Multimaterial Mesh-Based Surface Tracking

Fang Da* Columbia University
Christopher Batty† University of Waterloo
Eitan Grinspun‡ Columbia University

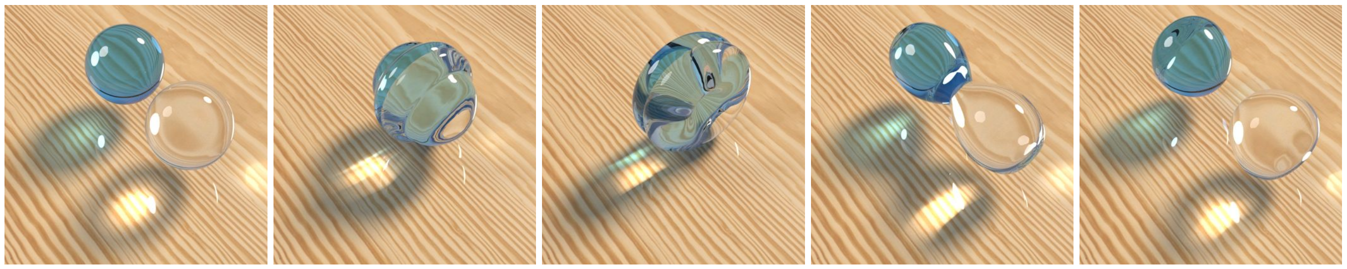


Figure 1: ***Two-Droplet Collision:*** *Using our multimaterial mesh-based surface tracker, two immiscible liquid droplets with different materials but identical physical properties impact symmetrically in zero gravity under strong surface tension. The collision merges the droplets so that a new interface separates the two liquids, and a non-manifold triple-curve is created where the two liquids meet the ambient air.*

Abstract

We present a triangle mesh-based technique for tracking the evolution of three-dimensional *multimaterial interfaces* undergoing complex deformations. It is the first non-manifold triangle mesh tracking method to simultaneously maintain intersection-free meshes and support the proposed broad set of multimaterial remeshing and topological operations. We represent the interface as a non-manifold triangle mesh with material labels assigned to each half-face to distinguish volumetric regions. Starting from proposed application-dependent vertex velocities, we deform the mesh, seeking a non-intersecting, watertight solution. This goal necessitates development of various collision-safe, label-aware non-manifold mesh operations: multimaterial mesh improvement; T1 and T2 processes, topological transitions arising in foam dynamics and multiphase flows; and multimaterial merging, in which a new interface is created between colliding materials. We demonstrate the robustness and effectiveness of our approach on a range of scenarios including geometric flows and multiphase fluid animation.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation;

Keywords: surface tracking, topology change, multimaterial flows, nonmanifold meshes

Links: DL PDF WEB VIDEO CODE

*e-mail:fang@cs.columbia.edu
†e-mail:christopher.batty@uwaterloo.ca
‡e-mail:eitan@cs.columbia.edu

1 Introduction

Mesh-based surface tracking iteratively advances a triangle mesh to capture the temporal evolution of dynamic interfaces. Such interfaces, often employed in fluid animation and geometric modeling, separate distinct materials that are deforming and undergoing topology changes. Mesh-based approaches have been touted for their ability to preserve volume, mesh-scale detail, and sharp features [Wojtan et al. 2011], and robust strategies have been proposed to perform topological merging and splitting: Boolean-like geometric operations [Campen and Kobbelt 2010], hybrid implicit/explicit approaches [Müller 2009; Wojtan et al. 2009], and local zippering/pinching combined with robust collision resolution [Brochu and Bridson 2009]. These approaches are restricted to interfaces separating exactly two material regions: an exterior and an interior.

Less explored in the graphics literature are *multimaterial* mesh interfaces, separating *multiple* distinct materials. This introduces non-manifold triple- and higher-order junctions, expanding the space of possible mesh entanglements, giving rise to new topological operations, and opening a rich space of research problems.

Stepping into the multimaterial setting, we address a suite of non-manifold mesh operations: (1) merging of like materials, (2) splitting, (3) mesh improvement, (4) merging of different materials, and the so-called (5) T1 and (6) T2 processes of foam dynamics [Weaire and Hutzler 2001]. Operations (4)–(6), which arise only for multimaterial cases, will be the focus of our discussion.

We build on past work that addresses subsets of our goal. Multimaterial codes exist that support some of these operations, but they ignore collisions. Those collision-aware codes noted above are limited to manifold meshes. Alas, staggering or gluing available codes does not offer the best of both worlds; it is unclear how to adapt known multimaterial operations for collision-safety. Furthermore, merging in multimaterial settings—an outcome of considering both collisions *and* non-manifold structure—has never been addressed.

To our knowledge, no triangle mesh-based surface tracking method *simultaneously* preserves watertight intersection-free meshes, *and* supports the above suite of operations. Our contribution is to develop such a unified, robust, multimaterial mesh-based surface tracking scheme, and demonstrate its effectiveness.

2 Related Work

2.1 Multimaterial Surface Tracking Methods

Level set methods [Sethian 1999; Osher and Fedkiw 2002] 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 2010; Saye and Sethian 2012]. In some methods one signed distance field is used per region, while other methods reduce storage and computational costs by using a single unsigned distance field for all regions. Starinshak et al. [2014] discuss one challenge specific to multimaterial level set methods—overlaps or vacuums at the triple junctions—which is typically corrected via projection. By contrast, our non-manifold mesh-based approach explicitly tracks such triple-junctions, avoiding vacuums/overlaps by construction. Previous work in mesh-based surface tracking has explored trade-offs between explicit and implicit approaches in the standard two-material setting (e.g., [Du et al. 2006; Wojtan et al. 2009]).

Particle-based surface representations augment each particle with a material label or color [Müller et al. 2005; Solenthaler and Pajarola 2008], or assign material properties to particles directly.

Volume-of-fluid methods [Hirt and Nichols 1981] store a volume fraction per cell, and their multimaterial generalizations store a partition of unity (one fraction *per material*); various approaches exist to reconstruct continuous interface geometry from this data [Dyadechko and Shashkov 2008; Anderson et al. 2010].

Moving mesh or Lagrangian volumetric mesh methods [Quan and Schmidt 2007; Pons and Boissonnat 2007a; Pons and Boissonnat 2007b; Quan et al. 2009; Misztal et al. 2012] assign material labels to each *volume* element. For example, for a tetrahedralization of space, the interface is the subset of triangular faces bordering two differently labeled tetrahedra. Several recent works address mesh quality maintenance [Wicke et al. 2010; Clausen et al. 2013].

2.2 Triangle Meshes with Merging and Splitting

Mesh Surgery / Boolean Approaches. Topology changes are recognized as a challenge for surface mesh evolution, or "front tracking" [Glimm et al. 1998; Glimm et al. 2000]. Colliding surfaces entangle and must be stitched via *mesh surgery* or *Boolean*(-like) operations [Campen and Kobbelt 2010; Zaharescu et al. 2011; Bernstein and Wojtan 2013]. Existing two-phase solutions do not map directly to the multimaterial setting, where new topological changes arise. For instance, the collision of two different materials immersed in a third requires that a separating interface be established; this interface is not a component of any existing surface. Bernstein and Wojtan [2013] specifically highlight this limitation of their method, noting that a broader set of fundamentally different topological operations may be required for some applications.

Hybrid Implicit Approaches sidestep the requisite mesh operations by converting colliding regions into implicit (e.g., Eulerian) representations, either globally or locally, and then reconstructing (e.g., via marching cubes) a new mesh [Glimm et al. 2000; Du et al. 2006; Bargteil et al. 2006; Müller 2009; Wojtan et al. 2009; Wojtan et al. 2010]. In computational physics the best known is the *FronTier* package [Du et al. 2006]. These methods rely on a binary inside/outside classification, and a multimaterial extension appears nontrivial.

Proximity-Based Merging locally stitches proximate meshes, maintaining an *intersection-free invariant* via either conservative bounds on time step [Stanculescu et al. 2011] or robust post-collision processing [Brochu and Bridson 2009]. The reliance on a kernel of simple, local mesh operations led us to extend this line of work to multiple materials, building on *El Topo* [Brochu and Bridson 2009].

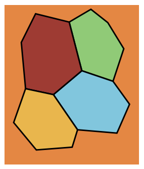
2.3 Triangle Mesh-Based Multimaterial Techniques

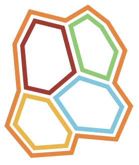
Surface Evolver [Brakke 1992], a popular software package, is a great source of inspiration for our work. The package uses an evolving non-manifold triangle mesh to find equilibria for myriad variational problems. It supports the foam and film-like topological transformations arising in many multimaterial settings, known as T1 and T2 processes (§4.4). We differ principally in (1) considering collisions, which induce new merging-type topology changes, and (2) ensuring intersection-safety throughout, which is difficult when topological transitions have special cases or affect large mesh neighborhoods. We develop a new approach for T1 processes (§5) that avoids special cases and, crucially, iteratively applies only a single local vertex operation.

Surface Evolver and related approaches have been applied to foam coarsening and grain growth problems in material sciences, for which collisions are often unimportant [Weygand and Brechet 1999; Wakai et al. 2000; Kuprat et al. 2003; Mora et al. 2008; Syha and Weygand 2010]. Most recently, Lazar et al. [2011] considered large-scale simulations of up to 100,000 distinct grains. Authors in graphics have also studied constant mean curvature surfaces using related techniques [Pan et al. 2012]. In general, these methods do not consider collision-induced merging or intersection-safety, and apply topological operations with larger mesh stencils.

3 Multimaterial Mesh Representation

Two-material mesh-based methods rely on a binary inside/outside classification of space. This classification may employ parity counting of ray casts, which does not readily extend to multiple materials; alternatively, it may employ a *consistent orientation* of mesh triangles: each triangle's vertex ordering and the right-hand rule define a normal direction, which by convention points to the interior. If each triangle is oriented consistently with its neighbors, and the mesh is watertight and non-self-intersecting, we have a strict binary classification that enables safe topological operations.





The non-manifold setting requires n-ary material classification, so mesh normals do not suffice. Instead, we follow previous authors [Brakke 1992; Yuan et al. 2012] in giving each material a unique integer label, and applying labels to the front *and* back of each triangle (i.e., each *half-face* is labeled). In this analogous 2D polygon example, colors indicate labels. The top image shows the represented material regions (including the "exterior" orange region), while the bottom image shows the corresponding mesh where edges each have two labels. This representation does *not* require consistent triangle orientation, even for triangle-pairs sharing a manifold edge. Instead, it requires *consistent material labels*: all half-faces bounding a closed region must be labeled identically. Maintaining and exploiting this extended notion of mesh orientation allows us to preserve watertight regions.

Throughout, the term *region* refers to a closed volume of space, while *material* refers to a region's type, indicated by its labels. Thus, two regions may be composed of the same material, but a region cannot consist of more than one material.

4 Method Overview

Beginning from an initially valid and non-self-intersecting multimaterial mesh, our overall approach proceeds as follows. The client application proposes a set of vertex trajectories, which we process with collision resolution techniques to recover an intersection-free state. We then perform merging, mesh improvement, and splitting, ensuring that each operation maintains intersection-safety. Algorithm 1 lays out the structure of our method.

Algorithm 1 Multimaterial Mesh-Based Surface Tracking

```
  Note: mesh is intersection-free after each checkmark (✓)
  while simulating do
      Advance vertices to predicted positions (§4.1)
      Resolve interpenetrations (§4.2)                    ✓
      Perform topological merging (§7)                    ✓
      Perform mesh improvement (§4.3)                     ✓
      Perform topological separation (§5 and §6)          ✓
  end while
```

4.1 Computing Vertex Positions

We first ask the domain-application to advance the mesh vertices to their proposed positions. A typical application will respond by time-integrating a physical equation or geometric flow, but ultimately we are agnostic to the source of the motion. The proposed trajectory may induce mesh intersections.

4.2 Resolving Interpenetrations

Next, we perturb the proposed trajectory to a *non-intersecting state*. We apply exact continuous collision detection (CCD) [Brochu et al. 2012], and resolve collisions using the method of Bridson et al. [2002] as extended by Harmon et al. [2008]. Our enforcement of an *intersection-free invariant* is crucial: subsequent steps rely on it to ensure that remeshing and topology changes are performed safely and successfully, following the general 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 collisions 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**; at all other times the mesh is intersection-free. Treating mesh edits atomically also implies that we should prefer fine-grained local edits, since these can be more cheaply collision-checked and succeed more often in the presence of tangled geometries.

4.3 Mesh Improvement

Strong mesh deformations necessitate remeshing to maintain triangle sizes and aspect ratios. We apply an incremental, feature-preserving scheme based on that of Brochu and Bridson [2009], which is collision-safe per the discussion above. Complete details, including extensions for non-manifoldness and multimaterial labeling, are in the supplemental material.

4.4 Multimaterial Topology Changes

Multimaterial scenarios in 3D undergo new topological transitions that do not arise with only two materials. We first discuss the 2D analogs to provide intuition (Figure 2).

Multimaterial Merging, discussed in §7, occurs when two distinct material regions collide while immersed in a third; the middle region divides in two, and a new interface maintains separation between the originally disjoint outer regions (Figure 2, top). This contrasts with the usual two-material case, where colliding regions always have the same material, and merge into one.

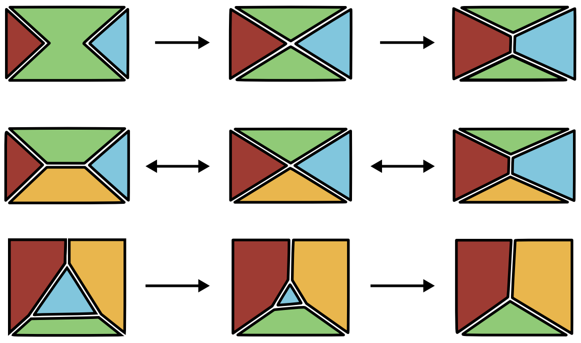


Figure 2: ***Multimaterial topology changes in 2D.*** *Merging (top): Two material regions separated by a third collide to yield a shared interface.* T1 process *(middle): Two regions that share a border separate while the remaining pair become connected.* T2 process *(bottom): One region (cyan) collapses away.*

T1 and T2 Processes are known from the literature on soap films and bubbles [Weaire and Hutzler 2001], and can describe how air bubbles (i.e., materials) within a connected soap foam can change their local connectivity under flow. In a *T1 process* two materials that originally share a border separate from one another while two different neighbouring materials become connected (Figure 2, middle). This forms a new interface while leaving the region count unchanged, and can only occur with four or more materials. In a *T2 process* a region collapses to a point and disappears as occurs commonly in convergent flows (Figure 2, bottom).

5 T1 Processes

Two Dimensions We analyze the behavior of a 2D T1 process by distinguishing two types of non-manifold mesh configurations: *regular* vertices with edge valences of two or three, and *irregular* vertices with edge valences above three (Figure 3). Irregular vertices are related to the degenerate case where two or more valence three (regular) vertices coincide, and infinitesimal perturbations would eliminate the degeneracy. In applications, irregular vertices usually exist only transiently or not at all. Meyer et al. [2008] discuss this notion in the context of multimaterial meshing, dubbing our regular vertices "generic" (as in *general position*) and treating irregular configurations by approximating with nearby regular ones. Similarly, Plateau's laws in 2D prohibit stable equilibria for soap films with valences above three [Weaire and Hutzler 2001].

However, as illustrated by Figure 2, middle, a T1 process requires passing directly through an irregular configuration: an irregular vertex is created by an edge collapse, and then separated or *resolved* into two regular vertices. In the figure, this allows red and cyan regions to connect while green and yellow regions disconnect. Without the explicit resolution mechanism we will propose, the incident regions remain artificially glued together at this lingering irregular vertex, regardless of the flow field. (By analogy with Lagrangian elastic simulations, this can be viewed as a kind of non-physical "locking" effect. In both cases, insufficient degrees of freedom prohibit the intended motion.)

A T1 process is perfectly reversible, so the separation direction must depend on the underlying physics (e.g., *Surface Evolver* examines surface tension forces on vertices [Brakke 1992]). For generality, we allow this decision to be application-dependent; typi-

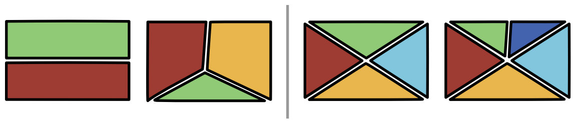


Figure 3: ***Regular and irregular vertex configurations in 2D:*** *Left: An interface separating two regions is regular, as is a triple-point vertex separating three regions. Right: Vertices with edge valence of four or higher are irregular.*

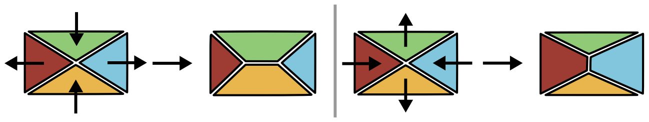


Figure 4: ***Choice of separation direction:*** *Resolution of an irregular vertex depends on the underlying flow field.*

cally, we will analyze the local velocity field (Figure 4).

Three Dimensions Our philosophy in 3D is identical: we allow irregular configurations to arise through collapsing of short edges during mesh improvement, and then separate them back into regular configurations as dictated by the flow.

Unfortunately, while a T1 process in 2D is fairly simple, a complete 3D T1 process involves many individual mesh operations (Figure 5). Rather than treat this situation with potentially fragile, special-case code, we show that **all irregular configurations can be treated in a simple and unified manner** by analyzing the mesh topology and iterating on a single, low-level *vertex separation* operation which we propose. Consistent with our overall approach, collision-safety can be assured by canceling individual operations that violate it. In theory this can delay topology changes, but we observed no artifacts across thousands of T1 processes.

Region Graphs Now consider the space of possible configurations about a vertex in 3D. As in 2D, the regular configurations (Figure 6, left) mirror the stable states described by Plateau's laws: a manifold surface by itself; three surfaces meeting along a triple curve; and four triple curves meeting at a point. However, in 3D both vertices *and edges* may have high valences, leading to diverse irregular configurations (Figure 6, right). Because visualizing 3D vertex configurations is difficult, we need a better tool to characterize their topology and determine how to resolve them. We will therefore define the concept of a *region graph* of a mesh vertex.

For brevity, we adopt the following incidence definitions. A vertex and an edge (resp. triangle) are incident if the vertex is one of the two (resp. three) vertices composing the edge (resp. triangle). A

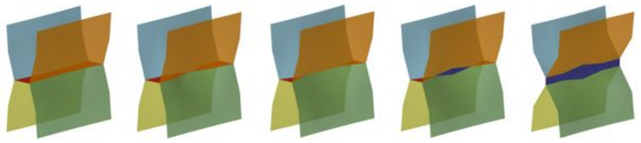


Figure 5: *A* ***3D T1 process*** *begins when a short edge between distinct triple-junction curves collapses to become a vertex incident on four materials. Adjacent edges on the original interface (red) also collapse yielding a quadruple-junction curve. Then, one vertex on the curve separates perpendicularly to create a new interface (dark blue). Nearby vertices follow suit to complete the process.*

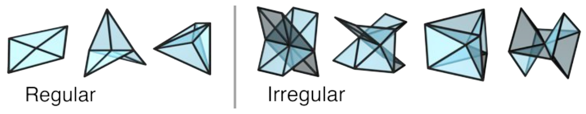


Figure 6: ***Regular and irregular configurations in 3D:*** *Left: (1) A manifold surface separating two regions (here, the top and bottom half-spaces), (2) a triple-curve bordering three regions, and (3) a quadruple-point at which four regions meet. Right: Four of the infinitely many possible irregular non-manifold configurations that must be resolved by vertex separation.*

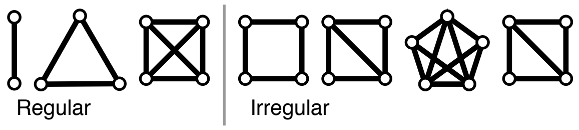


Figure 7: ***Region graphs:*** *Left: The region graphs for vertices at: (1) a two-region surface, (2) a 3-region curve, and (3) a 4-region junction, corresponding to Figure 6, left. Their graphs are* complete. *Right: The region graphs for the central vertices in the irregular configurations of Figure 6, right, are* incomplete.

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 *only if they share a triangle*; two regions joined only by a vertex or edge are *not* incident.

The **region graph** of a vertex v is an undirected graph in which each *node* corresponds to a region incident on v. Two graph nodes are joined by an *arc* if the two corresponding regions are incident. (We use the terms *node* and *arc* for graph elements, and *vertex* and *edge* for 3D mesh elements). Figure 7 shows the region graphs for the regular and irregular configurations in Figures 6. (This region graph is a subgraph of a larger dual (pseudo-)graph of the entire mesh, but per-vertex region graphs offer simpler visuals.)

Next, we make the key observation that the region graph for a regular vertex will be *completely connected* (Figure 7, left); all other graphs, which lack some arcs, correspond to irregular vertex configurations (Figure 7, right). Since graph arcs correspond to region incidence relationships, this implies that a vertex v is regular if every pair of regions incident on v is also *mutually* incident (share a face). Missing arcs therefore indicate pairs of regions that are not incident, an irregularity that we must correct. As suggested by Figures 8 and 9, this requires vertex separation: splitting apart the irregular vertex, and filling the gap with new geometry.

Vertex Separation As a concrete example, consider the central vertex v in Figure 9, left. Its region graph has a missing arc corresponding to the left and right (non-incident) regions, $\mathbf{A}$ and $\mathbf{B}$. We duplicate and pull the mesh vertex v apart into new vertices v_a and v_b. Their separation distance is set to a fraction of the average length of the surrounding edges, typically 10%. Triangles originally incident to region $\mathbf{A}$ are updated to use vertex v_a rather than v, by simply relabeling one of its vertex indices. Similarly, triangles incident to region $\mathbf{B}$ are updated to use v_b. All remaining triangles incident on v, but not on region $\mathbf{A}$ or $\mathbf{B}$, are updated to use v_b. Figure 9, middle, shows the mesh after it has been pulled apart.

This *may* yield a gap in the mesh that we need to fill, depending on the incident regions' material types. The gap's shape arises from the existing edges incident on v "opening" into triangular gaps, as the original vertex v splits in two. Therefore, for each edge vw incident on region $\mathbf{A}$, vertex separation creates a potential gap that we may fill with a new face $v_a v_b w$ (Figure 9, right).

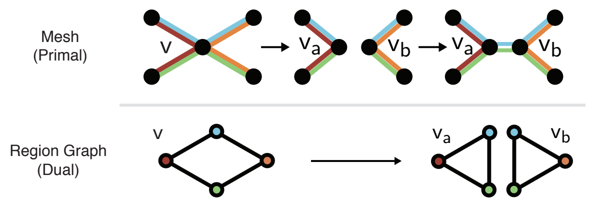


Figure 8: ***Vertex separation in 2D:*** *Top: An irregular vertex* v *in 2D (left) is separated into two regular vertices,* v_a *and* v_b *(middle); the resulting gap is filled by a new edge with appropriate labels (right). Bottom: The incomplete region graph of* v *is correspondingly converted into two complete region graphs for* v_a *and* v_b*.*

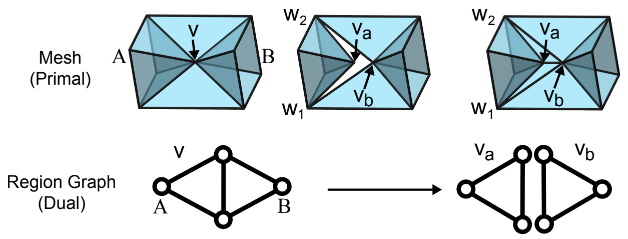


Figure 9: ***Vertex separation in 3D:*** *Top: An irregular vertex in 3D (left) is duplicated and separated (center), and the resulting gap is filled by new triangles with appropriate labels (right). Bottom: The incomplete region graph is replaced by two complete region graphs.*

However, we do not always need to fill the resulting gap with triangles. In some cases, adding these triangles would erroneously separate regions consisting of the same material; i.e., the new triangle would have the same label on both sides, making it redundant. (Consider a 3-region variant of Figure 9 where the top and bottom middle triangles are absent; our supplementary video illustrates this case.) This can be detected in advance by examining the two triangles incident on the edge vw and the region $\mathbf{A}$. If the two triangles' labels on the *outside* of region $\mathbf{A}$ differ, the triangle must be created to maintain the separation of these materials.

Returning to the topological view, Figure 9, 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 subsumes the two-material pinching of "singular" non-manifold vertices of Brochu and Bridson [2009].

Resolving Complex Irregular Vertices Vertex separation can be performed on any irregular vertex, but when its original region graph has multiple missing arcs (i.e., multiple pairs of regions are not mutually incident), one vertex separation may not suffice; v_a or v_b can remain irregular. This can be solved by iterating vertex separation as we discuss next. We will require a decision on which arc to process at each step, but for the moment, consider picking one missing arc arbitrarily at each step.

After applying vertex separation, each of the two new vertices' region graphs may still have missing arcs. However, these region graphs will have strictly fewer nodes, since the region graph for v_a does *not* contain node $\mathbf{B}$, and *vice versa*. Since vertex separation only applies to irregular vertices, this monotonic decrease in nodes per region graph halts when a vertex becomes regular with either 3 or 4 regions (nodes). Therefore, we place any new irregular vertices back into a global queue to be processed, assured that iteration

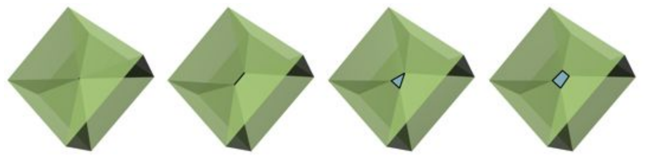


Figure 10: ***Complex vertex resolution:*** *This geometry's central irregular vertex is incident on six regions, and has a region graph with multiple disconnected node pairs. A sequence of three vertex separations are needed to resolve this case. The result is a new interface (blue) between the front and back regions, bounded by four regular vertices. Region graphs are shown in Figure 11.*

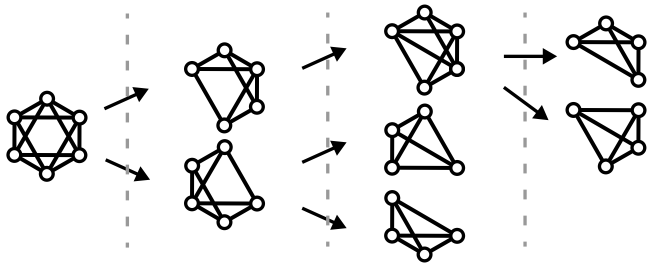


Figure 11: ***Complex vertex region graphs:*** *The sequence of region graphs corresponding to Figure 10. The original irregular vertex is split into two irregular vertices, each incident on five regions. Each of these vertices are split into two regular vertices, each incident on four regions. While the top vertex is not directly involved in the mesh edits in the second step, its region graph* is *modified because the newly created interface connects the front and back regions.*

will terminate within finite steps with a set of regular vertex configurations. Figures 10 and 11 show a vertex initially incident on six regions that requires three vertex separations, and the corresponding region graph sequence, respectively.

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 should be determined by the underlying flow. For each pair $\{\mathbf{A}, \mathbf{B}\}$, we first compute the direction of separation as the vector between the centroids of the one-ring vertices of v that are incident to either region. For ambient velocity fields, we then take the dot product between the directional derivative of the velocity with the direction of separation, which indicates how strongly the velocity field is diverging along that direction, i.e., how strongly the pair desires separation. We compute this *separation strength* (either analytically or with finite differencing) for each candidate pair, selecting the one with the highest value for processing. (If the separation strength in a given direction is zero, no vertex separation is performed.) Algorithm 2 summarizes the complete process.

For generality, we can support alternate measures of separation strength. Mean curvature flows seek to reduce surface area, so we examine the change in mesh surface area for each proposed vertex separation, selecting the one which most reduces the area. Our normal flows tests do not give rise to T1 processes (§8.2).

Vertex separation is performed in descending order of separation strength over *all* irregular vertices. Because edits to the mesh can slightly change its shape and resulting behavior, at the discrete level a complex T1 process may depend on the outcome of others nearby. This global sort assures that irregular vertices are processed in a consistent order, so that the outcome depends on physical quantities rather than implementation-specific vertex ordering. This also minimizes temporal flickering due to the mesh alternating among

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 $\mathbf{d}_{AB}$
            $t_{AB} \leftarrow$ 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
        $t_{AB} \leftarrow$ 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 if needed
        end for
    end for
end while
```

nearby configurations. While more costly than greedily separating in the optimal direction for each vertex individually, irregular vertices typically comprise a small and sparse subset of the mesh.

Consistency with Edge Collapses Since edge collapse and vertex separation are essentially dual operations, their triggering criteria should be consistent. Otherwise, the same irregular vertex may be repeatedly created and destroyed causing temporal coherence issues. Therefore, when performing a collapse of a short edge that would create an irregular vertex, we also check if the velocity field is acting to decrease the edge length. If not, this indicates that the physics is driving the geometry away from the potential T1 process, so the instigating edge collapse is cancelled.

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 CCD checks to identify any collisions 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 $\mathbf{A}$ 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.

In addition to colliding with the rest of the mesh, triangles and edges moved in the second pseudo-motion can end up intersecting the geometry they were pulled apart from. This cannot be tested via CCD since these elements are 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 $\mathbf{A}$ are tested against all mesh edges, and the final positions of edges incident on $\mathbf{A}$ are tested against all mesh triangles. This will guarantee no intersections, but could potentially allow tunneling of very small components. We rule this out by 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.

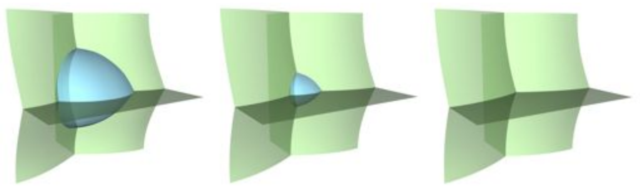
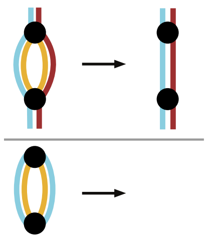


Figure 12: *In a* ***3D T2 process****, one region (blue) collapses away.*

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

6 T2 Processes

We now consider the simpler T2 process in which a region shrinks until it disappears, as occurs in convergent velocity fields (Figure 12). In the discrete case, a closed tetrahedron shaped region may undergo an edge collapse or edge flip during remeshing 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.



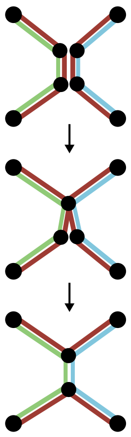
To resolve this degeneracy, we detect if the outer regions of the two triangles have different material labels; if so, we delete *one* of the two triangles and relabel the other, effectively removing the degenerate region while leaving in place a separating interface between materials (top). Otherwise, the outer regions have the same material and should be connected, so we simply delete both triangles (bottom). The latter subsumes the two-material analog described by Brochu and Bridson [2009].

Degeneracy removal also provides a second natural splitting mechanism. If thread-like geometry becomes very slender its tetrahedra collapse and are removed, without the explicit detection or cutting of spindles (e.g., [Wojtan et al. 2010]).

7 Multimaterial Merging

T1 and T2 processes involve regions that are locally connected by the non-manifold mesh. Multimaterial surface tracking also requires the ability to handle collision-induced topology changes between regions that are initially disjoint, either merging them into one or establishing a new separating interface. Like previous authors [Brochu and Bridson 2009; Stanculescu et al. 2011], we initiate merging when geometry comes within a user-defined proximity threshold. However, these existing two-phase approaches depend on a local zippering operation. While we adapted zippering to multiple materials (detailed in the supplemental material), in practice it requires well-aligned geometry to avoid causing intersections. Hence many zippering operations are canceled, preventing merging and causing lingering interface noise (cf. [Brochu et al. 2010]).

7.1 Snap-Based Merging



We propose a new merging strategy based on *snapping* together of nearby vertices. Our snapping operation is identical to an edge collapse, which coalesces two existing vertices, except that the original vertices do not share an edge. This simple operation can lead to more effective merging.

Ideal Snapping Consider first an idealized scenario in which two perfectly aligned triangles approach 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. Degeneracy removal (§6) deletes one or both triangles to complete the merge. The figure illustrates the 2D analog.

Generalized Snapping Since meshes rarely collide in ideal alignment, we generalize snapping taking loose inspiration from compatible remeshing (e.g., [Alexa 2002]). Concretely, we seek to modify the nearby meshes such that there *are* readily snappable vertex pairs. We identify close 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 trisected at the closest point to the vertex (face splits can be collision-checked similar to an edge split). Each vertex now has a counterpart on the opposing mesh, and snapping proceeds as in the idealized setting. Figure 13 shows the analogous 2D process.

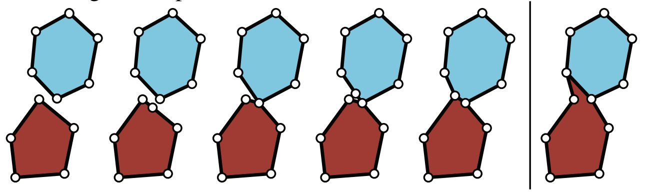


Figure 13: ***Generalized snapping in 2D:*** *Left: For nearby meshes, an edge in a proximate edge-vertex pair is split at the closest point, and the resulting vertex pair is snapped. Similarly snapping a second edge-vertex pair completes the merge. For matching materials, the separating edge would then be deleted. Right: A zipper-based merge is more disruptive and adds more volume.*

Because this split-and-snap strategy uses smaller, incremental edits to the non-manifold mesh rather than zippering's simultaneous creation of a manifold tube, each edit is less likely to be halted by collisions. Furthermore, our method actively modifies the mesh to enable merging, whereas zippering relies on encountering a feasible state by chance, which becomes unlikely in tangled collisions. Accordingly, snapping offers more effective merging.

Implementation Details When snapping a particular vertex pair, we place the newly snapped vertex at the average position of the original vertices. Individual splitting and snapping operations are checked for collision-safety and the introduction of degenerate geometry exactly as for edge splits and collapses, respectively, and canceled if necessary (see supplemental material). 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 perform vertex-vertex snapping directly. Likewise, if a triangle would be subdivided close to one of its edges, we split the edge instead, and snap using its new vertex. We do not snap vertices that share an edge since that is an edge collapse. We allow snapping between components of triangles that share an edge only if the angle between the triangles is less than 90°. This allows snapping of folded geometry, which occurs often during collisions, while preventing snapping in the plane of a triangle.

8 Discussion and Results

Recapping, our method can be seen to handle the entire targeted suite of high-level tasks, using a concise set of local, collision-checked, low-level transformations: edge flip, edge split, edge collapse, face split, vertex smoothing, vertex snapping, vertex separation, and degenerate face removal. In addition, the use of exact CCD to detect and cancel unsafe operations further *guarantees* that our topological changes never introduce intersecting meshes.

We are making the source code of our C++ implementation, dubbed *Los Topos*, available in order to encourage application of our method and to foster further research. It can be downloaded at `http://www.cs.columbia.edu/cg/multitracker/`.

Below, we examine the capabilities of our method on several complex multimaterial mesh flows. We report the results of two scaling tests using our implementation in the supplemental material. To give an example, it took 4 hours to run our largest mean curvature flow test to completion (2200 frames), which involves 2000 initial regions and 220K triangles, on an Intel Xeon 3.47GHz machine with 24GB RAM.

Experiments consistent with **first order convergence** are documented in a technical report [Da et al. 2014].

8.1 Prescribed Velocity Flows

We start with multimaterial variants of two traditional stress tests: the Zalesk sphere and Enright tests [Enright et al. 2002]. In the Zalesak sphere test a notched sphere is rotated through 360° about an external point to test sharp feature preservation under rigid body motions, a traditional strength of meshes. In the Enright test a sphere is advected through a highly deforming velocity field to induce very thin features. We used fourth-order Runge-Kutta to compute vertex trajectories from the ambient vector fields.

Our multimaterial Zalesak disk test (see supplementary video) exhibits no perceptible smoothing or lost volume. Our multimaterial Enright test (Figure 14) preserves thin features even under extreme stretching in multi-layered regions. While gradually accumulated position error due to time integration and remeshing manifests as mild "wrinkling" on the final sphere, this is consistent with *El Topo*'s results [Brochu and Bridson 2009] and difficult to avoid for such purely passive flows. In a simulation scenario, this effect should be eliminated either through problem-dependent physical processes or additional regularization [Bojsen-Hansen and Wojtan 2013]. This effect has been repeatedly observed in computational physics settings too, and treated by volume-preserving smoothing or "undulation removal" [de Sousa et al. 2004; McKee et al. 2008; Toutant et al. 2012].

Alternating steps of outward normal flow and the curl noise velocity field of Brochu and Bridson [2009] yields a stress test featuring both substantial stretching and merging. We applied this process

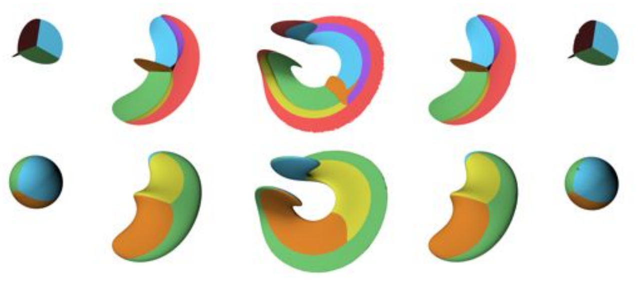


Figure 14: ***Multimaterial Enright test:*** *A sphere divided into four material regions is advected through the "Enright test" velocity field. The distinct regions are well-preserved despite the extreme stretching. Top: Interior interfaces. Bottom: Exterior interfaces.*

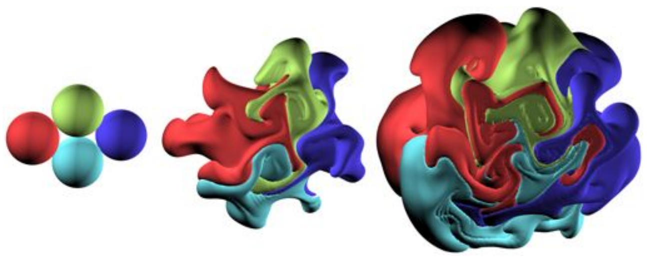


Figure 15: ***Normal Flow + Curl Noise:*** *Alternating steps of normal flow and curl noise applied to four spheres yield complex deformations and topology changes.*

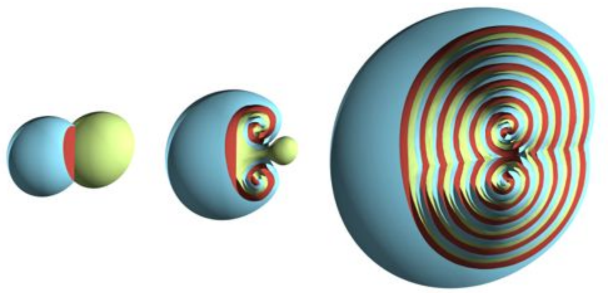


Figure 16: ***Cyclical normal flow:*** *Two overlapping spheres sharing a circular internal interface undergo normal flow with a cyclical ordering, with the front cut away for visualization. The result is a curling effect around the stationary triple curve.*

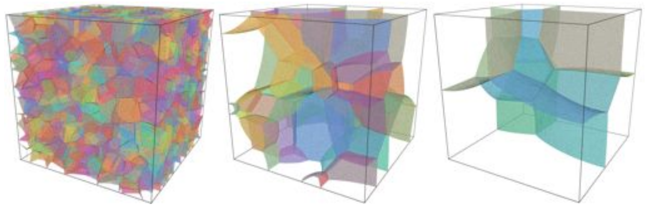


Figure 17: ***Mean curvature flow:*** *A collection of 2000 random Voronoi cells undergoes many T1 and T2 topology changes.*

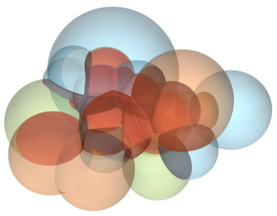
to four spheres of different materials, which resulted in the complex geometry in Figure 15. For this test, we performed normal flow by computing area-weighted vertex normals (normalized average of incident triangle normals, weighted by area) and applying a constant offset distance (i.e., forward Euler).

The final prescribed velocity flow in our video is a *single-phase* grid-based fluid flow inside a cubic domain. External forcing is applied to induce a swirling motion on a collection of 27 spheres with various material labels being passively advected using second-order Runge-Kutta.

8.2 Geometric Flows

Geometric surface flows have various applications within computer graphics [Eckstein et al. 2007; Pan et al. 2012; Crane et al. 2013]. We consider multimaterial normal and mean curvature flows.

Normal Flows To examine merging in more detail, we apply multimaterial normal flow on two special cases of interfacial speed functions. 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. Our video shows two expanding spheres colliding, 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, feature-preserving remeshing strategy. When the direction of flow is reversed, the spheres seamlessly shrink until finally pinching off and disappearing. The figure shows the result of normal flow applied to 25 initially disjoint spheres of various materials (colors). Mixed ma-



terial merging produces interior interfaces, while matching materials merge into one region. Reversing the direction of flow causes the interfaces to smoothly shrink and disappear. Our video contains an example of two inflating Stanford bunnies that maintain good surface detail.

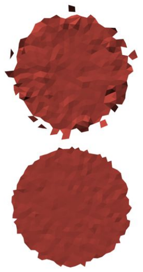
The second special case of normal flow we consider follows an example by Saye and Sethian [2012]. A set of three interfaces move at constant speed in the normal direction, with the signs of the velocities chosen to satisfy a cyclical ordering; that is, material A flows into material B, B flows into C, and C flows into A. The initial configuration is two overlapping spheres sharing a flat circular interior interface bounded by a triple curve. This causes a curling motion around the stationary triple-curve (Figure 16).

We computed normal flow vertex trajectories using the *face offsetting method* (FOM) [Jiao 2007]. FOM alleviates artifacts in discretizations based on vertex normals, at the cost of a stringent time step restriction. Because FOM does not (yet) support *general* evolution of triple curves undergoing normal flow, these examples compute triple junction trajectories as special cases. (Multimaterial normal flow with arbitrary speeds and triple junctions is non-trivial even for level sets (e.g., [Zhao et al. 1996]), and has not been addressed for triangle meshes.) For the expanding and shrinking spheres, we compute triple-junction trajectories by considering only the outer manifold surface, ignoring the (stationary) interior interface. For cyclical normal flow, we treat the triple curve as a zero-velocity boundary, consistent with the true solution.

Mean Curvature Flows Figures 5 and 12 show basic T1 and T2 processes, driven by mean curvature flow. An arrangement of 2000 distinct materials evolving inside a cube (Figure 17) demonstrates robustness across thousands of topology changes.

We computed mean curvature flow vertex motions using the discretization of Meyer et al. [2002] with forward Euler. Based on minimization of surface area, it extends to non-manifold vertices simply by considering the area of all incident triangles.

8.3 Comparing Snapping and Zippering



To compare snap-based merging against zippering we consider two spheres merging under normal flow, with the resulting shared interfaces shown to the right. With zippering (top), the growing intersection curve and internal interface develop in a discontinuous and noisy fashion. By contrast, snapping (bottom) yields smoother growth of the new interface. The video shows that the outer geometry also exhibits greater overall symmetry and smoothness.

For these tests, we intentionally used normal flow based on *vertex normals*. This gives merging velocities more reflective of fluid or solid animations, which undergo many topological operations during collisions; by contrast, FOM generates vertex velocities that (combined with small time steps) avoid merge operations after initial contact. The latter yields better normal flows, as in §8.2, but does not adequately stress test complex merging.

8.4 Liquid Animation

We demonstrate full integration with a grid-based multiphase liquid solver [Losasso et al. 2006], in which the evolution of the interfaces is tightly coupled to the fluid physics. We first consider liquid spheres colliding in zero gravity with surface tension. Figure 1 shows a two-droplet case merging and separating. The simulation takes on average 48 seconds per frame, of which 8.6% is spent on surface tracking. Our video includes a three-droplet variation, in which two matching droplets briefly merge through the third surrounding droplet, briefly forming an unstable non-standard double bubble. Our third example (Figure 18) applies high viscosity (per Batty and Bridson [2008]) to the collapse of nine viscous Stanford bunnies falling in a heap under gravity; the various impacting interfaces merge as expected. Each frame takes 174 seconds, and surface tracking takes 23.8% of the total simulation time. Taking best advantage of mesh detail for liquids can require appropriate sub-grid physics, adaptive grids, or coupling with particles [Brochu et al. 2010; Yu et al. 2012; Bojsen-Hansen and Wojtan 2013], which has not yet been explored in the multimaterial setting. Following Thuerey et al. [2010], we applied local volume-preserving mean curvature flow to gradually regularize sub-grid geometric features, deferring further study of this question to future work.

9 Limitations and Future Work

While we focused on topology change, enhancements to remeshing would be welcome. In particular, we preserve intersection-safety, but it remains uncertain what, if any, absolute and simultaneous guarantees on triangle quality and feature preservation may be achievable. Spatially adaptive, anisotropic, or high-order remeshing could improve our results [Jiao et al. 2010; Narain et al. 2012; Clark et al. 2012]. Some edge-length scale popping or flickering can be visible when edits must modify vertices on nearby distinct feature curves or when folding occurs, such as in the curl noise example (Figure 15); however, merging level sets also "pop" at the grid scale.

Exact CCD only provably guarantees collision *detection*; floating point precision hinders true guarantees on *response* in degenerate scenarios [Brochu et al. 2012]. Hence our topological operations are truly safe or canceled, but impulsive response [Bridson et al. 2002; Harmon et al. 2008] applied after time integration has no theoretical guarantees; we observed no failures. Pervasive collision

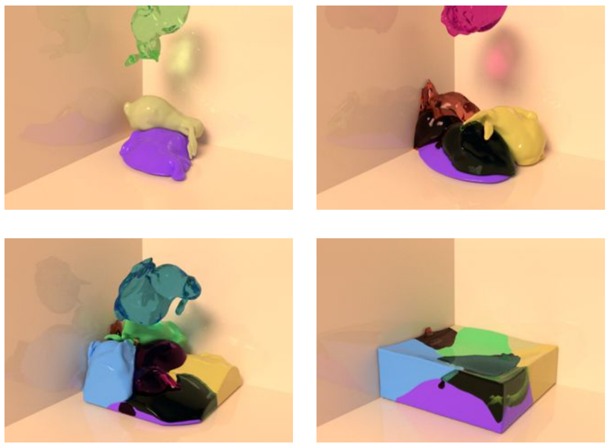


Figure 18: ***Viscous Bunnies:*** *A multiphase scenario in which nine viscous bunnies with different materials are dropped into a pile.*

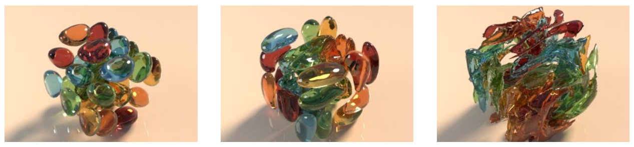


Figure 19: ***Swirling:*** *We passively advect 27 spheres through a single-phase fluid simulation in a cubic domain with a rotational forcing function applied to induce mixing.*

detection is also a bottleneck. We applied broad-phase culling via uniform grids; adaptive alternatives could offer speed-ups. The locality of our mesh edits also suggests parallelization opportunities.

Faithful modeling of soap films and bubble blowing requires *open* surfaces, for which surface meshes have strong natural advantages over implicit models. For example, Bernstein and Wojtan [2013] developed topological operations for geometric modeling with open surfaces. The core challenge in our setting is to eliminate the dependence on consistent region labels during topological operations.

The wide adoption of *Surface Evolver* and *FronTier* highlights the many domains beyond computer graphics that involve deforming interfaces, including material sciences and computational physics. Though outside the scope of the current work, we plan to explore these domains in the future, beginning with a thorough examination of convergence behavior [Da et al. 2014].

Acknowledgments

This work was supported in part by the NSF (Grants IIS-1319483, CMMI-1331499, IIS-1217904, IIS-1117257, CMMI-1129917, IIS-0916129), the Israel-US Binational Science Foundation, the Natural Sciences and Engineering Research Council of Canada (NSERC), Intel, The Walt Disney Company, Autodesk, Side Effects Software, NVIDIA, and the Banting Postdoctoral Fellowships program. We would like to thank Brochu et al. for releasing the *El Topo* source code [Brochu and Bridson 2009; Brochu et al. 2012], which helped to make our work possible. We would also like to thank Henrique Teles Maia, Nora Wixom and Rohan Sawhney for their assistance with the work.

References

ALEXA, M. 2002. Recent advances in mesh morphing. *Computer Graphics Forum 21*, 2, 173–198.

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.

BATTY, C., AND BRIDSON, R. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*, 219–228.

BERNSTEIN, G., AND WOJTAN, C. 2013. Putting holes in holey geometry: Topology change for arbitrary surfaces. *ACM Trans. Graph. (SIGGRAPH) 32*, 4, 34.

BOJSEN-HANSEN, M., AND WOJTAN, C. 2013. Liquid surface tracking with error compensation. *ACM Trans. Graph. (SIGGRAPH) 32*, 4, 79:1–79:10.

BRAKKE, K. 1992. The surface evolver. *Experimental Mathematics 1*, 2, 141–165.

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.

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4, 2472–2493.

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., EDWARDS, E., AND BRIDSON, R. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. (SIGGRAPH) 31*, 4, 96.

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, adaption, and untangling with high-order accuracy. In *International Meshing Roundtable*, Springer, Berlin, X. Jiao and J.-C. Weill, Eds., 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.

CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph. (SIGGRAPH) 32*, 4, 61.

DA, F., BATTY, C., AND GRINSPUN, E. 2014. A convergence study of multimaterial mesh-based surface tracking. Tech. rep., Columbia University.

DE SOUSA, F. S., MANGIAVACCHI, N., NONATO, L. G., CASTELO, A., TOMÉ, M. F., AND MCKEE, S. 2004. A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces. *J. Comp. Phys. 198*, 2, 469–499.

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.

DYADECHKO, V., AND SHASHKOV, M. 2008. Reconstruction of multi-material interfaces from moment data. *J. Comp. Phys. 227*, 11, 5361–5384.

ECKSTEIN, I., PONS, J.-P., TONG, Y., KUO, C.-C. J., AND DESBRUN, M. 2007. Generalized surface flows for mesh processing. In *Symposium on Geometry Processing*, 183–192.

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. 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.

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.

JIAO, X., COLOMBI, A., NI, X., AND HART, J. 2010. Anisotropic mesh adaptation for evolving triangulated surfaces. *Engineering with Computers 26*, 4, 363–376.

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

KIM, B. 2010. Multiphase fluid simulation using regional level sets. *ACM Trans. Graph. (SIGGRAPH Asia) 29*, 6, 175.

KUPRAT, A., GEORGE, D., STRAUB, G., AND DEMIREL, M. C. 2003. Modeling microstructure evolution in three dimensions with Grain3D and LaGriT. *Computational Materials Science 28*, 1, 199–208.

LAZAR, E. 2011. *The evolution of cellular structures via curvature flow*. PhD thesis, Columbia University.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH) 25*, 3, 812–819.

MCKEE, S., TOMÉ, M. F., FERREIRA, V. G., CUMINATO, J. A., CASTELO, A., DE SOUSA, F. S., AND MANGIAVACCHI, N. 2008. The MAC method. *Computers & Fluids 37*, 8 (Sept.), 907–930.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, Springer-Verlag, Berlin, Germany, 35–54.

MEYER, M., WHITAKER, R. T., KIRBY, R. M., LEDERGERBER, C., AND PFISTER, H. 2008. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE TVCG 14*, 6, 1539–1546.

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.

MORA, L. B., GOTTSTEIN, G., AND SCHVINDLERMAN, L. S. 2008. Three-dimensional grain growth: Analytical approaches and computer simulations. *Acta Materialia 56*, 1, 5915–5926.

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.

MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. In *Symposium on Computer Animation*, ACM, New York, NY, USA, 237–245.

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.

PAN, H., CHOI, Y.-K., LIU, Y., HU, W., DU, Q., POLTHIER, K., ZHANG, C., AND WANG, W. 2012. Robust modeling of constant mean curvature surfaces. *ACM Trans. Graph. (SIGGRAPH) 31*, 4, 85.

PONS, J.-P., AND BOISSONNAT, J.-D. 2007. 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. 2007. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *CVPR*, IEEE, Minneapolis, Minnesota, USA, 1–8.

QUAN, S., AND SCHMIDT, D. P. 2007. A moving mesh interface tracking method for 3D incompressible two-phase flows. *Journal of Computational Physics 221*, 2, 761–780.

QUAN, S., LOU, J., AND SCHMIDT, D. P. 2009. Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations. *Journal of Computational Physics 228*, 7, 2660–2675.

SAYE, R., AND SETHIAN, J. 2012. Analysis and applications of the Voronoi Implicit Interface Method. *J. Comp. Phys. 231*, 18, 6051–6085.

SETHIAN, J. 1999. *Level set methods and fast marching methods*. Cambridge University Press.

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.

STARINSHAK, D. P., KARNI, S., AND ROE, P. L. 2014. A new level set model for multimaterial flows. *J. Comp. Phys. In press*.

SYHA, M., AND WEYGAND, D. 2010. A generalized vertex dynamics model for grain growth in three dimensions. *Modelling Simul. Mater. Sci. Eng. 18*, 1, 015010.

THUEREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph. (SIGGRAPH) 29*, 3.

TOUTANT, A., MATHIEU, B., AND LEBAIGUE, O. 2012. Volume-conserving smoothing for front tracking methods. *Computers & Fluids 67*, 16–25.

WAKAI, F., ENOMOTO, N., AND OGAWA, H. 2000. Three-dimensional microstructural evolution in ideal grain growth - general statistics. *Acta Materialia 48*, 1, 1297–1311.

WEAIRE, D., AND HUTZLER, S. 2001. *Physics of Foams*. Oxford University Press, New York.

WEYGAND, D., AND BRECHET, Y. 1999. Three-dimensional grain growth: a vertex dynamics simulation. *Philosophical Magazine B 79*, 5, 703–716.

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., THUEREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 76.

WOJTAN, C., THUEREY, N., GROSS, M., AND TURK, G. 2010. Physically-inspired topology changes for thin fluid features. *ACM Trans. Graph. (SIGGRAPH) 29*, 3, 50.

WOJTAN, C., MULLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *SIGGRAPH Courses*, ACM, Vancouver, 8.

YU, J., WOJTAN, C., TURK, G., AND YAP, C. 2012. Explicit mesh surfaces for particle based fluids. *Computer Graphics Forum (Eurographics) 31*, 2, 815–824.

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.

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.