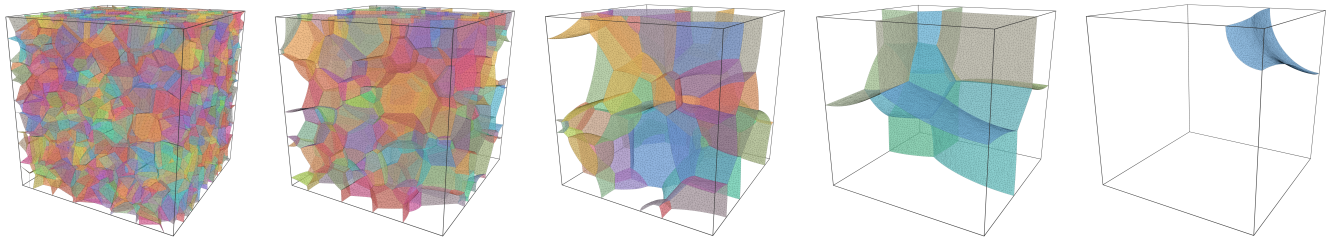


Fig. 24. **Mean curvature flow:** An initial collection of 2000 random Voronoi cells in a cube evolves by mean curvature flow, leading to a large number of complex topology changes.

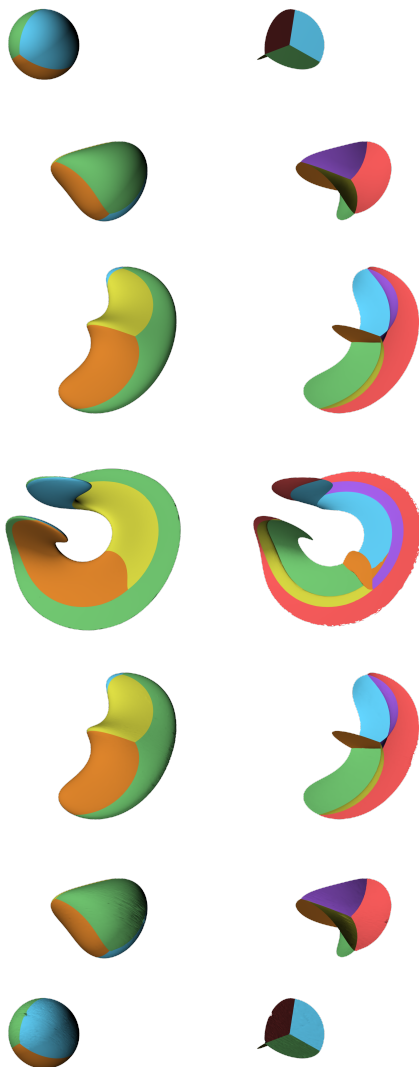


Fig. 20. **Multimaterial Enright test:** A sphere divided into four material regions is advected through the “Enright test” velocity field. The distinct regions are well-preserved despite the extreme stretching. Left: Exterior surface. Right: Interior interfaces.

- BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH)* 29, 4, 47.
- BROCHU, T. AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput.* 31, 4, 2472–2493.
- BROCHU, T., EDWARDS, E., AND BRIDSON, R. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. (SIGGRAPH)* 31, 4, 96.
- BRONSON, J. R., LEVINE, J. A., AND WHITAKER, R. T. 2012. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *International Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds. Springer, Berlin, 191–209.
- CAMPEN, M. AND KOBBELT, L. 2010. Exact and robust (self-)intersections for polygonal meshes. *Computer Graphics Forum (Eurographics)* 29, 2 (June), 397–406.
- CLARK, B., RAY, N., AND JIAO, X. 2012. Surface mesh optimization, adaptation, and untangling with high-order accuracy. In *International Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds. Springer, Berlin, 385–402.
- CLAUSEN, P., WICKE, M., SHEWCHUK, J. R., AND O’BRIEN, J. F. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph.* 32, 2, 17.
- DE GOES, F., GOLDSTEIN, S., AND VELHO, L. 2008. A simple and flexible framework to adapt dynamic meshes. *Computers and Graphics* 32, 2, 141–148.
- DESBRUN, M., MEYER, M., SCHRODER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 1999*. ACM, Los Angeles, 317–324.
- DU, J., FIX, B., GLIMM, J., JIA, X., LI, X., LI, Y., AND WU, L. 2006. A simple package for front tracking. *J. Comp. Phys.* 213, 2, 613–628.
- DUNYACH, M., VANDERHAEGHE, D., BARTHE, L., AND BOTSCH, M. 2013. Adaptive remeshing for real-time mesh deformation. In *Eurographics short papers*. Eurographics Association, Girona, Spain, 29–32.
- DYADECHKO, V. AND SHASHKOV, M. 2008. Reconstruction of multimaterial interfaces from moment data. *J. Comp. Phys.* 227, 11, 5361–5384.
- ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.* 183, 1, 83–116.
- GLIMM, J., GROVE, J., LINDQUIST, B., OLIVER A. MCBRYAN, AND TRYGGVASON, G. 1988. The bifurcation of tracked scalar waves. *Siam J. Stat. Sci. Comput.* 9, 1, 61–79.
- GLIMM, J., GROVE, J. W., LI, X., SHYUE, K.-M., ZENG, Y., AND ZHANG, Q. 1998. Three-dimensional front tracking. *SIAM J. Sci. Comput.* 19, 3, 703–727.

- GLIMM, J., GROVE, J. W., LI, X. L., AND TAN, D. C. 2000. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comput.* 21, 6, 2240–2256.
- HARLOW, F. H. AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids* 8, 12, 2182–2189.
- HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *ACM Trans. Graph. (SIGGRAPH)* 27, 3, 23.
- HIRT, C. W. AND NICHOLS, B. D. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comp. Phys.* 39, 1, 201–225.
- HUBELI, A. AND GROSS, M. 2000. Fairing of non-manifold models for visualization. In *Proceedings of Visualization '00*. IEEE Computer Society Press, Salt Lake City, Utah, USA, 407–414.
- JIAO, X. 2007. Face offsetting: A unified approach for explicit moving interfaces. *J. Comp. Phys.* 220, 2, 612–625.
- JIAO, X. AND ALEXANDER, P. J. 2005. Parallel feature-preserving mesh smoothing. In *Proceedings of the 2005 international conference on Computational Science and Its Applications - Volume Part IV*. Springer-Verlag, Singapore, 1180–1189.
- JIAO, X. AND BAYYANA, N. 2008. Identification of C1 and C2 discontinuities for surface meshes in CAD. *Computer Aided Design* 40, 2, 160–175.
- JIAO, X., COLOMBI, A., NI, X., AND HART, J. 2010. Anisotropic mesh adaptation for evolving triangulated surfaces. *Engineering with Computers* 26, 4, 363–376.
- KANAI, T., SUZUKI, H., AND KIMURA, F. 2000. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics and Applications* 20, 2, 62–75.
- KIM, B. 2010. Multiphase fluid simulation using regional level sets. *ACM Trans. Graph. (SIGGRAPH Asia)* 29, 6, 175.
- KIM, B., LIU, Y., LLAMAS, I., JIAO, X., AND ROSSIGNAC, J. 2007. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. (SIGGRAPH)* 26, 3, 98.
- LAZAR, E. 2011. The evolution of cellular structures via curvature flow. Ph.D. thesis, Columbia University.
- LAZAR, E., MASON, J., MACPHERSON, R., AND SROLOVITZ, D. 2012. A more accurate three-dimensional grain growth algorithm. *Acta Materialia* 59, 17, 6837–6847.
- LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH)* 25, 3, 812–819.
- LUCAS, B. C., KAZHDAN, M., AND TAYLOR, R. H. 2012. Multi-object spring level sets (MUSCLE). In *MICCAI*. Springer-Verlag, Nice, France, 495–503.
- MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In *Vis-Math*. Springer-Verlag, Berlin, Germany, 35–54.
- MISZTAL, M. AND BAERENTZEN, A. 2012. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.* 31, 3, 24.
- MISZTAL, M., ERLEBEN, K., BARGTEIL, A. W., CHRISTENSEN, B. B., BAERENTZEN, A., AND BRIDSON, R. 2012. Multiphase flow of immiscible fluids on unstructured moving meshes. In *Symposium on Computer Animation*. Eurographics Association, Lausanne, Switzerland, 97–106.
- MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. In *Symposium on Computer Animation*. ACM, New York, NY, USA, 237–245.
- MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. In *Symposium on Computer Animation*. ACM, Los Angeles, CA, USA, 237–244.
- NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6, 147.
- OSHER, S. AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York.
- PELLERIN, J., LÉVY, B., AND CAUMON, G. 2011. Topological control for isotropic remeshing of nonmanifold surfaces with varying resolution: application to 3D structural models. In *IAMG*. International Association of Mathematical Geosciences, Salzburg, Austria, 678–688.
- PONS, J.-P. AND BOISSONNAT, J.-D. 2007a. A Lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space. *Computer Graphics Forum* 26, 2, 227–239.
- PONS, J.-P. AND BOISSONNAT, J.-D. 2007b. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *CVPR*. IEEE, Minneapolis, Minnesota, USA, 1–8.
- REITICH, F. AND SONER, H. M. 1996. Three-phase boundary motions under constant velocities. I: The vanishing surface tension limit. *Proceedings of the Royal Society of Edinburgh* 126A, 837–865.
- SAYE, R. AND SETHIAN, J. 2012a. Analysis and applications of the Voronoi Implicit Interface Method. *J. Comp. Phys.* 231, 18, 6051–6085.
- SAYE, R. AND SETHIAN, J. 2012b. The Voronoi Implicit Interface Method for computing multiphase physics. *Proceedings of the National Academy of Sciences* 108, 49, 19498–19503.
- SOLENTHALER, B. AND PAJAROLA, R. 2008. Density contrast SPH interfaces. In *Symposium on Computer Animation*. Eurographics Association, Dublin, 211–218.
- STANCULESCU, L., CHAINE, R., AND CANI, M.-P. 2011. Freestyle: Sculpting meshes with self-adaptive topology. *Computers and Graphics* 35, 3, 614–622.
- UNVERDI, S. O. AND TRYGGVASON, G. 1992. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comp. Phys.* 100, 1, 25–37.
- WEAIRE, D. AND HUTZLER, S. 2001. *Physics of Foams*. Oxford University Press, New York.
- WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. (SIGGRAPH)* 29, 4, 49.
- WOJTAN, C., MULLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *SIGGRAPH Courses*. ACM, Vancouver, 8.
- WOJTAN, C., THUREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH)* 28, 3, 76.
- WOJTAN, C., THUREY, N., GROSS, M., AND TURK, G. 2010. Physically-inspired topology changes for thin fluid features. *ACM Trans. Graph. (SIGGRAPH)* 29, 3, 50.
- WU, Z. AND SULLIVAN, J. M. 2003. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering* 58, 2, 189–207.
- YING, L. AND ZORIN, D. 2001. Nonmanifold subdivision. In *Proceedings of Visualization '01*. IEEE, San Diego, CA, USA, 325–332.
- YUAN, Z., YU, Y., AND WANG, W. 2012. Object-space multiphase implicit functions. *ACM Trans. Graph. (SIGGRAPH)* 31, 4, 114.
- ZAHARESCU, A., BOYER, E., AND HORAUD, R. 2011. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE TPAMI* 33, 4, 823–837.
- ZHANG, Y., WANG, H., WANG, S., TONG, Y., AND ZHOU, K. 2012. A deformable surface model for real-time water drop animation. *IEEE TVCG* 18, 8, 1281–1289.
- ZHAO, H.-K., CHAN, T., MERRIMAN, B., AND OSHER, S. 1996. A variational level set approach to multiphase motion. *J. Comp. Phys.* 127, 1, 179–195.

ZHENG, W., YONG, J.-H., AND PAUL, J.-C. 2006. Simulation of bubbles. In *Symposium on Computer Animation*. Eurographics Association, Vienna, 325–333.

ZILSKE, M., LAMECKER, H., AND ZACHOW, S. 2008. Adaptive remeshing of non-manifold surfaces. In *Eurographics short papers*. Eurographics Association, Crete, Greece.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 1996*. ACM, New Orleans, 189–192.

APPENDIX

A. DETAILS OF MULTIMATERIAL MESH IMPROVEMENT

At each iteration, we perform passes of edge splitting, edge collapsing, edge flipping, and vertex smoothing. We cancel any operations that would introduce collisions, as well as those that would yield unacceptably poor quality angles, inverted normals, tiny-area triangles, and large volume loss; following Brochu and Bridson [2009] we expose the angle bounds, edge length bounds, minimum area bound, and volume change bound as user parameters, as these may be problem-dependent. These bounds are respected by our topological operations as well. (In particular, the maximum volume change bound should be chosen with this in mind; an overly strict limit can hinder merging.)

We detect feature edges by thresholding dihedral angles between triangle pairs sharing an edge [Botsch and Kobbelt 2004; Dunyach et al. 2013], rather than analyzing the local quadric metric tensor [Brochu and Bridson 2009; Jiao et al. 2010]; we found this to be more robust and intuitive to control, and it extends naturally to the non-manifold case. We used a threshold of 30° . Vertices lying on one or two feature edges are considered to belong to a feature curve or *ridge*, and vertices lying on three or more feature edges are identified as *peaks*.

A.1 Edge Flipping

The primary subtlety in performing edge flips is to ensure that triangle labeling remains correct, since two consistently labeled triangles sharing a manifold edge may nevertheless have opposing orientations. We pick one of the two incident triangles as a reference triangle, and use it to construct the resulting flipped two-triangle patch with the *same vertex winding order* (i.e., orientation) for both. The new triangles can then simply be assigned the label of the reference triangle. To preserve sharp features we disallow flipping of feature edges, though we do allow flipping of smooth *non-feature* edges that connect two feature vertices. We do not perform flipping on non-manifold edges, as this operation is not well-defined. Collision-checking follows Brochu and Bridson [2009].

Rather than flipping based on a Delaunay(-like) criterion, we follow Botsch and collaborators in seeking a mesh with more regular connectivity [Botsch and Kobbelt 2004; Dunyach et al. 2013]. That is, we perform flips that drive valences towards six for interior manifold vertices and four for boundary vertices. To handle manifold patches that border on non-manifold edges or vertices, we conceptually trim off the incident non-manifold geometry for the purposes of valence-counting, so that non-manifold vertices are treated as boundary vertices (i.e., with an ideal valence of four). We perform the flip if it reduces the total least squares difference between the

ideal valences and the current valences.

$$\min \sum_{i=1}^4 (\deg(v_i) - \deg(v_i)^{opt})^2 \quad (1)$$

In the above, v_i is one of the four vertices of the two-triangle patch, \deg indicates the vertex valence (or degree), and $\deg(v_i)^{opt}$ is the optimal valence of v_i (either four or six).

A.2 Edge Splitting and Collapsing

Splitting and collapsing of non-manifold edges are straightforward extensions of the manifold case (Figure 25), and can be checked for collisions in the same way [Brochu and Bridson 2009]. We use an upper and lower edge-length bound to trigger splitting and collapsing, respectively. Child triangles created by an edge split inherit the labels of their parents. Edge collapses do not require relabeling, since the two triangles bordering the edge are deleted and the surrounding labels remain correct.

To generate new vertex positions when splitting or collapsing, we use the modified butterfly scheme, as previous authors have advocated [Brochu and Bridson 2009; Wojtan et al. 2010]. We extend this scheme to treat both non-manifold edges and feature edges in exactly the same manner as boundary edges [Zorin et al. 1996]. This has two benefits. First, sequences of non-manifold or feature edges are subdivided by fitting a smooth cubic curve, thereby better preserving their shape. Second, a smooth region separated by a non-manifold curve or sharp feature curve from an adjacent smooth region uses information only from the “same side” to perform subdivision; just like in the boundary edge case, reflection is used to derive ghost-data for “missing” triangles. This preserves the smoothness of patches on either side, such as in the case of the sharp intersection curve produced by the merging of two spheres (Figure 15). Similarly, for situations in which two or more smooth patches are connected at only a single non-manifold vertex (but no edges), each patch is treated as a distinct manifold.

When collapsing an edge to a point, we can choose the position of the final point to be either one of the existing endpoints or a new point computed using subdivision. In smooth regions, subdivision is preferred, whereas near sharp features selecting one of the end points better maintains the shape. Following Brochu and Bridson [2009], we preserve peak vertices over ridge vertices, and ridge vertices over smooth non-feature vertices. In our non-manifold setting, when two vertices have the same feature type, we prefer to keep a non-manifold vertex over a manifold one. If two vertices are equivalent in terms of features and manifoldness, we simply use the midpoint. Taken together, these choices minimize the disruption of features and non-manifold boundaries that collapses can otherwise cause.

A.3 Vertex Smoothing

We apply smoothing of vertex positions to improve the shape of mesh triangles. Because naïve Laplacian smoothing tends to rapidly destroy volume, particularly in high curvature regions, we follow previous authors in applying *tangential* or *null-space* smoothing, which removes the normal component of the displacement induced by smoothing [Botsch and Kobbelt 2004; Jiao and Alexander 2005; Jiao et al. 2010]. For feature ridge vertices, smoothing-induced displacement perpendicular to the ridge direction is projected out instead so that smoothing occurs only along the ridge. For peak vertices, no smoothing is applied. We compute the vertex normal and ridge direction as in the work of Jiao and

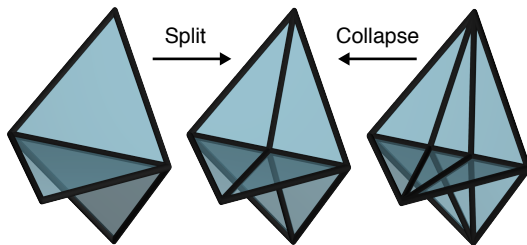


Fig. 25. **Non-manifold edge splits and collapses** closely parallel their manifold counterparts.

Bayyana [2008]. No special treatment is required for non-manifold geometry.

For strongly folded geometry (dihedral angles exceeding 165°) we instead perform Laplacian vertex smoothing in the average plane of the fold. The goal is to widen the angle at the fold, under the assumption that such sharp folds correspond to a crease that is in the process of merging. Treating this case with smoothing rather than relying entirely on collision-induced merging produces smoother intersection curves, for example during the initial collisions of spheres undergoing normal flow (Figure 22).

Smoothing yields an updated position for each vertex; these displacements are treated as pseudo-motions allowing collisions to be detected and resolved in the same manner as time integration [Brochu and Bridson 2009].

A.4 Eliminating Very Poor Triangles

Individual remeshing operations are canceled if they damage features or violate bounds on volume change, areas, angles, or edge lengths, as described by Brochu and Bridson [2009]. However, in complex scenarios these potentially conflicting constraints can limit the remesher's ability to resolve poor triangles. For example, eliminating a poor quality triangle may require smoothing a vertex in a manner that damages a feature, or performing a collapse that temporarily introduces a very small angle. Therefore, if triangles with very poor angles (e.g., outside $[2^\circ, 178^\circ]$) remain after our standard remeshing pass, we apply a more aggressive strategy: we apply our remeshing operations on those elements alone while ignoring all constraints beyond intersection-safety. This infrequently invoked fail-safe is effective at eliminating poor triangles at the cost of additional localized regularization.