

Physics-Inspired Topology Changes for Thin Fluid Features

Chris Wojtan
Georgia Institute of Technology

Nils Thürey
ETH Zürich

Markus Gross
ETH Zürich

Greg Turk
Georgia Institute of Technology

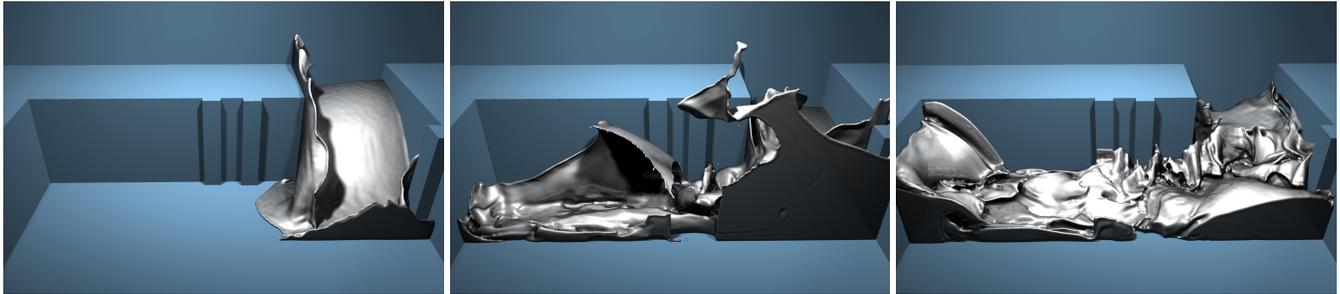


Figure 1: Our algorithm efficiently produces detailed thin sheets and liquid droplets, even with low-resolution fluid simulations — The main corridor in this example is only 30 fluid cells wide.

Abstract

We propose a mesh-based surface tracking method for fluid animation that both preserves fine surface details and robustly adjusts the topology of the surface in the presence of arbitrarily thin features like sheets and strands. We replace traditional re-sampling methods with a convex hull method for connecting surface features during topological changes. This technique permits arbitrarily thin fluid features with minimal re-sampling errors by reusing points from the original surface. We further reduce re-sampling artifacts with a subdivision-based mesh-stitching algorithm, and we use a higher order interpolating subdivision scheme to determine the location of any newly-created vertices. The resulting algorithm efficiently produces detailed fluid surfaces with arbitrarily thin features while maintaining a consistent topology with the underlying fluid simulation.

Keywords: surface tracking, topology changes, fluid dynamics, deforming meshes

1 Introduction

In the recent past, researchers have developed several impressive techniques for tracking the surface of a flowing liquid [Osher and Fedkiw 2002; Bargeil et al. 2006; Enright et al. 2002]. Traditionally, these surface tracking methods impose topological changes upon the fluid surface whenever surface features are smaller than the computational resolution. These topological changes cause droplets to pinch off, thin sheets to rupture, and small features to vanish when they become smaller than a predefined length.

In contrast, the topological behavior of liquid surfaces in the real world does *not* depend upon a single spatial threshold. Instead,

these splits and merges are closely linked to the geometry and motion of the fluid. For example, thin cylinders of liquid tend to break up into droplets in the presence of large surface tension; this phenomenon is known as a Rayleigh-Plateau instability, and it is caused by a positive feedback loop between the decreasing cylinder radius and increasing tension forces. As a result, liquid jets and spindles along the boundary of a liquid sheet will reliably break apart as soon as they become thin enough for the surface tension forces to dominate the inertia forces [De Gennes et al. 2004; De Luca and Costa 2009]. However, the main body of a liquid sheet is actually stabilized by surface tension forces, so it will resist rupturing until it becomes microscopically thin and torn apart by van der Waals forces [Ivanov 1988]. Thin sheets of air, on the other hand, tend to immediately break up due to a Rayleigh-Taylor instability when surrounded by bodies of denser liquid [Rayleigh 1883].

Based on these observations, we believe it is physically incorrect to induce topological splits and merges based on a single grid resolution. In order to faithfully recreate the phenomena mentioned in the previous paragraph, we have devised the following physically-inspired rules for the topological persistence of fluid features in an animation:

1. Thin sheets of air should be quickly deleted.
2. Thin sheets of liquid should never rupture.
3. Thin spindles of liquid should break up at a scale defined by the animator.

In this paper, we will use these guidelines to develop a physically-inspired model for handling topological changes in liquid simulations.

Our work begins with a method for explicitly tracking a fluid surface, similar to the methods of Du et al. [2006] and Wojtan et al. [2009]. These techniques deform a high-resolution surface mesh and locally detect and correct topological inconsistencies by re-sampling from a grid-based signed distance function. This tactic successfully preserves high resolution surface details in topologically simple regions of the surface, but it relies on marching cubes [Lorensen and Cline 1987] to re-sample from a signed-distance function at sites where topology changes occur. Unfortunately, because marching cubes cannot represent features smaller than a grid cell, this approach cannot perform topological changes on thin features without deleting large sections of the surface at a time.

The technique presented by Müller [2009], on the other hand, is ca-

pable of performing topological changes on some types of thin features, because it uses an extended set of marching cubes tables that is specifically designed to permit sheets thinner than the grid resolution. However, this strategy still cannot represent certain small features like thin columns or small droplets of liquid that do not intersect the edges of the marching cubes cells. Furthermore, this method re-samples the surface for all cells in every time step so it will not preserve small-scale surface details through time.

We propose a mesh-based surface tracking method for fluid animation that both preserves fine surface details and robustly adjusts the topology of the surface in the presence of arbitrarily thin features like sheets and strands. Instead of using lookup-table-based re-sampling methods to generate a surface from a topologically valid signed distance function, we introduce a local convex hull method for connecting surface features during topological changes. We sew this new surface together with the original surface mesh using a subdivision-based mesh-stitching algorithm, and then we create new vertices by subdividing these newly-created triangles until the surface is adequately sampled. Our contributions are as follows:

- Local convex-hull algorithm for topological operations**
 We combine information from both the input mesh and a volumetric signed distance function in order to reconstruct a physically-valid topology that is easily controllable.
- Detail preservation** We avoid re-sampling the surface everywhere except where topological changes occur. In those locations, both our local convex hull operation and a subdivision-based method for stitching together surfaces preserve as many Lagrangian surface details as possible. In places where we cannot avoid re-sampling the mesh, we apply an interpolating subdivision scheme to ensure a highly accurate surface.
- Thin Features** Our method naturally produces arbitrarily thin fluid features like sheets and strands while preserving high-resolution surface details.
- Constrained Topology** Our algorithm outputs a high resolution surface mesh whose topology is constrained to match that of a lower resolution fluid grid. This construction allows for the efficient simulation of highly detailed surface animations while preventing many potential artifacts caused by inconsistent mesh and grid topologies.



Figure 2: When combined with a mesh-based surface tension technique, our method produces realistic breakup behavior of thin liquid films. In this example, a ball of water smashes downward into a rectangular podium, rapidly spreads out into a chaotic sheet, and breaks up into droplets.

2 Related Work

In order to construct a surface from volumetric data, Lorensen and Cline [1987] proposed an isosurface extraction technique known as *marching cubes*, which uses a lookup table to generate piecewise linear surface geometry. Because the original marching cubes algorithm produced holes in the surface mesh, several researchers proposed strategies for making it topologically consistent [Nielson and Hamann 1991; Montani et al. 1994]. Another way to ensure that marching cubes templates have a consistent topology is to generate the lookup tables using a convex hull. Bhaniramka et al. [2000; 2004] showed that convex hulls can be used to reconstruct surfaces for volume data in any dimension, which is one of the reasons we found convex hulls so appealing for our algorithm. Kobbelt et al. [2001] improved the marching cubes method by allowing it to reconstruct sharp features, Ju et al. [2002] used dual contouring to construct a surface from hermite data, and Schaefer and Warren [2005] proposed a method for contouring the dual of the marching cubes data in order to reconstruct thin structures. Varadhan et al. [Varadhan et al. 2004] proposed a method for locally increasing the resolution of an implicit surface until it has the correct topology. Like our strategy for matching topologies between different surfaces, Varadhan et al. use the notion of a *topologically complex* grid cell.

Volumetric implicit surfaces are convenient when computing topological changes, though many researchers prefer to use a discrete surface representation. McNerney and Terzopoulos [2000] developed a grid-based method for imposing topological changes upon a mesh that is constrained to surface-offsetting motions. Lachaud et al. [2003] merge and split triangle meshes by explicitly sewing and cutting the mesh when nodes come close together, while Zaharescu et al. [2007] sew together the exact intersections of overlapping meshes. Pons and Boissonnat [2007] use a restricted Delaunay triangulation to sew meshes together, and Brochu and Bridson [2009] explicitly detect collisions and perform topological mesh surgery where possible. Wojtan et al. [2009] locally re-sample a triangle mesh from a volumetric grid in order to perform topological changes, and Müller [2009] globally re-samples the mesh from a grid using specially-constructed marching cubes tables.

Fluid simulation in computer graphics was pioneered by Kass and Miller [1990]. Stam [1999] later made the advection term unconditionally stable, while Foster and Fedkiw [2001] simulated water with a dynamic free surface. For the animation of liquid surfaces in graphics, an Eulerian technique for moving an implicit surface known as the level set method [Osher and Fedkiw 2002] has become standard. Enright et al. [2002] added Lagrangian particles to the level set in order to maintain additional surface details, and Bargteil et al. [2006] presented a semi-Lagrangian contouring strategy. Losasso et al. developed a method for increasing the fluid simulation resolution near the level set surface using an octree [2004] and coupled a level set with a particle-based fluid simulation [2008] in the effort to ameliorate volume loss where the fluid surface is under-resolved. To further decrease the memory footprint of high resolution level sets, hierarchical run-length encoding [Houston et al. 2006] can also be used. Several authors have simulated a dynamic implicit surface at a higher resolution than the simulated fluid [Goktekin et al. 2004; Bargteil et al. 2006; Kim et al. 2009], although we will explain in Section 4 that this strategy can lead to visual artifacts when the topology of the fluid differs from that of the surface.

3 Fluid Simulation with an Explicit Mesh

We use an Eulerian fluid simulation (see [Bridson 2008]) to advect a Lagrangian triangle mesh, and we perform local mesh operations

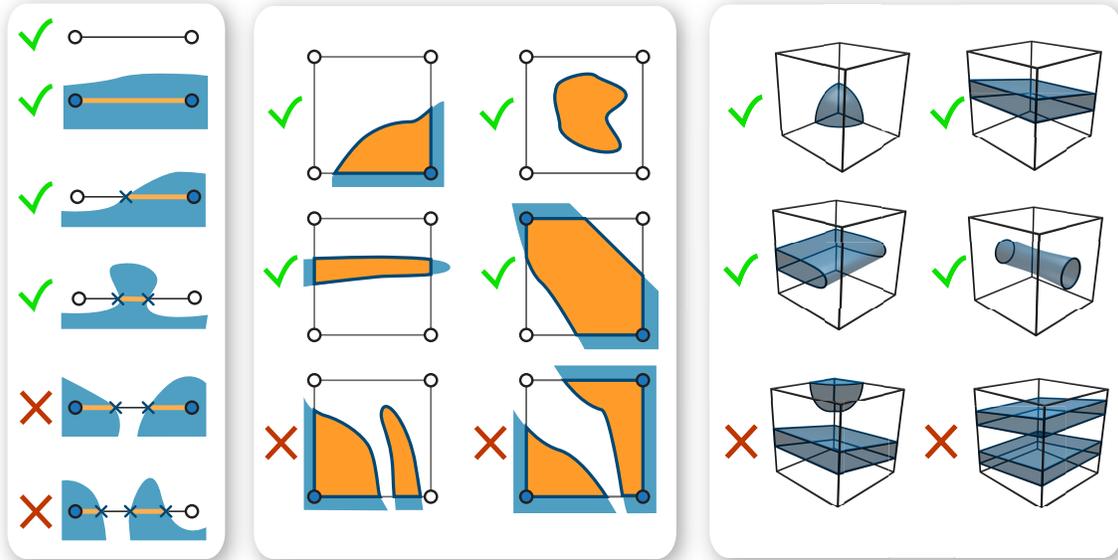


Figure 3: Examples of simple (green check mark) and complex (red X) topology for edges, faces, and cells.

like subdivision and edge collapses in order to ensure reasonably high quality triangle shapes. To inform the fluid simulation which cells should be simulated as liquid, we use the surface mesh to create a signed distance function with samples collocated with the pressure values in the fluid grid.

We employ a voxelization-style method [Müller 2009] to compute the signed distance function. We first voxelize the triangle mesh onto a grid by computing intersections with a rays in the x , y , and z directions. For each ray, we keep track of its inside/outside status with a counter. At the start of the ray (which is guaranteed to be outside of the mesh), the counter has a value of zero. For each intersection with the triangle mesh, we increment the counter if it intersects a triangle whose normal is facing the opposite direction of the ray, and we decrement it if it intersects a triangle with a normal facing the same direction of the ray. This way, all regions outside of the mesh will have a counter value of zero, regions inside the mesh will have a value of one, and inside-out regions will have values less than zero or greater than one. We store this counter value at each grid point to assign an inside/outside status, and then we compute the exact distance from the point to the surface in order to complete the signed distance calculation.

Every pressure value in the fluid grid that is collocated with an “inside” value is to be treated as an active fluid cell. Very thin features may not have any “inside” values nearby, so we additionally mark any cells that overlap the surface as active fluid cells. We then compute new velocities with the fluid simulation and use them to advect the surface mesh using an explicit Runge-Kutta technique. Finally, at the end of each time step, we detect and correct the topology of the surface. The remainder of this paper will explain our novel method for handling topological changes.

4 Topological Connectivity

The main goal of changing the topology of the mesh is to ensure that the liquid surface is connected to other regions of fluid in the same way as the pressure values. If these topologies differ, then distracting visual artifacts will occur. For example, if two disconnected surface components lie within the same cell in the fluid grid, then the fluid pressure values will be unable to distinguish between these independent components. The low resolution fluid velocities

will then move both surface pieces together in the same direction, creating an invisible link between them that tends to persist for the entirety of the simulation.

4.1 Detecting Topological Incompatibility

We detect disagreements between the explicit surface mesh and the fluid grid by locally contrasting the topology of the explicit surface with a surface that possesses the same topology as the fluid simulation. We define any region where these topologies disagree as *topologically complex*. As mentioned in Section 3, the fluid volume is defined by the inside/outside values of a signed distance function in addition to all cells that overlap the surface geometry. If the surface geometry contained no thin features, then we could simply compare the topology of the explicit surface to the topology of the isosurface of its signed distance function, as in Wojtan et al. [2009]. However, we wish to preserve thin features that cannot be resolved by the signed distance function, so we must use a more versatile definition of topological complexity.

Before specifying what it means to be topologically complex, we first define a topological cell as a cube with its eight corners located at sample points in the signed distance function. The cube is bounded by six faces, twelve edges, and eight corners. We analyze the topology of the explicit surface mesh by examining the intersection between the the solid geometries of the triangle mesh and each topological cell. The intersection is empty if the cell is completely outside of the surface, the intersection is identical to the original cube if the cell is completely inside the surface, and the intersection is more complicated if the cell overlaps the surface. In order for the cell to be topologically simple, the surface of this intersection should not have any more topological features than a single fluid cell. That is, it must contain at most one surface component, with no holes or voids — the surface of this intersection should be homeomorphic to a sphere (more formally, a 2-sphere). Similarly, the transitions from this topological cell to its neighbors must be topologically simple, so the surface intersections with faces and edges of this cell should be homeomorphic to 1-spheres and 0-spheres, respectively. Lastly, a corner of the cell is topologically simple if it is not located in an inside-out region of the surface. Figure 3 shows some examples of simple and complex geometry.

We can efficiently detect the topological complexity of a cell corner

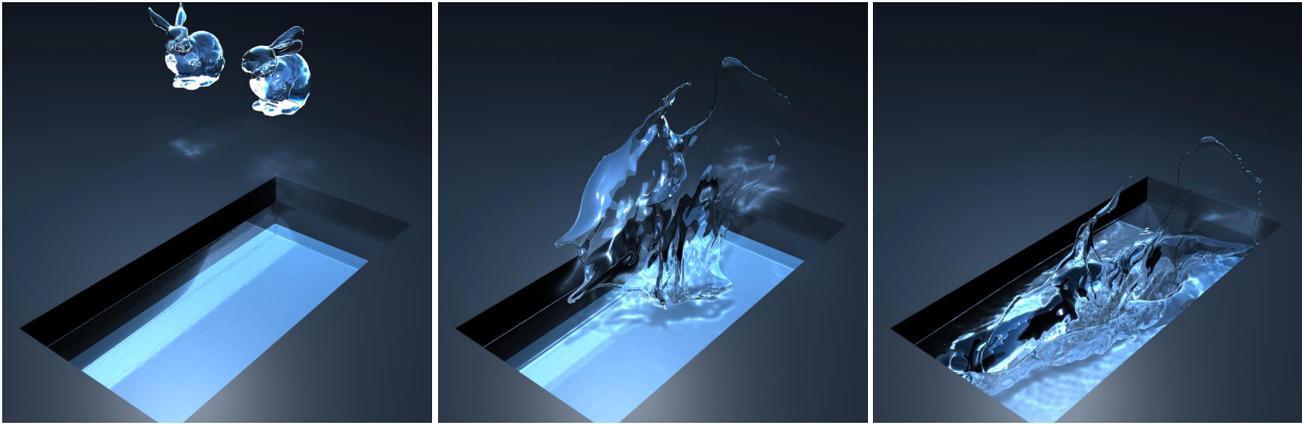


Figure 4: Two bunnies splatter into each other, producing a thin liquid sheet that merges with the pool of water below.

by checking its counter from the signed distance function calculation in Section 3 (the counter of a simple corner must be either zero or one, otherwise it means the surface is inside out or self-intersecting). Next, we determine the complexity of a cell edge by counting the number and orientation of its intersections with triangles in the surface mesh and comparing the result with a single line segment (0-sphere). If there are too many components, or if the surface is oriented the wrong way at any of the intersection points, then the edge is complex. This step is performed at the same time as the complex corner test (during the creation of the signed distance field). We can check the validity of a face by computing its intersection with the mesh and counting the number of components, and we can test the topological status of a cell by similarly counting connected components and ensuring that the Euler characteristic detects no holes.

The complex corner test samples the signed distance function at several regularly-spaced sample points, and it is guaranteed to identify any self-intersections larger than the grid spacing. This test will catch any topological flaws that are well-resolved in the x , y , and z dimensions, just like marching cubes will faithfully reproduce a surface as long as there are no features smaller than the grid size. The complex edge test checks all of the edges in the grid, so it is guaranteed to identify any topological flaws that are well-resolved in at least two dimensions. This means that self-intersections and pockets of air that look like thin sheets will be identified by the complex edge test. This test can also catch thin spindles and voids if they happen to intersect one of the grid edges. The complex face test checks for intersections with all faces in the grid, so it is guaranteed to catch topological flaws that look like thin spindles (which span more than one cell in a single dimension, but are very thin in the other two). However, the complex face test cannot catch topological flaws smaller than a single grid cell unless they happen to intersect the face. Finally, the complex cell test is guaranteed to identify all topological flaws, because it checks within every cell.

In practice, such topological problems smaller than a grid cell do not exist, because we start with a well-resolved surface and then

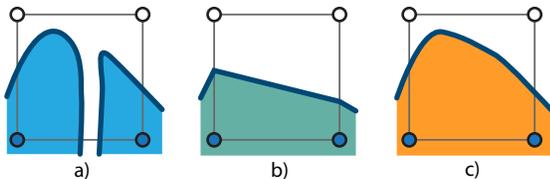


Figure 5: Given a surface with complex topology (a), methods based on marching cubes-style lookup tables will ignore interior vertices (b), while our method preserves many of the original details (c).

smoothly deform it according to a low resolution fluid velocity field. This means that large features morph into small features through a gradual process, by first becoming thin sheets and then thin spindles. Because topological inconsistencies are caught by lower-dimensional complexity tests before they have time to shrink smaller than a grid cell, we have not found it necessary to perform the full complex cell test. Complex face tests are mostly redundant for our purposes as well, because perturbations from the fluid velocity tend to force thin structures to intersect cell edges — the complex edge and complex corner tests tend to quickly catch all problems. As a result, we have found it practical to bypass the testing of any cells and faces unless we specifically have to guarantee valid topology in a particular region.

After a complex cell is detected, we will soon replace its intersecting surface with a similar surface that is topologically simple. Because this cell shares its boundaries with other cells that may not have been classified as topologically complex, we have to specifically guarantee that its faces are topologically simple. In this case, we count the face components and pass complexity information to neighboring cells, similar to the complex cell propagation strategy of Wojtan et al. [2009]. To summarize the frequency of topological tests in our implementation: we exhaustively test every corner and edge in the signed distance grid, we never check cells, and we only check faces when it is absolutely necessary to ensure simple connectivity between a complex cell and its topologically simple neighbors.

4.2 Local Topological Repair

After deciding that a cell has complex topology, we must replace the surface/cube intersection in that cell with a new one. We require that this new intersection surface meets two constraints: it should be topologically simple, and it should preserve the connectivity of the original surface. In addition, because each vertex of the original surface represents a valuable piece of Lagrangian simulation data, we also desire that the new surface preserves as many of the original surface vertices as possible.

According to our definition, a sphere is topologically simple. Consequently, a straightforward strategy for reducing the topological complexity of a surface/cube intersection is to wrap a single surface around all of the original surface components until it is homeomorphic to a sphere. Fortunately, this operation can be performed efficiently by replacing the original surface/cube intersection with its convex hull. If the input surface intersected the cell boundary, then the convex hull preserves this connectivity. Furthermore, the convex hull’s intersections with faces and edges are lower dimensional convex hulls, so they are also topologically simple. In addi-

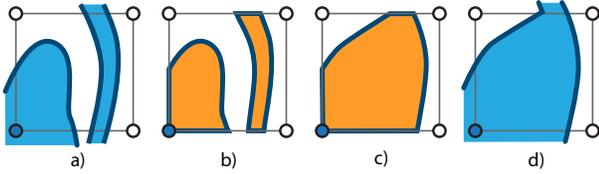


Figure 6: Given a triangle mesh and topological cell (a), we examine the intersection between the the solid geometries of the mesh and the cell (b). If the intersection has complex topology, we replace it with its convex hull (c). Finally, we remove facets belonging to the boundary of the cell and reconnect the surface (d). Note that the cell below this one will also be re-sampled, because the bottom edge in (a) is topologically complex.

tion, all of the vertices on this convex hull are preserved from the input surface, so we reduce re-sampling errors during this process, as illustrated in Figure 5.

In practice, we use the qhull library [Barber and Huhdanpaa 1995] to compute the convex hull of the following points: all original surface vertices that lie within that cell, the vertices created by intersecting the original mesh with the edges and faces of the cell (described in Section 5), and all “inside” corners of the cell. Finally, because the convex hull represents the intersection between a cube and the new surface, we extract the final surface by deleting any facets that are co-planar with the cell boundary. See Figure 6 for an example.

This strategy of reducing complex surface regions to topological spheres has physical implications when used in a fluid simulation: (1) thin sheets of air will be detected as topologically complex and destroyed, and (2) thin sheets and strands of liquid will persist throughout the simulation. Thus, we have already satisfied two out of the three physically-based topological rules laid out in Section 1, and we will explain how to allow the breakup of thin spindles in Section 4.3.

4.3 Topological Control

The method presented in Section 4.2 will preserve all thin sheets and spindles unless there are significant self-intersections. In case we desire different behaviors, we can modify the presented algorithm by altering our definition of topological complexity. For example, we could force the breakup of thin liquid spindles by declaring that a cell face is complex when the surface intersects the face but not its bounding edges. To create a re-sampled surface that respects this altered definition of topological complexity, we can simply remove any vertices on that face from the input to the convex hull algorithm. We can similarly remove many vertices from the convex hull input if we want to speed up computation time, although we did not find this optimization necessary.

An alternative to modifying the definition of topological complexity is to manually perform topological changes to the surface through explicit mesh surgery. We have found that this approach for cutting thin spindles of liquid produces better results than the method outlined in the preceding paragraph, because explicit cutting allows us to have tighter control over the thickness of the surface before the separation. We can detect thin spindles of liquid while performing the local surface maintenance operations (Section 3), when we flag edges that will produce non-manifold geometry if collapsed. Of these non-manifold cases, we can easily identify any thin spindles by their triangular cross-section. To perform the mesh surgery, we cut the mesh at the triangular cross-section, seal each end with a new triangle, and perturb the two new strands away from each other by a small offset. A similar operation is explained in detail by Lachaud et al. [2003].

As explained in Section 1, thin spindles of liquid consistently lead to topological changes in real-world liquids. Due to a surface-tension-based Rayleigh-Plateau instability, the liquid spindle rapidly collapses until it is infinitely thin and then breaks in two. Our explicit cutting operation precisely mimics this behavior in the discrete setting by separating the thinnest possible unit of a discrete surface. This cutting method is also quite efficient — it is detected for free during standard surface maintenance operations, and it requires about as much work as a single edge collapse.

Both of these proposed methods give us control over the breakup of thin liquid spindles, satisfying our final physically-based topology requirement from Section 1. Using these proposed topology modifications, the animator can specify either the topological grid length or the maximum allowed edge length in order to force the breakup of thin liquid spindles.

5 Sewing Meshes Together

Before re-sampling the surfaces in Section 4.2, we subdivide the original surface mesh so that it perfectly lines up with the boundaries of each complex cell. We first subdivide triangles where they intersect the boundary edge of a complex cell, and then we subdivide edges where they intersect the boundary faces of a complex cell, following the algorithm of Du et al. [2006]. Next, we discard the original surface within the complex cells and replace them with topologically simple convex hull surfaces from Section 4.2. Finally, we must stitch the surfaces together along the cell boundaries.

5.1 Subdivision Stitching

One possible mesh-stitching strategy is to progressively collapse triangles on both sides of the cell boundary until the detailed intersection between the surface and cell face is reduced to a line segment. However, such a strategy deletes important Lagrangian surface samples and can lead to robustness problems [Wojtan et al. 2009]. Instead of altering the original surface and destroying its Lagrangian data, we choose to only alter the new re-sampled surface.

Initially, if the new triangles output from the convex hull already perfectly match up with the triangles on the other side of the face (the piecewise linear curve formed by the triangle edges coincident with each boundary face is identical for both the original and new surfaces), then we are finished sewing the surfaces together at this face. Otherwise, we have to subdivide the new surface until its boundary curve matches that of the original surface. First, we limit the number of cases that we need to address by ensuring that each new triangle shares an edge with at most one triangle outside of the complex cell region — for each new triangle that shares edges with two or more triangles outside of the complex region, we place a vertex at its barycenter and split it into three new triangles. Next, for each new triangle that does not perfectly match up with a segment of the boundary curve, we subdivide it in two by adding a point on its boundary edge and snapping it to one of the vertices on the boundary curve. We then recursively subdivide each of these newly-created triangles in the same way until the curves perfectly match (each curve eventually reduces to a single line segment in the base case), similar to the algorithm used by Bischoff and

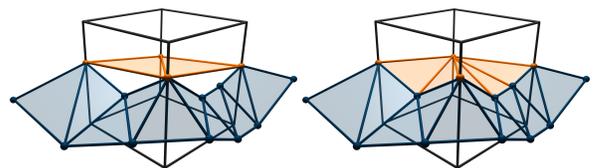


Figure 7: Before and after sewing the meshes together.



Figure 8: *Two streams of water collide, filling a domain with liquid.*

Kobbelt [2005] to stitch together NURBS surfaces. Our algorithm is guaranteed to terminate, and it is linear in the number of vertices coincident with the face. After this step, the newly-resampled mesh surface from Section 4.2 will be properly sewn together with the original surface along its boundary, creating a closed, oriented, manifold surface mesh. Please see Figure 7 for a visual aid.

5.2 Accurate Surface Interpolation

Both marching cubes-style lookup tables as well as our convex hull algorithm in Section 4.2 use piecewise linear surfaces to connect vertices together. These low-order connecting surfaces are indistinguishable from the original surface mesh when the triangles in the surface mesh are about the same size as a fluid grid cell. However, our method allows the surface mesh to have a much higher resolution than the simulation grid, so the triangles output by our convex hull will often seem disproportionately large. To maintain a detailed surface with many small triangles, we repeatedly subdivide these large triangles until their edges are as short as the maximum edge length allowed in our surface mesh. These subdivisions create new vertices, and we are free to place them wherever we like. Similar to Brochu and Bridson [2009] we decided to use butterfly subdivision when placing our new vertices. We found that the modified scheme of Zorin et al. [Zorin et al. 1996] worked best, because a significant portion of the vertices in our simulations did not have a valence of six. After applying this interpolating subdivision to the triangles output by our topological correction algorithm, the surfaces exhibit both correct topology and high degrees of smoothness.

6 Results and Discussion

Our simulations all show different combinations of thin sheets, strands, and droplets due to a high resolution surface mesh being advected through a lower resolution fluid simulation. Figure 8 shows two streams of water colliding to fill up a domain. Most other simulation techniques would be unable to resolve such a continuous stream of thin sheets, but ours easily avoids the deletion of the large thin structures. The simulation took between 0.5 and 10 seconds per frame, and it finished with 800k triangles in a $100 \times 50 \times 100$ fluid domain. Figure 9 shows a comparison between our method and a level set surface tracker on a 60^3 domain. Our method clearly resolves more surface details and thinner sheets, although our method is considerably more expensive due to the mesh maintenance. This example took about 8 seconds per frame and finished with 200k surface triangles. Figure 4 shows two bunny meshes creating a thin liquid sheet after violently colliding into each other. The thin sheet then merges with a larger body of water below. This example av-

eraged 28 seconds per frame for a $60 \times 120 \times 120$ fluid simulation with at most 520k triangles. Figure 2 shows how our method for persistent liquid sheets combines with a mesh-based surface tension technique [Thürey et al. 2010] in order to simulate the realistic breakup of a thin liquid film. The domain is $160 \times 80 \times 160$ (although most of it is empty), and it took 1 to 12 seconds per frame to simulate 320k triangles. Finally, Figure 1 shows an example of a hallway flooding. This simulation was able to simulate thin sheets and droplets despite its coarse fluid simulation resolution. It simulated at 2 to 13 seconds per frame and finished with 600k triangles in a $120 \times 60 \times 60$ fluid domain. Our simulations typically output one frame of animation for each time step in the simulation.

Because we use fairly low resolution fluid simulations, and because the thin sheets in our videos do not occupy the full volume of the fluid simulation, we found that the performance of our method is more closely tied to the resolution of the surface mesh than to the fluid grid. The majority of the computation time in our examples was spent on mesh-based operations (subdivision and edge collapses, advection of the surface vertices, topology changes, and mesh-based surface tension calculations), although the expense of the surface tracker becomes negligible with higher resolution volumetric fluid simulations and coarser surface meshes. The fraction of total simulation time spent in the topological change algorithm varied from 17% to 57% for the examples in this paper (Figure 2 and Figure 8, respectively). The signed distance calculation is by far the most expensive part of the topology change algorithm, followed by a global recomputation of the triangle mesh connectivity data structure after any mesh surgery. We believe that we can significantly speed up our implementation in the future by using a hardware voxelization routine and only locally updating mesh connectivity.

While grid-based approaches have one dominant parameter (grid resolution), our method has two: topological resolution (grid cell size) and surface detail resolution (triangle edge length). These independent resolution parameters allow our method to track hundreds of surface vertices within a single grid cell while maintaining the same topology as the underlying fluid simulation. The cost of maintaining extra vertices in a single grid cell is similar to that of a particle level set [Enright et al. 2002], though a particle level set with 64 particles per cell will resolve significantly fewer surface details than our method with 64 surface vertices per cell.

The cost of computing topological changes in our method is modest, though its expense grows linearly with surface detail resolution. For example, if our surface detail resolution matches the topological change resolution, then our method will behave similar to that of Müller [2009] (near real-time performance), with significantly reduced diffusion of surface details. As the surface resolution increases, so does the cost of advecting the mesh and computing topological changes. The memory consumption of our method scales linearly with the surface resolution, in order to store the surface mesh and the grid cells that intersect it.

Figure 10 illustrates a failure mode of common surface trackers with a low fluid resolution. In this scenario, grid-based implicit

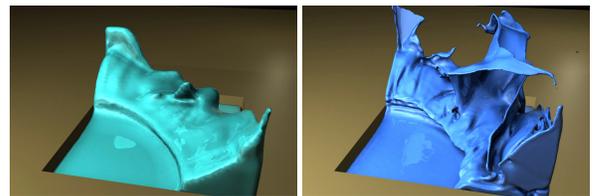


Figure 9: *Comparison between a level set surface tracker with 2nd order advection (left) and our method (right) on a 60^3 grid.*

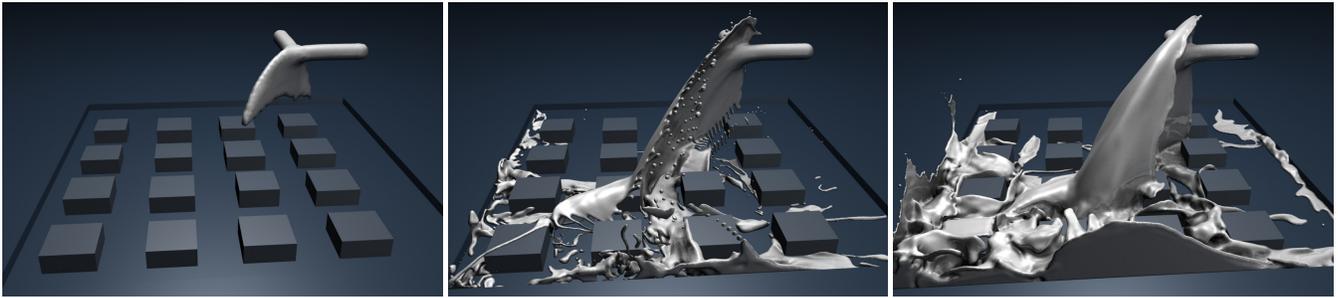


Figure 10: In this example, the level set method (left) deletes thin sheets before they even touch the ground. The method of Wojtan et al. 2009 (middle) re-samples the surface from a grid during topological changes, so it cannot merge together multiple thin sheets without deleting large portions of the surface. Our method (right) merges together thin sheets without rampant thin feature deletion.

surface trackers (level sets, particle level sets, and semi-Lagrangian contouring, for example) limit the surface detail to that of the grid resolution, which causes catastrophic deletion of thin features. Naively increasing the surface resolution can cause the topology of the fluid simulation to differ from that of the surface tracker. Such mismatched topologies can lead to artifacts such as several surface components floating within a single fluid cell, as well as an unphysical breakup of thin sheets. Artificially diffusing the level set [Kim et al. 2009] can force these floating components to merge together, at the expense of limiting surface detail, losing volume, and punching holes in thin sheets. Our method is the only surface tracker to date that can showcase arbitrarily high levels of surface detail without sacrificing topological agreement between the surface and the fluid.

Although we did not use exact arithmetic to compute the topological details of our algorithm, we did take precautions to make it robust in the face of numerical precision errors. In the event that a numerical error causes impossible topology or leads to difficulty sewing together the mesh, we mark all cells near the error as topologically complex and re-sample the geometry within them.

We tried to preserve as much Lagrangian surface information as possible throughout the surface tracking process, and our topological constraints allow us to advect surface meshes that are much higher resolution than the fluid velocity field. Consequently, we can animate highly detailed surfaces through time without losing surface features to a re-sampling process. However, this retention of high frequency details may not always be desirable. Some possible ways to smooth out extra details are with geometric smoothing or simulated surface tension. In addition, our topological corrections will often replace high-frequency surface details with a less detailed surface whenever they detect complex geometry. This replacement can lead to noticeable temporal discontinuities when simulating with an extremely low resolution fluid simulation grid.

Because our topological algorithm primarily merges fluid regions together and rarely splits things apart, the liquid in our simulations tends to exhibit an overall gain in volume, especially after long, violently splashing simulations with many topological merges. This is the opposite of the common volume-loss problem with level set-style surface trackers, and the amount of volume gain is closely tied to the coarseness of the fluid grid.

Nevertheless, we intentionally used low-resolution fluid simulations in many of our examples to show the advantage of separating the visible surface resolution from the resolution of the velocity field. The hallway in Figure 1, for example, is only 30 grid cells wide. Despite the coarse physics, our method is able to develop many high resolution surface details while still matching the topology of the low resolution fluid simulation. The additional detail gained from separating the fluid simulation from its surface allows

us to avoid simulating expensive high resolution physics when only a highly detailed surface is necessary.

7 Conclusion and Future Work

We have presented a mesh-based surface tracking method for fluid animation that both preserves fine surface details and robustly adjusts the topology of the surface in the presence of thin features like sheets and strands. Our local convex-hull-based method for correcting the topology of surfaces enforces the overall topological behavior of real-world liquids, and we can achieve arbitrarily thin sheets and strands without re-sampling important Lagrangian surface details. We further reduce re-sampling artifacts with a subdivision-based mesh-stitching algorithm, and we use a higher order interpolating subdivision scheme to determine the location of any newly-created vertices. We have shown how our method can be applied to efficiently produce high-quality fluid surface animations, even when coupled to under-resolved fluid simulations.

In the future, we would like to look into mesh-based methods for creating sub-grid scale physical behaviors. In addition to using a mesh-based surface tension method [Thürey et al. 2010], we believe that other surface-focused approximations to forces in the Navier-Stokes equations like the vortex sheet method of Kim et al. [2009] could yield highly detailed simulations without the expense of running a full fluid simulation. Lastly, one potentially fruitful area of future work is to investigate the possibility of topologically separating fluid surfaces instead of merging them together, as done for elastic models by Teran et al. [2005] and Nesme et al. [2009].

Acknowledgements

We would like to thank Tobias Pfaff for editing our supplemental video and Thomas Oskam for creating some of the figures in this paper. The first author was funded through the NSF grants CCF-0811485 and CCF-0625264.

References

- BARBER, C., AND HUHDANPAA, H., 1995. Qhull, Softwarepackage.
- BARGTEIL, A., GOKTEKIN, T., O'BRIEN, J., AND STRAIN, J. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics (TOG)* 25, 1, 38.
- BHANIRAMKA, P., WENGER, R., AND CRAWFIS, R. 2000. Iso-surfacing in higher dimensions. In *Proceedings of the conference on Visualization'00*, IEEE Computer Society Press Los Alamitos, CA, USA, 267–273.

- BHANIRAMKA, P., WENGER, R., CRAWFIS, R., ET AL. 2004. Isosurface construction in any dimension using convex hulls. *IEEE Transactions on Visualization and Computer Graphics* 10, 2, 130–141.
- BISCHOFF, S., AND KOBBELT, L. 2005. Structure preserving CAD model repair. *Computer Graphics Forum* 24, 3, 527–536.
- BRIDSON, R. 2008. *Fluid simulation for computer graphics*. AK Peters Ltd.
- BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing* 31, 4, 2472–2493.
- DE GENNES, P., BROCHARD-WYART, F., QUÉRÉ, D., AND REISINGER, A. 2004. *Capillarity and wetting phenomena: drops, bubbles, pearls, waves*. Springer Verlag.
- DE LUCA, L., AND COSTA, M. 2009. Instability of a spatially developing liquid sheet. *Journal of Fluid Mechanics* 331, 127–144.
- DU, J., FIX, B., GLIMM, J., JIA, X., LI, X., LI, Y., AND WU, L. 2006. A simple package for front tracking. *Journal of Computational Physics* 213, 2, 613–628.
- ENRIGHT, D. P., MARSCHNER, S. R., AND FEDKIW, R. P. 2002. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH 2002*, vol. 21, 736–744.
- FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *the Proceedings of ACM SIGGRAPH 2001*, 23–30.
- GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)* 23, 3, 463–468.
- HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics (TOG)* 25, 1, 175.
- IVANOV, I. 1988. *Thin liquid films: fundamentals and applications*. CRC.
- JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. In *ACM SIGGRAPH 2002 papers*, ACM New York, NY, USA, 339–346.
- KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *the Proceedings of ACM SIGGRAPH 90*, 49–57.
- KIM, D., SONG, O.-Y., AND KO, H.-S. 2009. Stretching and wiggling liquids. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, ACM, New York, NY, USA, 1–7.
- KOBBELT, L., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H. 2001. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, 57–66.
- LACHAUD, J., TATON, B., AND EN INFORMATIQUE, L. 2003. Deformable model with adaptive mesh and automated topology changes. In *Proc. 4th Int. Conference on 3D Digital Imaging and Modeling, Banff, Canada, IEEE*, 12–19.
- LORENSEN, W., AND CLINE, H. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH 1987 papers*, ACM, 169.
- LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (TOG)* 23, 3, 457–462.
- LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 797–804.
- MCINERNEY, T., AND TERZOPOULOS, D. 2000. T-snakes: Topology adaptive snakes. *Medical Image Analysis* 4, 2, 73–91.
- MONTANI, C., SCATENI, R., AND SCOPIGNO, R. 1994. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer* 10, 6, 353–355.
- MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 237–245.
- NESME, M., KRY, P., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM SIGGRAPH 2009 papers*, ACM, 52.
- NIELSON, G., AND HAMANN, B. 1991. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization '91*, IEEE Computer Society Press Los Alamitos, CA, USA, 83–91.
- OSHER, S., AND FEDKIW, R. 2002. *Level set methods and dynamic implicit surfaces*. Springer Verlag.
- PONS, J.-P., AND BOISSONNAT, J.-D. 2007. Delaunay deformable models: Topology-adaptive meshes based on the restricted Delaunay triangulation. *Proceedings of CVPR '07*, 1–8.
- RAYLEIGH, L. 1883. Investigation of the character of the equilibrium of an incompressible heavy fluid of variable density. *Proc. Lond. Math. Soc* 14, 1, 170–177.
- SCHAEFER, S., AND WARREN, J. 2005. Dual Marching Cubes: primal contouring of dual grids. In *Computer Graphics Forum*, vol. 24, Citeseer, 195–201.
- STAM, J. 1999. Stable fluids. In *the Proceedings of ACM SIGGRAPH 99*, 121–128.
- TERAN, J., SIFAKIS, E., BLEMKER, S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 317–328.
- THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. In *ACM SIGGRAPH 2010 papers*, ACM.
- VARADHAN, G., KRISHNAN, S., SRIRAM, T., AND MANOCHA, D. 2004. Topology preserving surface extraction using adaptive subdivision. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, 244.
- WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers*, ACM, 76.
- ZAHARESCU, A., BOYER, E., AND HORAUD, R. 2007. Transforms: a topology-adaptive mesh-based approach to surface evolution. *Lecture Notes in Computer Science* 4844, 166.
- ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating subdivision for meshes with arbitrary topology. In *ACM SIGGRAPH 1996 papers*, ACM, 192.