Surface-Only Simulation of Fluids

Fang Da

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Surface-Only Simulation of Fluids

Fang Da

Surface-only simulation methods for fluid are those that perform computation only on a surface representation, without relying on any volumetric discretization. Such methods have superior asymptotic complexity in time and memory than the traditional volumetric discretization approaches, and thus are more tractable for simulation of complex fluid phenomena. Although for most computer graphics applications and many engineering applications, the interior flow inside the fluid phases is typically not of interest, the vast majority of existing numerical techniques still rely on discretization of the volumetric domain. My research first tackles the mesh-based surface tracking problem in the multimaterial setting, and then proposes surface-only simulation solutions for two scenarios: the soap-films and bubbles, and the general 3D liquids. Throughout these simulation approaches, all computation takes place on the surface, and volumetric discretization is entirely eliminated.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my most sincere gratitude towards my advisor, Professor Eitan Grinspun. You saw the passion and drive in me when I was merely a young enthusiast with nothing to prove myself with, and took the risk of bringing me on to your research group. You went out of your way to guide and help me in many aspects of my graduate life, far more than just research. Most importantly, your devotion to the beauty of math and science and your relentless pursuit of perfection revolutionized my view of scientific research and beyond.

I would like to thank Professor Christopher Batty, Professor Matei Ciocarlie, Professor Chris Wojtan and Professor Changxi Zheng for serving on my committee. Special thanks to Christopher and Chris, with whom I had a long and most joyful collaboration. You showed me the meaning of patience, diligence, resourcefulness, astuteness and creativity. I'm proud to say I learned much from you after these years.

I'd like to also thank my collaborators at MIT, Professor Pedro Reis and Khalid Jawed, my mentors at Adobe Research, Miklós Bergou, Sunil Hadap and Byungmoon Kim, and my colleagues at Columbia, Breannan Smith, Etienne Vouga, Akash Garg, Danny Kaufman, Keenan Crane, Yotam Gingold, Papoj Thamjaroenporn, Dingzeyu Li, Timothy Sun, Yun Fei, Henrique Maia, Alec Jacobson, Yonghao Yue, Gabriel Cirio and Anne Fleming. My PhD years have been full of fun, because of every one of you.

I would like to express special thanks to my family. You have always believed in me even when I didn't, and you have always supported me even when my decisions cost you in many ways. You are the reason for what I have achieved so far, and you will always be the reason for whatever I will achieve in the future.

For my wife, Nan

Chapter 1

Introduction

Simulation of complex fluid phenomena has been an area of great interest in computer science research. The demand to construct large and complicated splash scenes frequently arises from the movie effects industry nowadays (such as the 4,000-feet wave in the feature film *Interstellar*), and physically-based simulation tools are playing an irreplaceable role in creating the rich details of computer generated imagery, which is prohibitively expensive if animated manually. The need for understanding and predicting the mechanical behavior of fluids also occurs commonly in various engineering fields such as hydraulic engineering. Due to the near zero resistance to shearing deformation, fluid materials often exhibit large and complex motions that are not easy to model analytically, making numerical simulation techniques indispensable tools for studying the fluid behavior in scenarios where direct physical experiments are not feasible.

These applications demonstrate the strength of computer over man in numerical computation tasks. However, as powerful as modern computers are in performing numerical computation, these simulations are still not fast enough when high quality details are desired. Seeking new techniques to improve the computation efficiency has been a major theme in computer graphics research. This can be tackled from many angles such as developing more powerful hardware, optimizing the computation code, and improving the solution algorithm. The last approach affords the greatest potential by exploiting domain-specific knowledge and insight for fluid behavior. It is the focus of this dissertation.

Pioneered by a few seminal papers towards the end of the twentieth century, fluid simulation research has gained significant traction in the computer graphics community over the past two decades,

and has heralded innovative techniques for each component of the simulation pipeline. The majority of these simulation approaches take on the task of solving the Navier-Stokes equations head on, discretizing the independent variable (the 3D velocity field) directly. Depending on the discretization techniques (grids, tetrahedral meshes or particles), different approaches show strength in different scenarios, but they all use geometric primitives to capture the concept of an infinitesimal volumetric element, and thus have a DOF (degree of freedom) count proportional to the fluid volume, and inversely proportional to the geometric resolution. This set of DOF is typically solved from a sparse Poisson equation which captures the key property of common fluid materials, the incompressibility. Therefore, most existing simulation approaches have the same asymptotic complexity with respect to the result's resolution.

Compared to the volume's cubic scaling with respect to the resolution, the surface (interface between different phases) area scales at a more favorable quadratic rate. In other words, *if the surface is the only desired output, then it may be possible to design algorithms with potentially better asymptotic complexity.* It is this philosophy that drove the development of new simulation techniques in this dissertation.

1.1 Surface-Only Simulation

Throughout this dissertation I refer to a class of simulation techniques termed *Surface-only simulation* techniques. In many fluid phenomena, there is a clear *surface* (or *interface*) separating different fluid phases of distinct physical properties that do not mix or diffuse into each other. Examples of such surfaces include the interface in a water-oil mixture, the soap film surface in a soap foam, or simply the water surface in a pool of water. Very often each fluid phase in these multiple phase scenarios can be assumed to be homogeneous and the interface turns out to be where the most interesting interaction happens. From a computer graphics viewpoint, the interface is the only visually important element of the final imagery, as it is where the most prominent optical effects such as reflection and refraction occur. For many engineering applications, the interface is also the only aspect of the outcome that is of interest in the engineering process under consideration (such as inkjet printing [Derby, 2010], spray combustion [Faeth, 1977], and liquid-liquid extraction [Park and Blair, 1975]). Therefore, it is natural to focus the computation power available at resolving the interface motion as accurately as

possible. The motion inside the interior of the homogeneous phases, although potentially influential on the evolution of the interface, is typically not sought after for its own sake.

There has been some exploration of surface-only simulation techniques, such as vortex filaments [Weissmann and Pinkall, 2010], vortex sheets [Brochu *et al.*, 2012; Pfaff *et al.*, 2012], and potential flow [Keeler and Bridson, 2014], among others, with distinct benefits: since there is no need to allocate storage or computation resources for the fluid interior or exterior, greater efficiency becomes possible. With the help of fast summation techniques such as the Fast Multipole Method [Greengard and Rokhlin, 1987] or Particle-Particle Particle-Mesh method [Zhang and Bridson, 2014], these techniques can generally achieve a time complexity linear in the surface size [Brochu *et al.*, 2012; Zhu *et al.*, 2015], which is asymptotically optimal.

1.2 Challenges facing surface-only methods

Before any dynamics can be formulated, the first instrument we need for a surface-only simulation framework is a surface tracking method. The most popular existing surface tracking techniques are based on level sets [Sethian, 1999; Osher and Fedkiw, 2002], which inevitably relies on volumetric discretization, usually in the form of a spatial grid. The alternatives are mesh-based surface tracking techniques, which are known for preservation of volume and sharp details [Wojtan *et al.*, 2011], but have long been plagued by challenges in robustly handling topology changes (such as splitting and merging). Not long before the work of this dissertation, several methods have been proposed to deal with this issue [Campen and Kobbelt, 2010; Müller, 2009; Wojtan *et al.*, 2009; Brochu and Bridson, 2009], but they are all limited to two-phase scenarios. The general case where multiple fluid phases swirl next to each other requires a more flexible representation and a more careful handling of topology changes, due to the exponentially more complex topological configurations possible at the vertices where more than two phases meet.

More challenges await in the formulation of dynamics: first and foremost, a surface-only simulation technique must be able to encode into a surface representation all the necessary information needed for the time evolution. For different fluid phenomena, different representations are suitable, and the dynamics model must be formulated accordingly. In general, we cannot expect a surface representation to be expressive enough to capture every possible 3D velocity field, and thus certain

assumptions that restrict the potential velocities must be adopted. Fortunately, velocity fields in many fluid phenomena relevant to computer graphics can be very well approximated by vector fields that are harmonic (both incompressible and irrotational) on the interior, which is restrictive enough to allow a surface representation to fully describe the volumetric velocity field. In this dissertation I consider two particular scenarios: the first is the simulation of air flow around soap films (as in bubbles and foams), and the second is the simulation of general 3D inviscid liquids. The reason why the harmonic velocity field assumption is justified will be discussed in detail for each scenario.

1.3 Contributions

My research has made novel contributions in the following directions:

1. Improved mesh-based surface tracking technique in the multimaterial setting. In Part I of this dissertation, I present *Los Topos*, the first non-manifold triangle mesh tracking method to simultaneously maintain intersection-free meshes and support the broad set of multimaterial remeshing and topological operations, including mesh splitting, merging, T1 and T2 transitions. Los Topos is developed from improving the El Topo library [Brochu and Bridson, 2009] with a more versatile and robust handling of complex topology and a less disruptive mesh-merging operation. The robust T1 transition for arbitrary configuration around any given vertex is achieved by introducing the concept of *region graph* which is a concise representation of the vertex local topology, thus converting the topology classification problem into the completeness of the region graph. No matter how complex the vertex local topology is, the proposed resolution algorithm guarantees to resolve the irregular configurations in finitely many steps. This work was published and presented at SIGGRAPH 2014 [Da *et al.*, 2014b].

2. The first surface-only technique for simulating the air flow around soap films. Previous techniques for simulating soap bubbles and foams either rely on volumetric discretization (typically an Eulerian grid), or model the film by itself while ignoring the dynamics of the air. Typical thickness of soap films is so thin that the air mass significantly outweighs the liquid mass, rendering the latter physically inaccurate, while also making the former computationally inefficient due to the high resolutions required to resolve the film in the transversal direction. In Part II I present a surface-only method based on vortex sheets for simulating the air flow around

soap films under surface tension force. By judiciously designing the surface representation, I formulated the surface tension force into one simple time integration rule that is extremely easy to implement and efficient to evaluate. This work was published and presented at SIGGRAPH 2015 [Da *et al.*, 2015].

3. The first surface-only technique for simulating general 3D inviscid liquids. Although there have been attempts to simulate liquid motion with surface-only techniques [Zhang *et al.*, 2012; Keeler and Bridson, 2014], these attempts are all limited to the specific phenomena they are originally designed for, and none is flexible enough to be applied to general 3D liquid motion such as water jets and splashes. This task has been traditionally accomplished by volumetric techniques such as SPH or a Eulerian grid based method. In Part III of this dissertation, I present a surface-only simulation framework for inviscid, uniform-density liquid undergoing complex motion in 3D. To accomplish this goal, I propose a novel incompressibility projection step for surface velocity fields based on Helmholtz decomposition, and a linear solve employing the Boundary Elements Method for integrating external forces such as surface tension, gravity and solid contact. This work was published and presented at SIGGRAPH 2016 [Da *et al.*, 2016].

Part I

Multimaterial Mesh-Based Surface Tracking

Chapter 2

Introduction

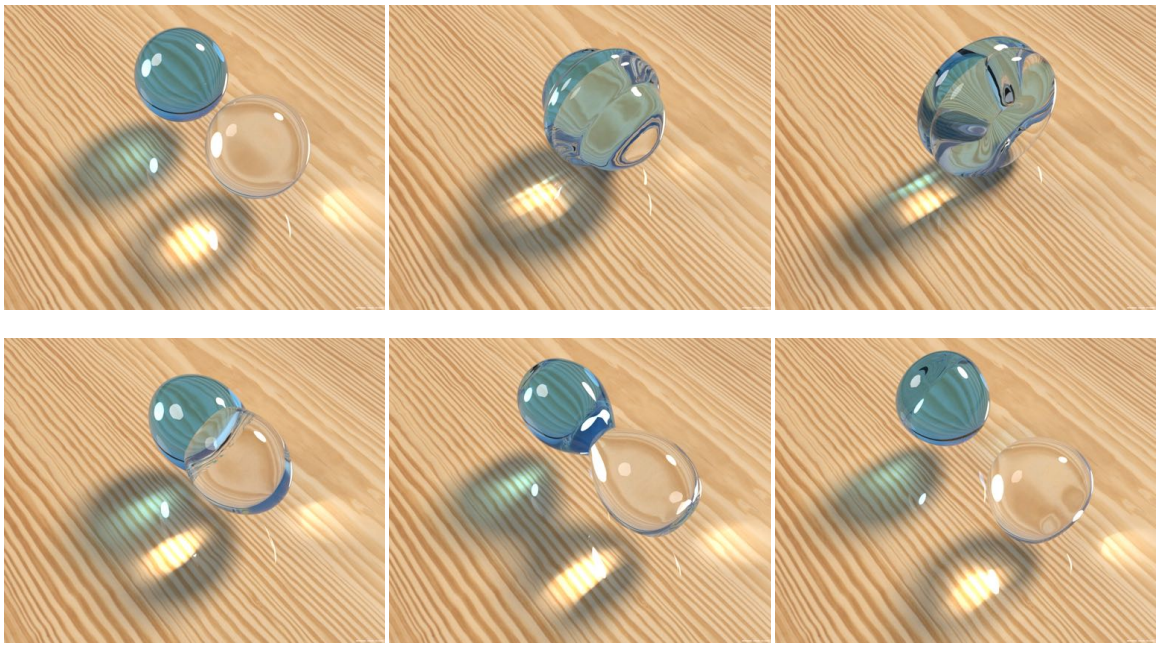


Figure 2.1: **Two-Droplet Collision:** Using our multimaterial mesh-based surface tracker, two immiscible liquid droplets with different materials but identical physical properties impact symmetrically in zero gravity under strong surface tension. The collision merges the droplets so that a new interface separates the two liquids, and a non-manifold triple-curve is created where the two liquids meet the ambient air.

One of the most important building blocks of a surface-only simulation framework is a versatile and robust surface tracking technique 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 surface tracking* techniques do so by iteratively advancing a triangle mesh. They do not rely on any volumetric discretization such as grids, and have been touted for their ability to preserve volume, mesh-scale detail, and sharp features [Wojtan *et al.*, 2011]. 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 nonmanifold 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.

Chapter 3

Related Work

3.1 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. Reconstructing quality multimaterial surface or volume meshes from this data is also of interest [Wu and Sullivan, 2003; Meyer *et al.*, 2008; Bronson *et al.*, 2012]. 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]. Bronson et al. [2012] demonstrated an application of their Lattice-Cleaving tetrahedral mesher to multiphase flows.

3.2 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].

3.3 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 (Chapter 5.4). 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 (Chapter 6) 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.

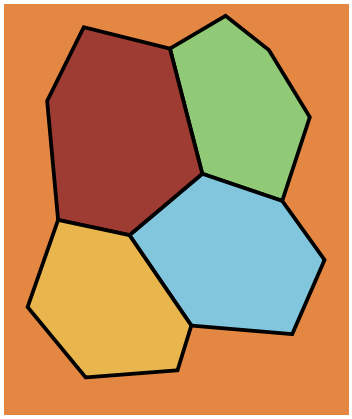
3.4 Surface Remeshing

Remeshing of surface meshes is a very broad topic, with many applications and goals. Alliez et al. [2008] provide an overview of recent developments; we will briefly review work that is relevant to our setting. We focus on incremental local remeshing, since it allows each remeshing operation to be robustly checked for collision-safety [Brochu and Bridson, 2009]. Various authors have applied this style of remeshing for surfaces undergoing gradual but large deformations (or *mesh adaptation*), and high quality results have been achieved by considering anisotropy, curvature, and feature-detection [Wicke *et al.*, 2010; Jiao *et al.*, 2010; de Goes *et al.*, 2008; Clark *et al.*, 2012; Narain *et al.*, 2012; Clausen *et al.*, 2013]. We employ uniform resolution isotropic meshes with one particular choice of feature detection; combining our method with more advanced remeshing strategies is a key future direction. Several authors have also studied remeshing, subdivision, and fairing targeted at non-manifold surfaces [Hubeli and Gross, 2000; Ying and Zorin, 2001; Zilske *et al.*, 2008;

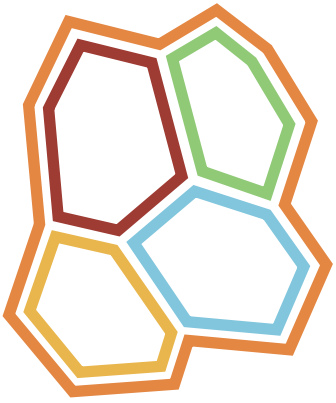
Pellerin *et al.*, 2011].

Chapter 4

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.

Chapter 5

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 (§5.1)	
Resolve interpenetrations (§5.2)	✓
Perform topological merging (Chapter 8)	✓
Perform mesh improvement (§5.3)	✓
Perform topological separation (Chapter 6 and 7)	✓

end while

5.1 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.

5.2 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.

5.3 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 are in Chapter 9.

5.4 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 5.1).

Multimaterial Merging, discussed in Chapter 8, 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 5.1, top). This contrasts with the usual two-material case, where colliding regions always have the same material, and merge into one.

T1 and T2 Processes are known from the literature on soap films and bubbles [Weaire and

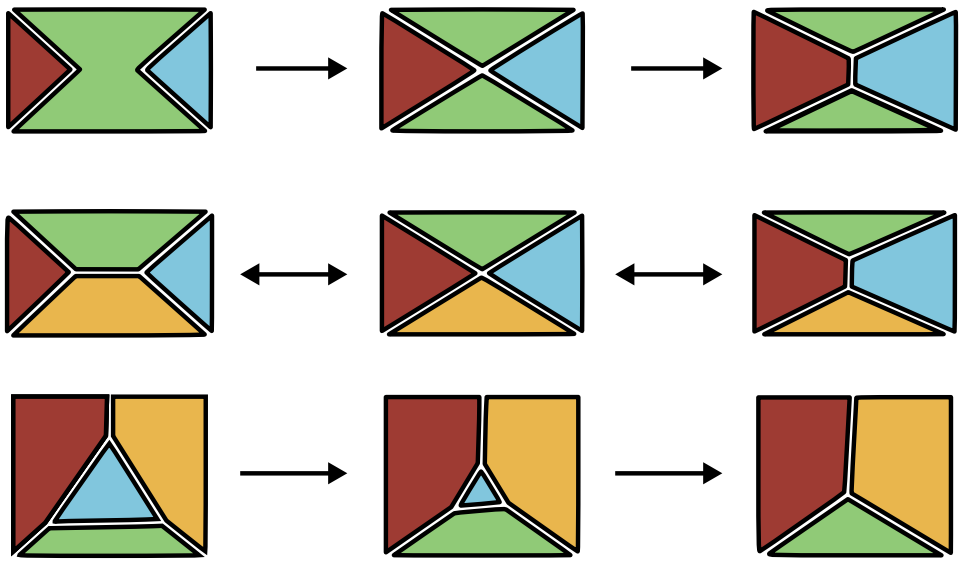


Figure 5.1: **Multimaterial topology changes in 2D.** Merging (top): Two material regions separated by a third collide to yield a shared interface. *T1 process* (middle): Two regions that share a border separate while the remaining pair become connected. *T2 process* (bottom): One region (cyan) collapses away.

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 5.1, 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 5.1, bottom).

For a complete set of operations to support surface tracking, mesh splitting would be necessary as well. However, in our framework, this is achieved automatically through the T1 process (generalizing the concept above, if we have the same material in the red and blue regions and another material in the yellow and green regions in Figure 5.1, top, the T1 process becomes what we intuitively understand as the *splitting* of the originally vertically connected region into two).

Chapter 6

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 6.1). 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 [Taylor, 1976] in 2D prohibit stable equilibria for soap films with valences above three [Weaire and Hutzler, 2001].

However, as illustrated by Figure 5.1, 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; typically, we will analyze the local

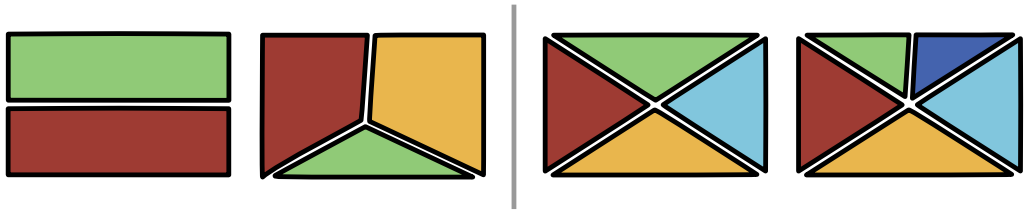


Figure 6.1: **Regular and irregular vertex configurations in 2D:** Left: An interface separating two regions is regular, as is a triple-point vertex separating three regions. Right: Vertices with edge valence of four or higher are irregular.

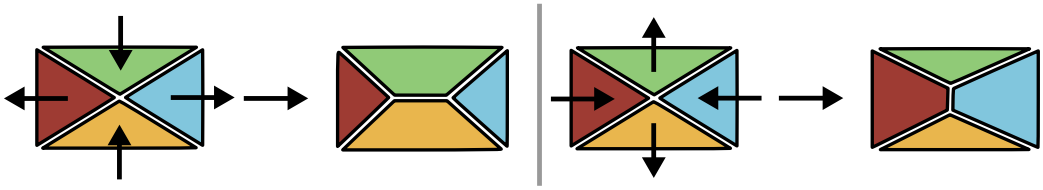


Figure 6.2: **Choice of separation direction:** Resolution of an irregular vertex depends on the underlying flow field.

velocity field (Figure 6.2).

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 6.3), and the (ir)regularity of a vertex cannot be defined simply in terms of valence, since many different distinct irregular topologies are now possible (see Figure 6.4 for a sneak peek). 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 a concrete definition of regularity that allows classification of vertex topology based on just its local neighborhood information, and 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.

Figure 6.3: 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.

Region Graphs Now consider the space of possible configurations about a vertex in 3D. As in 2D, the regular configurations (Figure 6.4, 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 6.4, 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 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 6.5 shows the region graphs for the regular and irregular configurations in Figures 6.4. (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*

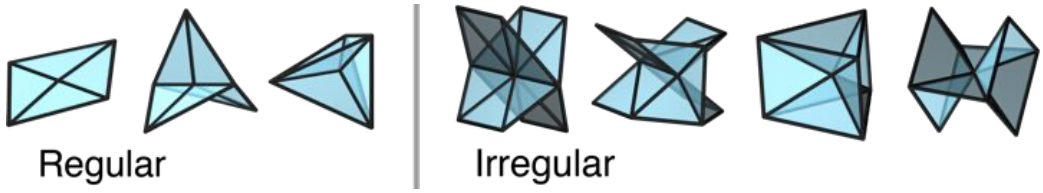


Figure 6.4: **Regular and irregular configurations in 3D:** Left: (1) A manifold surface separating two regions (here, the top and bottom half-spaces), (2) a triple-curve bordering three regions, and (3) a quadruple-point at which four regions meet. Right: Four of the infinitely many possible irregular non-manifold configurations that must be resolved by vertex separation.

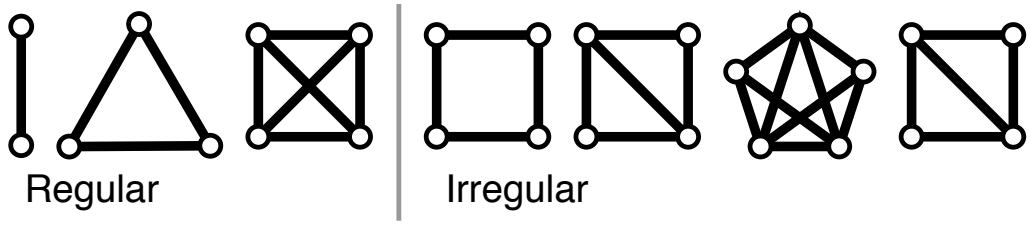


Figure 6.5: **Region graphs:** Left: The region graphs for vertices at: (1) a two-region surface, (2) a 3-region curve, and (3) a 4-region junction, corresponding to Figure 6.4, left. Their graphs are *complete*. Right: The region graphs for the central vertices in the irregular configurations of Figure 6.4, right, are *incomplete*.

connected (Figure 6.5, left); all other graphs, which lack some arcs, correspond to irregular vertex configurations (Figure 6.5, 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 6.6 and 6.7, 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 6.7, 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 $\mathsf{v_a}$ and $\mathsf{v_b}$. 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 $\mathsf{v_a}$ rather than v, by simply relabeling one of its vertex indices. Similarly, triangles incident to region $\mathbf{B}$ are updated to use $\mathsf{v_b}$. All remaining triangles

incident on v, but not on region $\mathbf{A}$ or $\mathbf{B}$, are updated to use v_b. Figure 6.7, 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 6.7, right).

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 6.7 where the top and bottom middle triangles are absent.) 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 6.7, 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

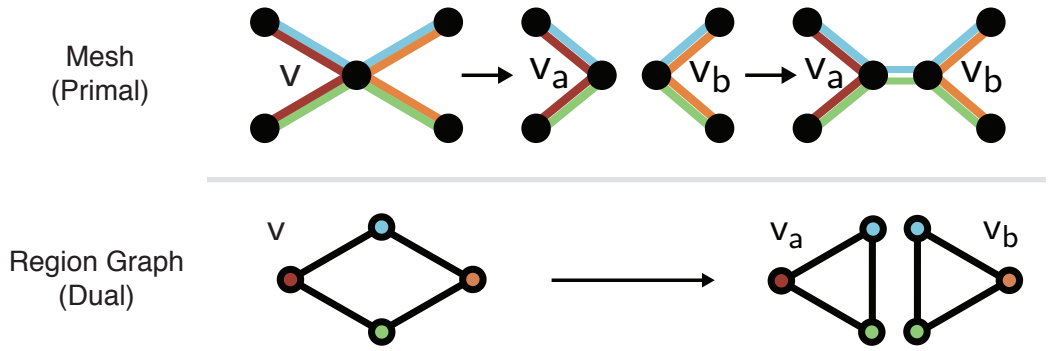


Figure 6.6: **Vertex separation in 2D:** Top: An irregular vertex v in 2D (left) is separated into two regular vertices, v_a and v_b (middle); the resulting gap is filled by a new edge with appropriate labels (right). Bottom: The incomplete region graph of v is correspondingly converted into two complete region graphs for v_a and v_b.

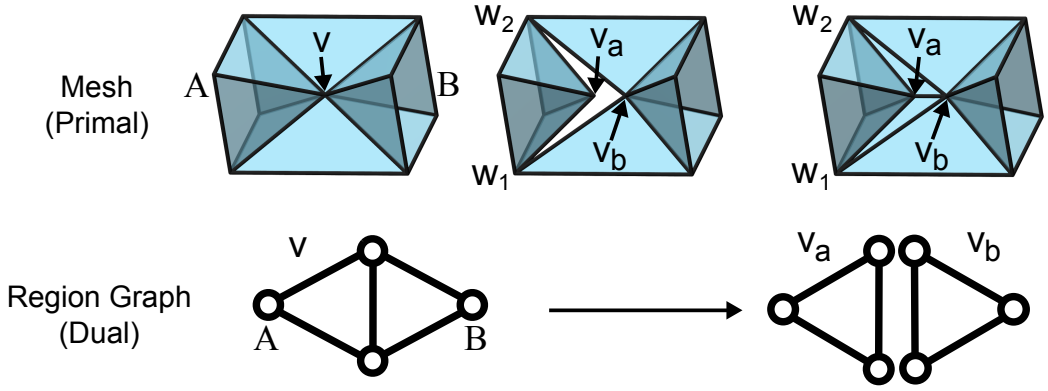


Figure 6.7: **Vertex separation in 3D:** Top: An irregular vertex in 3D (left) is duplicated and separated (center), and the resulting gap is filled by new triangles with appropriate labels (right). Bottom: The incomplete region graph is replaced by two complete region graphs.

to be processed, assured that iteration will terminate within finite steps with a set of regular vertex configurations. Figures 6.8 and 6.9 show a vertex initially incident on six regions that requires three vertex separations, and the corresponding region graph sequence, respectively.

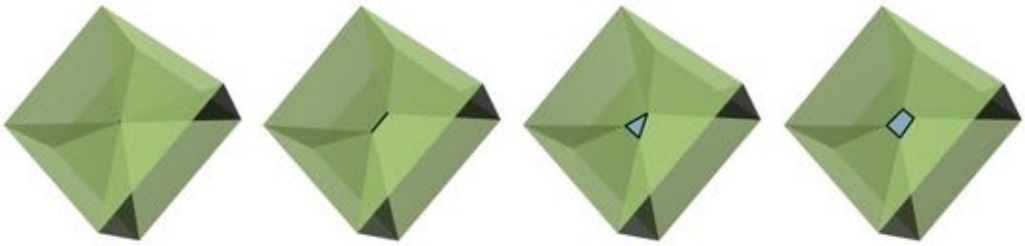


Figure 6.8: **Complex vertex resolution:** This geometry's central irregular vertex is incident on six regions, and has a region graph with multiple disconnected node pairs. A sequence of three vertex separations are needed to resolve this case. The result is a new interface (blue) between the front and back regions, bounded by four regular vertices. Region graphs are shown in Figure 6.9.

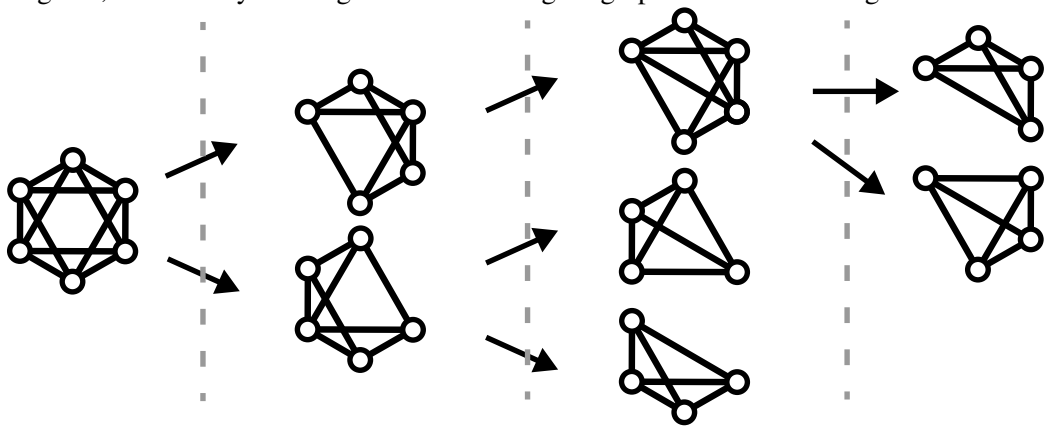


Figure 6.9: **Complex vertex region graphs:** The sequence of region graphs corresponding to Figure 6.8. The original irregular vertex is split into two irregular vertices, each incident on five regions. Each of these vertices are split into two regular vertices, each incident on four regions. While 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.

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.

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 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.

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.

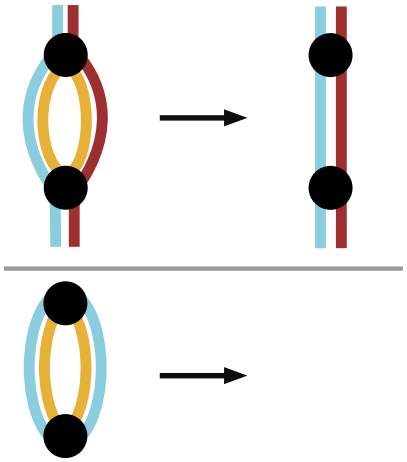
Algorithm 2 Vertex Separation

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 $\mathsf{v_a}$ and $\mathsf{v_b}$
    $\mathsf{v_a} \leftarrow \mathsf{v} + \epsilon\mathbf{d}_{AB}$
    $\mathsf{v_b} \leftarrow \mathsf{v} - \epsilon\mathbf{d}_{AB}$
    for all face f incident to v **do**
      if f is incident to region $\mathbf{A}$ **then**
        Change f's vertex v to $\mathsf{v_a}$, keeping its labels
      else
        Change f's vertex v to $\mathsf{v_b}$, keeping its labels
      end if
    end for
    for all vertex w adjacent to v and incident to $\mathbf{A}$ **do**
      Add face $\mathsf{v_a v_b w}$ with proper labels if needed
    end for
  end for
end while

Chapter 7

T2 Processes

We now consider the simpler T2 process in which a region shrinks until it disappears, as occurs in convergent velocity fields (Figure 7.1). 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]).

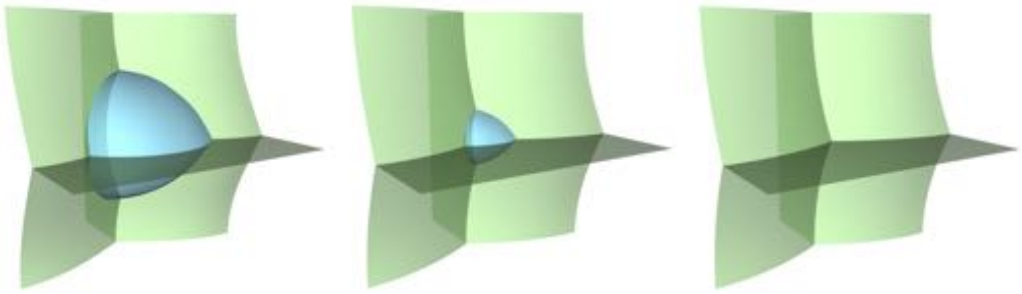
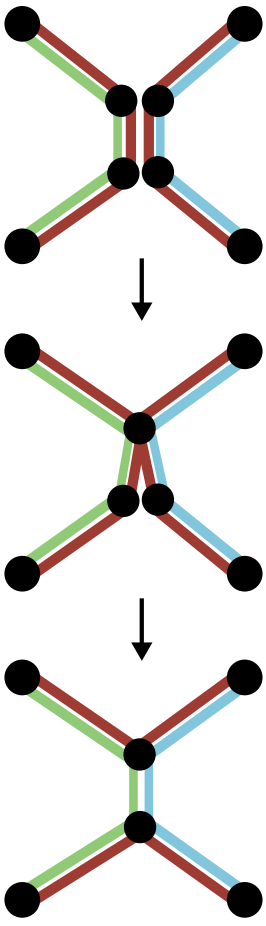


Figure 7.1: In a **3D T2 process**, one region (blue) collapses away.

Chapter 8

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, 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]).



8.1 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 (Chapter7) 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 8.1 shows the analogous 2D process.

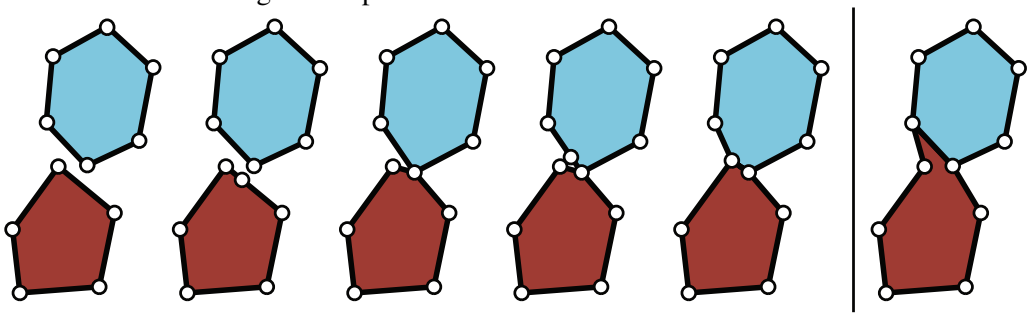


Figure 8.1: **Generalized snapping in 2D:** Left: For nearby meshes, an edge in a proximate edge-vertex pair is split at the closest point, and the resulting vertex pair is snapped. Similarly snapping a second edge-vertex pair completes the merge. For matching materials, the separating edge would then 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. 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.

Chapter 9

Multimaterial Mesh Improvement

Our multimaterial mesh-based surface tracking algorithm employs collision-safe, local remeshing [Brochu and Bridson, 2009] to maintain the quality of the triangle mesh, while preserving intersection-safety. This is achieved via a set of local multimaterial remeshing operations which are direct extensions of classic mesh operations, including edge flipping, edge splitting, edge collapsing, and vertex smoothing. For completeness, we describe the details of these operations and their multimaterial modifications below.

9.1 Our approach

At each iteration, we perform passes of edge splitting, edge collapsing, edge flipping, and vertex smoothing. We cancel any operations that would introduce collisions, as well as those that would yield unacceptably poor quality angles, inverted normals (relative to the original local patch), tiny-area triangles, and large volume loss; following Brochu and Bridson [2009] we expose the target angle bounds, edge length bounds, minimum triangle area bound, and volume change bound as user parameters, as these may be problem-dependent (see also Chapter 9.6). These bounds are respected by each of our topological operations as well. (In particular, the maximum volume change bound should be chosen with this in mind; an overly strict limit can hinder merging.)

We detect feature edges by thresholding dihedral angles between triangle pairs sharing an edge [Botsch and Kobbelt, 2004; Dunyach *et al.*, 2013], rather than analyzing the local quadric metric tensor [Brochu and Bridson, 2009; Jiao *et al.*, 2010]; we found this to be more robust and intuitive to

control, and it extends naturally to the non-manifold case. We used a threshold of 30°. Vertices lying on one or two feature edges are considered to belong to a feature curve or *ridge*, and vertices lying on three or more feature edges are identified as *peaks*.

9.2 Edge Flipping

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

Rather than flipping based on a Delaunay(-like) criterion, we follow Botsch and collaborators in seeking a mesh with more regular connectivity [Botsch and Kobbelt, 2004; Dunyach *et al.*, 2013]. That is, we perform flips that drive valences towards six for interior manifold vertices and four for boundary vertices. To handle manifold patches that border on non-manifold edges and vertices, we simply ignore triangles that are connected to the patch only through a non-manifold edge for the purposes of valence-counting, so that non-manifold vertices are seen as boundary vertices (i.e., with an ideal valence of four, within the manifold patch in question). We perform the flip if it reduces the total least squares difference between the ideal valences and the current valences.

$$\min \sum_{i=1}^{4} \left(\deg(\mathsf{v_i}) - \deg(\mathsf{v_i})^{opt} \right)^2 \tag{9.1}$$

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

9.3 Edge Splitting and Collapsing

Splitting and collapsing of non-manifold edges are straightforward extensions of the manifold case (Figure 9.1), and can be checked for collisions in the same way [Brochu and Bridson, 2009]. We

use an upper and lower edge-length bound to trigger splitting and collapsing, respectively. Child triangles created by an edge split inherit the labels of their parents. Edge collapses do not require relabeling, since the two triangles bordering the edge are deleted and the surrounding labels remain correct.

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

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

The collapsing operation is a potential source of certain degeneracy configurations. Most of the degeneracy cases are resolved by the same flap-removal operation introduced in 7; the most common cases are the collapse of an edge of a tetrahedral region, which effectively constitutes a T2 process, and the collapse of an edge of a triangle with the opposite vertex being of valence 3, which brings the two triangles neighboring the two uncollapsed edges to coincidence and thus deleting them.

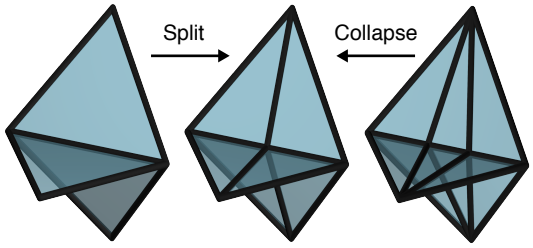


Figure 9.1: **Non-manifold edge splits and collapses** closely parallel their manifold counterparts.

9.4 Vertex Smoothing

We apply smoothing of vertex positions to improve the shape of mesh triangles. Because naïve Laplacian smoothing tends to rapidly destroy volume, particularly in high curvature regions, we follow previous authors in applying *tangential* or *null-space* smoothing, which removes the normal component of the displacement induced by smoothing [Botsch and Kobbelt, 2004; Jiao and Alexander, 2005; Jiao *et al.*, 2010]. For feature ridge vertices, smoothing-induced displacement perpendicular to the ridge direction is projected out instead so that smoothing occurs only along the ridge. For peak vertices, no smoothing is applied. We compute the vertex normal and ridge direction per Jiao and Bayyana [2008]. No special treatment is required for non-manifold geometry.

For strongly folded geometry (dihedral angles exceeding 165°) we instead perform Laplacian vertex smoothing in the average plane of the fold, as a special case. That is, we find the average normal of the incident triangles, and project out smoothing perpendicular to that direction. The goal is to widen the angle at the fold, under the assumption that such very sharp folds correspond to a crease that is in the process of merging. Treating this case with smoothing rather than relying entirely on collision-induced merging produces smoother intersection curves, for example during the initial collisions of spheres undergoing normal flow (Figure 10.3).

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

9.5 Eliminating Very Poor Triangles

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

9.6 Simulation Parameters

Our algorithm exposes a set of parameters that control the behavior of various mesh operations, allowing the method to cope with the diverse needs of different applications. Some of these parameters are drawn directly from the original *El Topo* method [Brochu and Bridson, 2009], while a few are specific to our new multimaterial setting.

For most of the simulations presented in the paper, we used the following default set of parameter values. Maximum volume change allowed in any remeshing operation is set to $0.1\%\bar{L}^3$ where $\bar{L}$ is the mean of the upper and lower target bounds for edge lengths during remeshing. The minimum area triangle allowed to be created during remeshing is set to $2\%\bar{L}^2$. Vertex separation distance is set to $10\%\bar{L}$. The range of triangle internal angles allowed to be created by any remeshing operation is set to $[3^\circ, 177^\circ]$. In addition to edge length bounds, we also attempt to split edges whose opposing angle exceeds 160°. The distance threshold that triggers proximity-based impulses during collision handling ranged from $10^{-4}\bar{L}$ to $10^{-3}\bar{L}$. The distance threshold at which merging is initiated ranged from $10^{-4}\bar{L}$ to $2 \times 10^{-2}\bar{L}$. The merge distance threshold should typically be set larger than the collision proximity threshold, since otherwise the collision code will act to keep surfaces too far apart to merge, which would clearly be counterproductive.

While these parameter settings were generally effective there are clear tradeoffs involved, which is why we chose to expose them to the user. For example, in situations where smaller perturbations at

T1 transitions are desired, the user can specify a smaller separation distance for the *vertex separation* operation used to resolve irregular vertices. This reduces the size of the required instantaneous mesh modifications at the expense of reduced mesh regularity (i.e., a shorter new edge). Similarly, a low threshold on the allowable volume change during remeshing operations can better preserve volume, but requires canceling some operations that may have lead to better mesh quality.

Chapter 10

Evaluation

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.

Below, we evaluate the capabilities of our method on several complex multimaterial mesh flows. 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.*, 2014a].

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/`.

10.1 Prescribed Velocity Flows

We start with multimaterial variants of two traditional stress tests: the Zalesk sphere and Enright tests [Enright *et al.*, 2002a]. 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

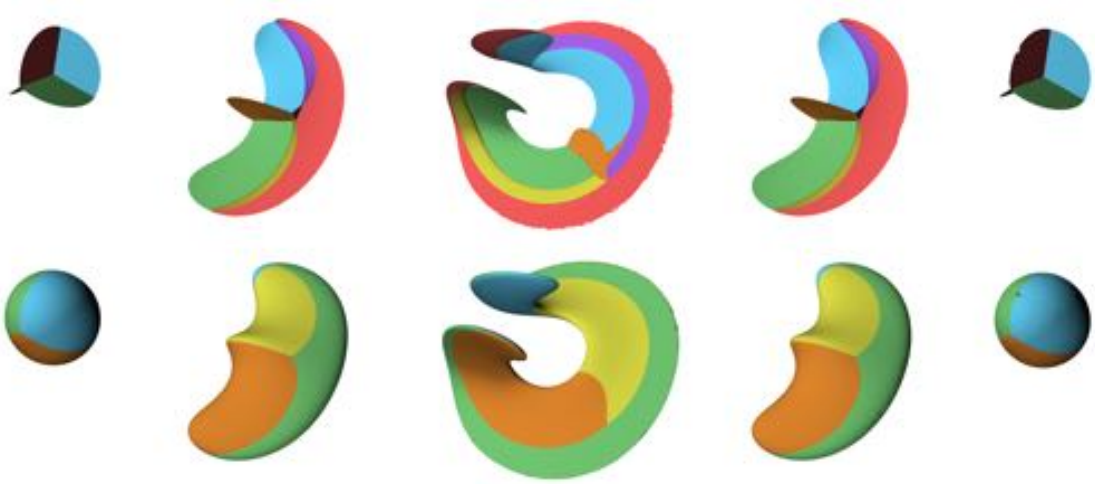


Figure 10.1: **Multimaterial Enright test:** A sphere divided into four material regions is advected through the "Enright test" velocity field. The distinct regions are well-preserved despite the extreme stretching. Top: Interior interfaces. Bottom: Exterior interfaces.

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. The final configuration shows remarkably faithful recovery of the original spherical shape; the largest error is seen around the region undergoing extreme stretching, where small errors introduced by the remeshing operations such as edge collapsing and T1 vertex separation are magnified significantly into the final configuration errors.

Our multimaterial Zalesak disk test exhibits no perceptible smoothing or lost volume. Our multimaterial Enright test (Figure 10.1) 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, 2013a]. 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 to four spheres of different materials, which resulted in the complex geometry in Figure 10.2. 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 example we consider 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.

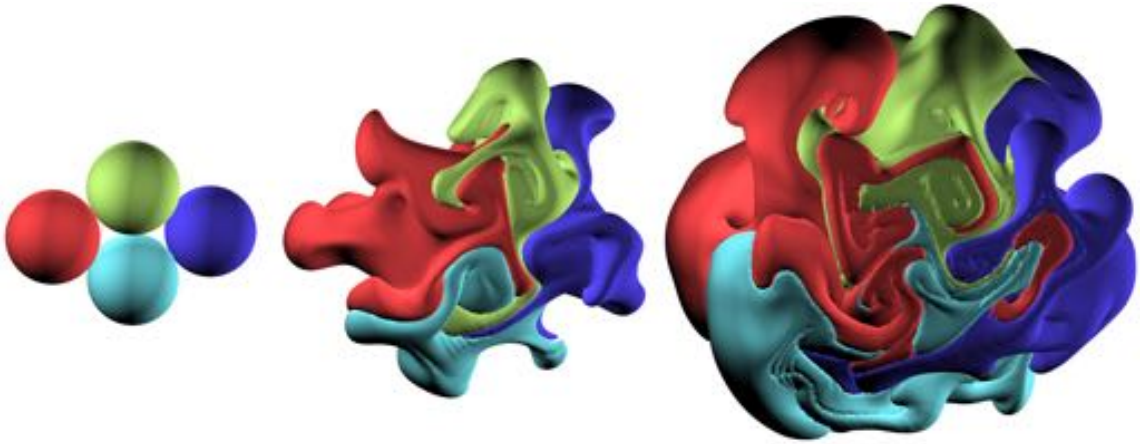


Figure 10.2: **Normal Flow + Curl Noise:** Alternating steps of normal flow and curl noise applied to four spheres yield complex deformations and topology changes.

10.2 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. Results show 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

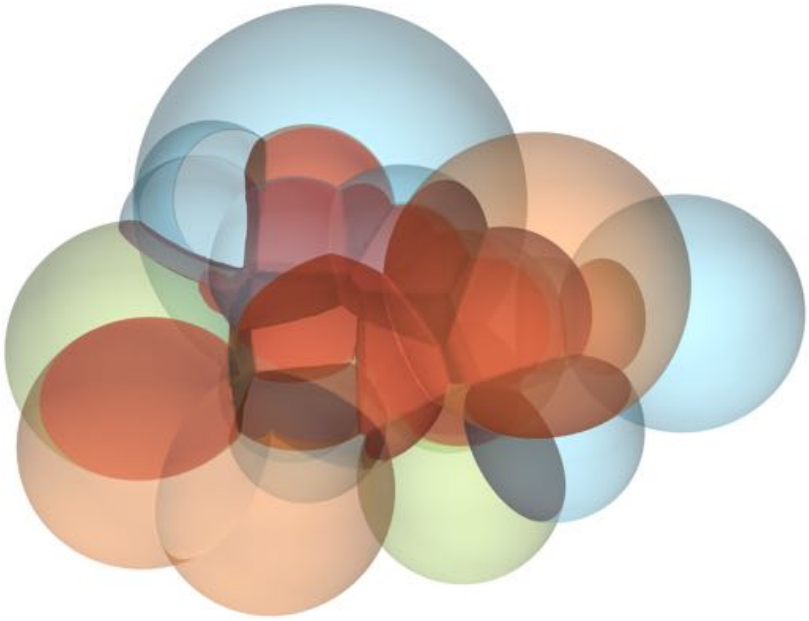


Figure 10.3: **Multimaterial normal flow:** Twenty-five spheres of different materials and sizes undergo outward normal flow collide to yield many internal interfaces, shown in red.

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 material merging produces interior interfaces, while matching materials merge into one region. Reversing the direction of flow causes the interfaces to smoothly shrink and disappear.

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 10.4).

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*

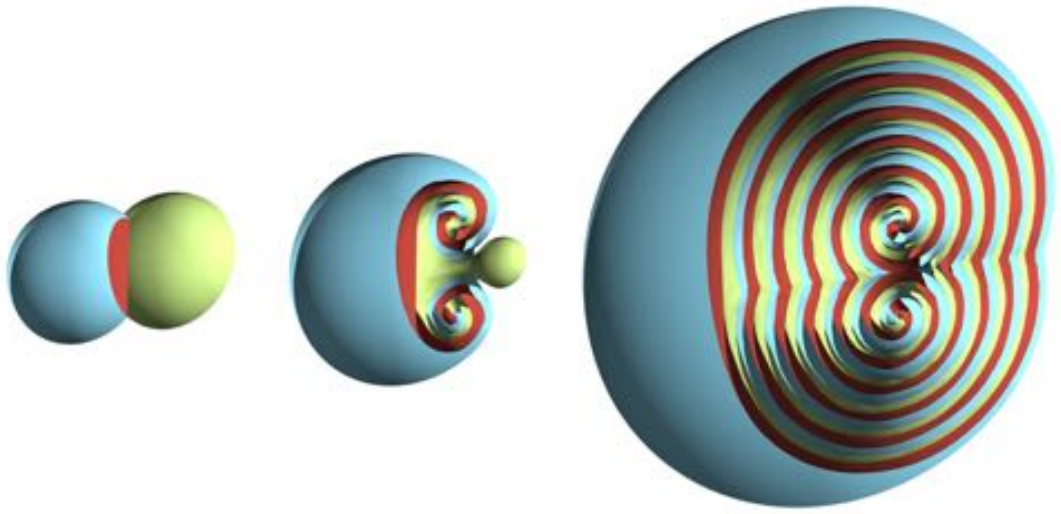


Figure 10.4: **Cyclical normal flow:** Two overlapping spheres sharing a circular internal interface undergo normal flow with a cyclical ordering, with the front cut away for visualization. The result is a curling effect around the stationary triple curve.

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 6.3 and 7.1 show basic T1 and T2 processes, driven by mean curvature flow. An arrangement of 2000 distinct materials evolving inside a cube (Figure 10.5) demonstrates robustness across thousands of topology changes.

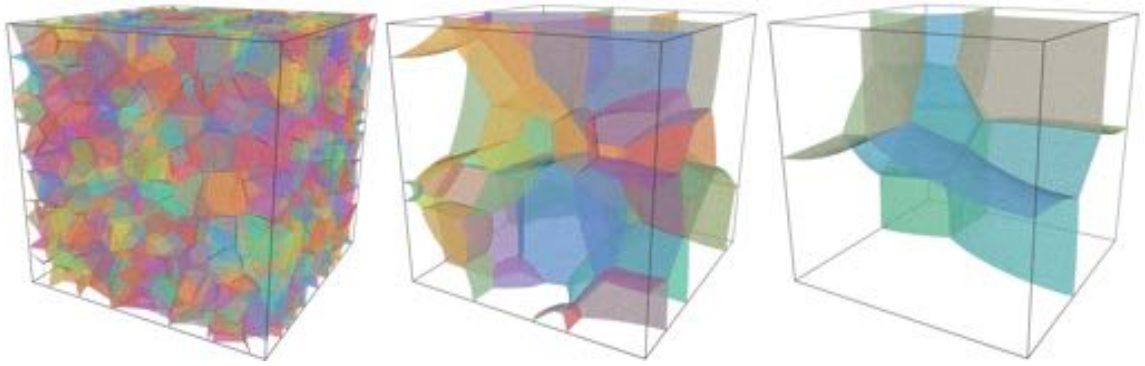
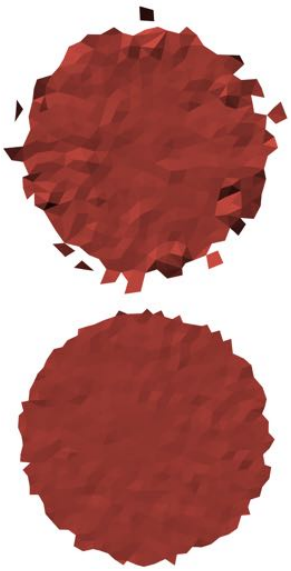


Figure 10.5: **Mean curvature flow:** A collection of 2000 random Voronoi cells undergoes many T1 and T2 topology changes.

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.



10.3 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 outer geometry also exhibits greater overall symmetry and smoothness with snapping.

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 Chapter 10.2, but does not adequately stress test complex merging.

10.4 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 20.1 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. We also consider 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 10.6) 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, 2013a], 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.

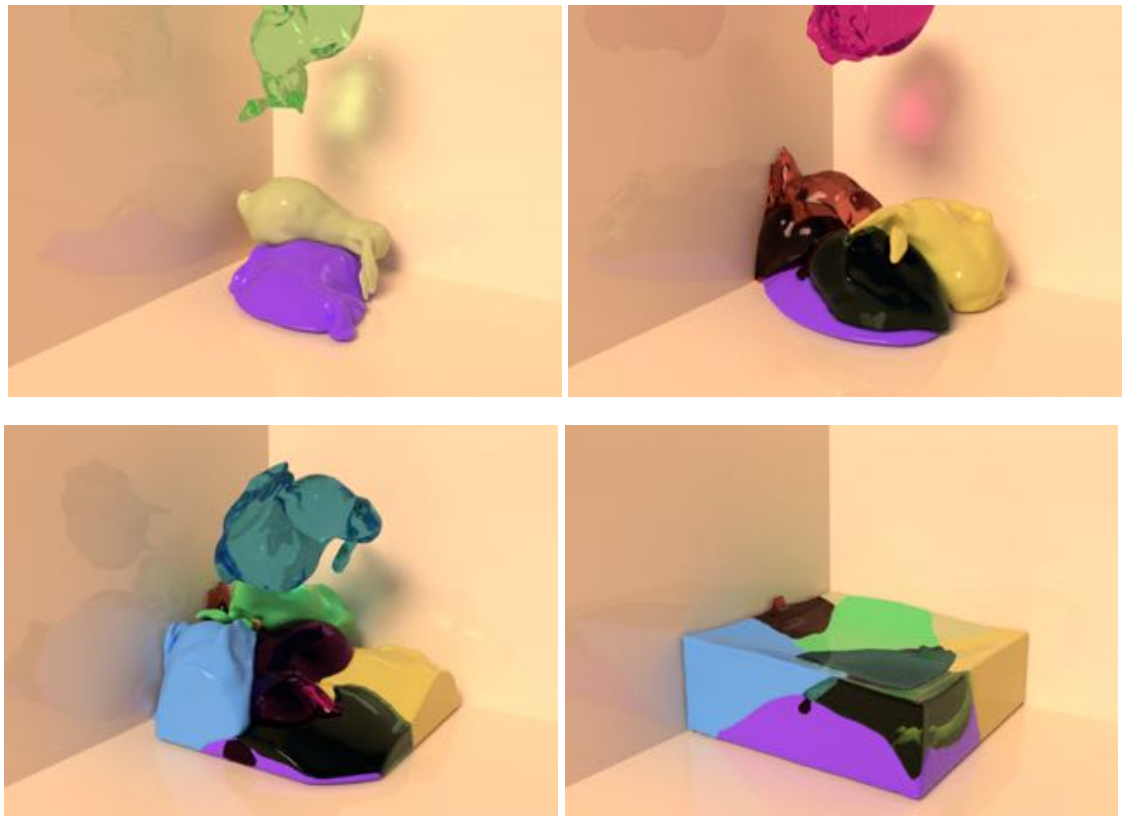


Figure 10.6: **Viscous Bunnies:** A multiphase scenario in which nine viscous bunnies with different materials are dropped into a pile.

Figure 10.7: **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.

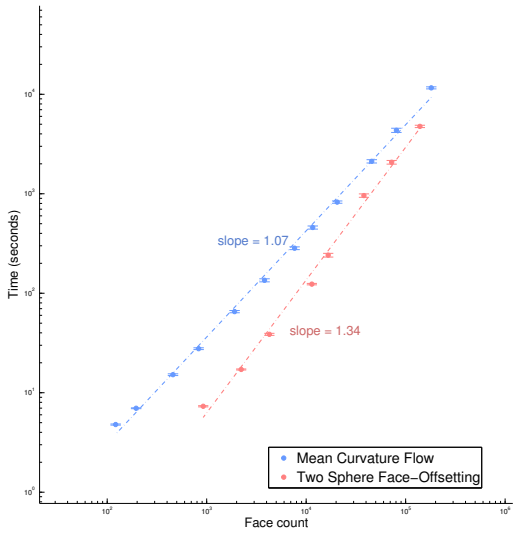


Figure 10.8: **Scaling** of the total run time against mesh complexity.

10.5 Scaling

To experimentally examine how our method's performance scales with mesh size, we conducted a series of simulations with identical initial geometry, but varying remeshing resolution, for two different simulation scenarios:

1. face offsetting on two initially disjoint spheres, and

2. mean curvature flow on a cube divided into 20 Voronoi cells.

The total simulation time was recorded for each run and plotted against the average number of triangles throughout the simulation, shown in figure 10.8. The slope of the two fit lines suggest our overall method scales close to linearly with the number of triangles in the mesh. Extending the algorithm to even larger examples should be possible by exploiting parallelism and more advanced collision-culling strategies.

Chapter 11

Discussion

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 10.2); 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. Pervasive collision detection is also a performance 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.

In order to make the algorithm scale better on a parallel architecture, operations that can be carried out concurrently should be identified. Thanks to the local nature of our remeshing operations

including the T1 processes, each operation reads and writes a relatively small patch on the mesh (at most the 2-ring neighborhoods) so in most cases one operation would not interfere with the next.

Another layer of parallelism can be added in the collision pipeline. Since we ensure collision-safety for each edit and thus the total collision checking cost can be significant, parallelizing the collision detection code is likely to provide a significant gain. It is also relatively easy to parallelize, since the narrow phase collision detection usually involves reading and performing computation on a number of nearby mesh elements without any write operation.

Part II

Surface-Only Simulation of Soap Films and Bubbles

Chapter 12

Introduction

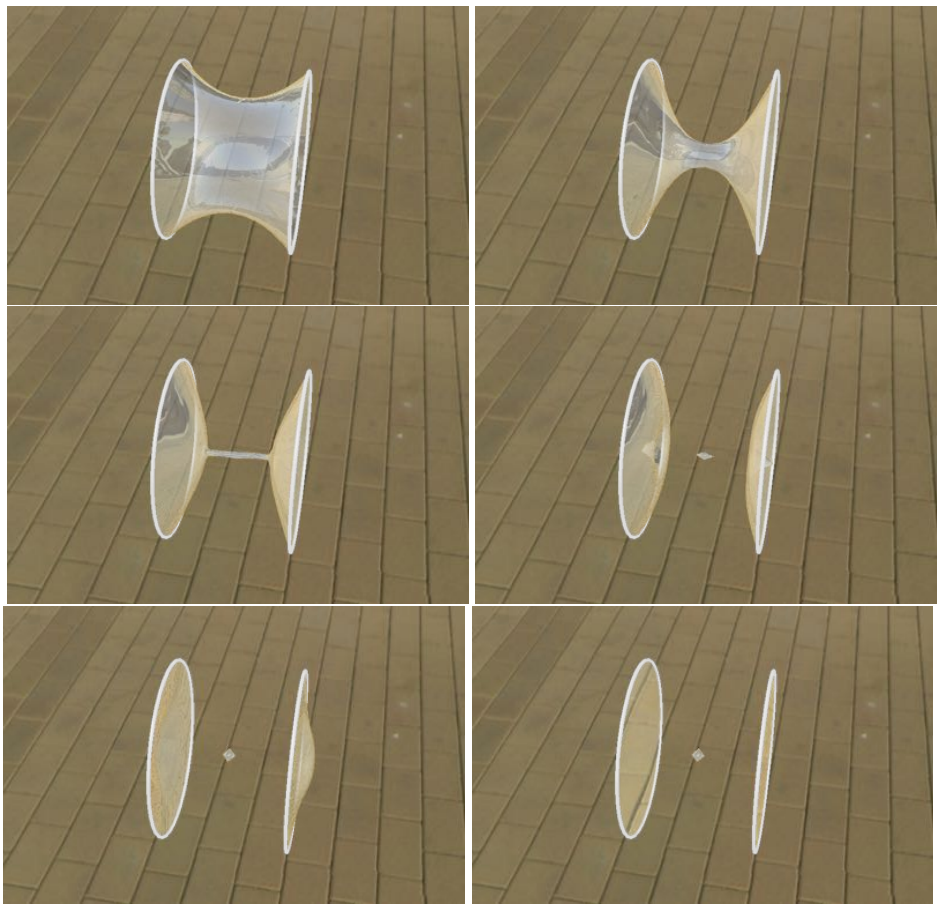


Figure 12.1: A series of snapshots of an evolving catenoid soap film joining two circular wires, an instant before the film pinches apart.

Figure 12.2: A variety of dynamic foam, film, and bubble scenarios captured by our method. Top: A small foam rearranges and settles to equilibrium. Bottom: A bubble with a wire constricting its mid-section gradually squeezes to one side.

In this part of the dissertation, I will discuss the development of a surface-only simulation framework for a particular fluid phenomenon: the air flow around soap films, which is behind the familiar motions of soap bubbles and foams. Although we encounter bubbles and foam on a daily basis while drinking coffee, washing the dishes, or playing with children's toys, the motion and structure of these phenomena are difficult to understand and even more difficult to simulate with a computer.

Soap bubbles are essentially layers of immiscible fluids: air on the inside, a thin film of liquid, and then air on the outside again. The surface tension of the liquid drives the bubble toward a shape with less surface area, while air pressure forces the bubble to maintain a constant volume. These behaviors are described by the Navier-Stokes equations, which can be exceptionally difficult to solve accurately, especially when confronted with the large density jumps, immiscible fluid interfaces, stiff surface tension forces, and extremely thin liquid surfaces required for bubble motion.

Prior work has successfully simulated foam dynamics using an Eulerian approach. These approaches utilize considerable computational resources for the detailed calculation of the air dynamics within each bubble and the accurate resolution of film geometry. We propose a new model which solves the problem more economically while guaranteeing several important theoretical properties by construction.

Our proposed model is the first numerical method for the dynamics of soap films, bubbles, and foams that is based on the equations of non-manifold vortex sheets.

We model soap film structures as non-manifold vortex sheets driven by surface tension forces. We propose a discrete model to evolve the sheet as it deforms and undergoes topology changes such as pinching, merging, and rearrangement.

We represent the geometry using a Lagrangian non-manifold triangle mesh decorated with material region tags. We store *circulation* variables on the vertices; by Kelvin's circulation theorem this ensures a circulation-preserving model *by construction*. Numerical results exhibit nearly dissipation-free dynamics.

The use of the vortex sheet equations ensures an exactly divergence-free velocity field via the Biot-Savart law, and integrating mesh vertex positions through the induced velocity field leads to excellent volume preservation over long simulations.

We derive a simple, discrete update rule for the effect of surface tension forces on the non-manifold sheet, based on a discrete measure of mean curvature. We formulate the update rule in a

manner that generalizes immediately to the case of non-manifold junctions. This yields stable film behavior even along complex foam junctions. Through the addition of a simple vertex constraint mechanism, we support wire loops and structures to control the bubbles and films, enabling a variety of familiar bubble interactions.

Our complete computational model provides a compact description of foam dynamics that requires no volumetric discretization or linear system solve, is computationally efficient as compared to traditional Eulerian techniques, and is robust enough to animate complicated foam dynamics on a standard workstation.

Chapter 13

Related Work

The behaviors of bubbles and foams are a subject of substantial interest across a variety of scientific and engineering disciplines (e.g., [Weaire and Hutzler, 2001; Weaire, 2013; Saye and Sethian, 2013]). The reproduction of their visual characteristics was addressed in a pair of articles by Glassner [2000a; 2000b]; more recent authors have explored the dynamic simulation of bubbles.

13.1 Particulate Bubbles

One simple approach to small-scale bubble dynamics is to treat each bubble as a particle, with various rules controlling their interactions, such as the early work of Kück et al. [2002]. Later work has yielded increasingly realistic behavior, and often used these models to add bubble details to traditional volumetric liquid simulations [Greenwood and House, 2004; Hong *et al.*, 2008; Kim *et al.*, 2010; Busaryev *et al.*, 2012; Patkar *et al.*, 2013].

13.2 Eulerian Bubbles

A different albeit more expensive tack is to model the bubbles themselves with volumetric *multiphase* fluid simulations. Since the thickness of the bubble's liquid film is on the order of micrometers, its mass is effectively negligible and can be treated as such; thus the main challenges are to simulate the air flow interior to the bubble with appropriate surface tension forces on the interface, and to represent the evolving non-manifold geometric structure of foams. Hong and Kim [2003] were the

first in graphics to make use of a multiphase flow solver to model bubbles, applying surface tension as a smoothed body force. Hong and Kim [2005] adapted earlier work in computational physics [Kang *et al.*, 2000], using ghost-fluid methods to incorporate surface tension boundary conditions for fully submerged bubbles. In a similar vein, Mihalef et al. [2006] simulated boiling water. To allow multiple bubbles in mutual contact and thereby support foam structures, Zheng et al. [2006] presented a regional level set method that generalizes the standard level set method to more than two regions. Zheng et al. also proposed a semi-implicit surface tension discretization to improve stability. Losasso et al. [2006] similarly extended the particle level set method to many regions to adapt Hong's work to the case of multiple interacting liquids. Kim et al. [2007] artificially inflated bubbles to compensate for volume drift. Saye and Sethian [2013] added a model for film drainage operating on different time scales.

13.3 Mesh-Based Bubbles

Since the visual appearance of bubbles is entirely dominated by the soap film itself, a seemingly natural approach is to focus modeling efforts there. Ďurikoviç [2001] proposed such a method in which the bubble surface is represented by a deformable mass-spring system on a triangle mesh. Simple surface tension forces based on the Laplace-Young equation are applied to the vertices, assuming bubbles maintain a near-spherical configuration. Similarly, Brochu [2006] proposed a preliminary boundary-element discretization of 2D droplet dynamics, and Zhang et al. [2012] used a surface-only mass-spring model for real-time animation of triangulated liquid droplets. Batty et al. [2012] modeled thin sheets of viscous liquid with surface tension using triangle meshes, but did not address bubbles. Though computationally attractive, these models concentrate the mass of the system on the (thin) liquid film, whereas in the case of bubbles the film is effectively massless relative to the surrounding air flow. Zhu et al. [2014] applied a similar thin-sheet idea to inviscid liquids and films, and modeled the influence of wind by coupling to a standard dense Eulerian grid method. This approach yielded compelling animations of blowing bubbles, soap film catenoids pinching apart, and waterbell shapes, though it cannot model multi-bubble foam structures.

A more accurate approach to mesh-based modeling of bubbles and foams *at equilibrium* is to construct a discrete surface energy based on the area of the interfaces subject to a volume constraint

per bubble, and seek to minimize it. This is the approach taken by the venerable *Surface Evolver* technique [Brakke, 1992]. Since surface tension forces are proportional to interface mean curvature, the resulting discrete surface energy gradient in this model is essentially equivalent to discrete mean curvature operators common in graphics [Pinkall and Polthier, 1993; Meyer *et al.*, 2002]. Recent work on constant-mean-curvature surfaces [Pan *et al.*, 2012] modifies the energy functional to simultaneously optimize mesh quality.

Methods for volumetric simulation of multiphase flows have also made use of explicit moving meshes. For example, conforming Lagrangian tetrahedral meshes with dynamic remeshing allow for a finite element discretization of multiple regions with area-based surface tension discretized on the interface triangles [Misztal *et al.*, 2012; Clausen *et al.*, 2013]. Rather than using a volumetric tetrahedral mesh, Part I of this thesis presented *Los Topos*, a multi-region extension of the *El Topo* triangulated mesh-based surface tracking package [Brochu and Bridson, 2009], and applied it to Eulerian grid-based volumetric multiphase liquids. The work in this Part makes use of *Los Topos* to handle the complex topological rearrangements exhibited by foams. In general, these mesh-based volumetric methods rely on a complete discretization of the fluid domain, rather than the surface alone, and have not been applied to bubble dynamics.

13.4 Vortex-Based Fluids

Vortex methods model incompressible flows by relying on an alternative basis representation [Cottet and Koumoutsakos, 2000]. Rather than choosing the primary variable to be velocity, which must be constrained to lie in the space of incompressible vector fields, vortex methods evolve the dynamics using the curl of velocity, or *vorticity*, $\omega = \nabla \times \mathbf{u}$. This family of methods guarantees incompressibility by construction, avoids numerical dissipation common to traditional methods, and provides a terser representation of the velocity field.

A variety of Lagrangian discretizations of the vorticity equation have been used including particles [Park and Kim, 2005], filaments [Angelidis and Neyret, 2005; Angelidis *et al.*, 2006; Weißmann and Pinkall, 2009; Weissmann and Pinkall, 2010; Barnat and Pollard, 2012], and sheets [Pfaff *et al.*, 2012; Brochu *et al.*, 2012], along with circulation-based Eulerian discretizations [Elcott *et al.*, 2007]. The conserved variable in all of these cases is circulation, though Mullen et al. [2009] presented an

interesting *energy-preserving* Eulerian discretization. Zhang and Bridson [2014] recently proposed an improved Particle-Particle Particle-Mesh (PPPM) strategy to accelerate the major bottleneck in Lagrangian vorticity-based schemes: evaluation of the Biot-Savart kernel to reconstruct the velocity field. Earlier work used the Fast Multipole Method (FMM) [Brochu *et al.*, 2012], truncated the required kernel evaluations [Pfaff *et al.*, 2012; Vines *et al.*, 2014], or suffered the $O(N^2)$ cost of pairwise evaluation.

Although vortex methods are most commonly applied to smoke, a few authors have considered the role of vortex sheets at or near a liquid interface. Golas et al. [2012] used a hybrid approach: a vortex particle method handles the large interior of the flow domain, and this is coupled to a regular grid-based simulator applied on a narrow band around the liquid interface or solid boundaries. In the context of deep ocean waves, Keeler and Bridson [2014] achieved a surface-only discretization by combining the *El Topo* surface tracker with the solution of a specific boundary integral equation derived under a potential flow assumption. Another use for vorticity equations at the liquid interface is to provide additional detail to a lower resolution fluid simulation [Kim *et al.*, 2009; Bojsen-Hansen and Wojtan, 2013a], similar in spirit to the use of vortex particles that augment smoke simulations with turbulent detail [Selle *et al.*, 2005; Pfaff *et al.*, 2009].

13.5 Vortex Sheets

We make use of a vortex sheet formulation of the vorticity equation. Early numerical work in this direction was carried out by Tryggvason [1988], Agishtein and Migdal [1989], and Pozrikidis [2000] among others. Recently Stock [2006; 2008] proposed a circulation-based discretization that has since been extended to smoke animation [Pfaff *et al.*, 2012; Brochu *et al.*, 2012]. The main source of vorticity for smoke is fluid density gradients (i.e., *baroclinity*), which requires some quantities to be converted back and forth between vorticity and circulation. For bubbles the vorticity source is *surface tension*, which possesses a simple integration rule that works directly in the space of circulations.

Chapter 14

Smooth Setting

A *soap film* is a thin sheet of liquid immersed in an ambient fluid medium, typically air. When the film has a spherical topology, it is known as a *soap bubble*, and when multiple such bubbles contact, they form a non-manifold network of films known as a *foam*.

We focus on centimeter-scale soap films, whose dynamics are generally dominated by (a) the surface tension of the film, along with (b) the (assumed) incompressibility and (c) inertia of the ambient air. For the centimeter length scale, the inertia of the liquid itself is negligible; the temporal evolution of the film is governed by the motion of the ambient air.

To capture this behavior, we model the interface as a *vortex sheet*.

14.1 Kinematics of a vortex sheet

A vortex sheet is an immersed surface delimiting a discontinuity in (only) the tangential component of a velocity field in 3-space. While the tangential component of velocity is generally discontinuous, the component normal to the vortex sheet remains continuous.

Consider a surface $f : M \to \mathbb{R}^3$ immersed in ambient space, where $M \subset \mathbb{R}^2$ is a reference domain. To each surface point $f(\mathbf{X})$ we associate a unit surface normal $\mathbf{n}(\mathbf{X}) : M \to S^2$.

The state of a vortex sheet is encoded by its position f and a scalar field $\Gamma : M \to \mathbb{R}$ called the *circulation* [Pozrikidis, 2000].

Circulation We note that *circulation* is most typically defined as a quantity associated to a given closed loop; in a slight abuse of terminology, borrowed from other works on vortex sheets [Pozrikidis, 2000], we take circulation to be a pointwise quantity, by identifying each point $f(\mathbf{X})$ with an arbitrary loop, as follows.

To understand the geometry behind $\Gamma(\mathbf{X})$, consider an *arbitrary* closed loop L that pierces the sheet at $\mathbf{x} = f(\mathbf{X})$ and some *fixed* reference point $\mathbf{x}_0 = f(\mathbf{X}_0)$ (refer to the inset figure). $\Gamma(\mathbf{X})$ measures the alignment of the velocity field to tangent motion along the closed loop, i.e.,

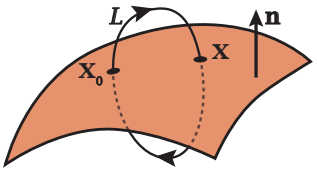


Figure 14.1: Sign convention for pointwise circulation.

$$\Gamma(\mathbf{X}) = \oint_L \mathbf{u} \cdot d\mathbf{x}.$$

Above, we emphasized the word *arbitrary* because for any two loops L_1 and L_2 both piercing the sheet only at $\mathbf{x}$ and $\mathbf{x}_0$, $\oint_{L_1} \mathbf{u} \cdot d\mathbf{x} = \oint_{L_2} \mathbf{u} \cdot d\mathbf{x}$ ([Pozrikidis, 2000]). We emphasized the word *fixed* because the scalar field Γ is meaningful only if the same reference point $\mathbf{x}_0$ is used for all evaluations of $\Gamma(\mathbf{X}), \mathbf{X} \in M$. Furthermore, selecting a different reference point $\mathbf{x}_0'$ is equivalent to uniformly offsetting the entire field by some fixed constant, $\Gamma'(\mathbf{X}) = \Gamma(\mathbf{X}) + C$. Note that Fig. 14.1 establishes a correspondence between the orientation of the surface (normal) and the sign of the circulation.

Recovering the ambient velocity from the circulation With Γ as our state variable, we can evaluate the velocity field in 3-space in two steps. First we recover the *vortex sheet strength*, $\gamma : M \to \mathbb{R}^3$,

$$\gamma = \mathbf{n} \times \nabla_f \Gamma, \tag{14.1}$$

where ∇_f is the surface gradient operator for f. Observe that by construction γ is tangential to the surface (because of $\mathbf{n}\times$) and divergence-free (because it is a right-angle rotation of a gradient field). Note that the direction of γ is determined by the local $\nabla_f \Gamma$ direction, independent of the choice of the reference point $\mathbf{x}_0$. Vortex sheet strength represents vorticity confined to the sheet, under the relationship $\omega = \gamma \delta(f)$, with Dirac delta δ.

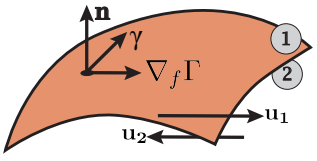


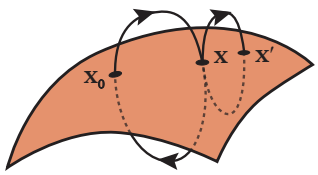
Figure 14.2: Conventions for vortex sheet strength.

From the vortex sheet strength we recover the ambient *velocity*, $\mathbf{u} : \mathbb{R}^3 \to \mathbb{R}^3$ through the Biot-Savart integral

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_f \frac{\gamma \times (\mathbf{x} - \mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|^3} df'. \tag{14.2}$$

The field $\mathbf{u}$ is divergence-free by construction. Thus, a velocity field derived from vortex sheet kinematics is automatically able to capture the dynamics of incompressible fluids without explicit enforcement of the incompressibility condition.

Next, we consider two nearby points $\mathbf{x}$ and $\mathbf{x}'$ on the vortex sheet. The integration loop for circulation at $\mathbf{x}'$ intersects the sheet at $\mathbf{x}'$ and the reference point $\mathbf{x}_0$, and can be broken down as the sum of two loops: one between $\mathbf{x}_0$ and $\mathbf{x}$, and one between $\mathbf{x}$ and $\mathbf{x}'$. The former corresponds to the circulation at $\mathbf{x}$. Therefore, the difference between $\Gamma(\mathbf{X})$ and $\Gamma(\mathbf{X}')$ is the velocity integral around the latter loop, which is represented by the tangential velocity jump $\mathbf{u}$ at $\mathbf{x}$ as the two points approach each other. This reveals the following relation, which is needed in Section 14.3 (see [Pozrikidis, 2000] for details):



$$\Delta \mathbf{u} = \nabla_f \Gamma. \tag{14.3}$$

14.2 Dynamics of self-advection

In the absence of external forces, the evolution of the circulation field is given by Kelvin's circulation theorem:

$$\frac{\mathrm{D}\Gamma}{\mathrm{D}t} = 0, \tag{14.4}$$

where D denotes the *total* (equivalently *material* or *advective*) derivative.

Consider the consequences on implementation. As the vortex sheet deforms under the advection induced by the ambient air, the scalar circulation field remains constant. This property sets circulation apart from a vorticity-based representation of the velocity field, avoiding explicit handling of vortex stretching and simplifying the numerical implementation.

The question arises whether Kelvin's circulation theorem is applicable since the vortex sheet represents a tangential velocity jump which seems to 'shear' the material loop open. Intuitively,

the (tangentially discontinuous) vortex sheet representation can be understood as the limit of a (tangentially continuous) finite-thickness sheet model as the thickness tends to zero, and thus the circulation theorem is well-formed. For a detailed derivation, we refer the interested readers to the vortex sheet formulation [Pozrikidis, 2000].

14.3 Dynamics with surface tension

With surface tension, the circulation field no longer remains constant; instead its evolution is governed by a simple, local law.

Neglecting viscosity and external forces, we have the momentum equation for the incompressible Euler equations

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\frac{\nabla p}{\rho} \tag{14.5}$$

everywhere away from the sheet, where ∇ is the canonical gradient of $\mathbb{R}^3$. Taking the difference of the limiting values across the sheet gives the relation

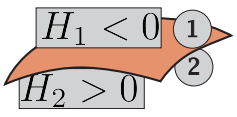
$$\frac{\mathrm{D}\Delta\mathbf{u}}{\mathrm{D}t} = -\frac{\nabla(p_1 - p_2)}{\rho} = -\frac{\nabla_f(p_1 - p_2)}{\rho} \tag{14.6}$$

on the sheet. For the second equality, we have used the fact that $\nabla(p_1 - p_2) = \nabla_f(p_1 - p_2)$, which holds because the velocity jump $\Delta\mathbf{u}$ is tangential to the surface.

In the equations above, the two pressures p_1 and p_2 are those of the air on two sides of the film. If we consider the (very thin) liquid volume in the film, we actually have three distinct regions (air-liquid-air), separated by two liquid-air interfaces, and the difference between p_1 and p_2 is therefore the sum of the two pressure jumps (from air on one side of the film to liquid, and from liquid to air on the other side) induced by the surface tension in the two liquid-air interfaces. As a result, the total pressure jump $p_1 - p_2$ is given by

$$p_1 - p_2 = \sigma(H_1 - H_2), \tag{14.7}$$

where p_1, p_2 are the pressures on each side of the surface, σ is the surface tension coefficient and H_1, H_2, are the (signed, scalar-valued) mean curvatures of the *two* liquid-air interfaces possessed by a soap film immersed in air. We use the convention that the mean curvature is positive when the interface is convex toward the liquid.



For a manifold patch of the vortex sheet we simply have that $H_1 = -H_2$, since the conceptual "two" sides of our idealized thin sheet geometrically coincide while possessing opposing orientations. However, we have explicitly broken out the two contributions as a foreshadowing of our treatment of triple-junctions.

Substituting (14.3) and (14.7) into (14.6) yields

$$\frac{\mathrm{D}\nabla_f \Gamma}{\mathrm{D}t} = -\frac{\sigma}{\rho}\nabla_f (H_1 - H_2). \tag{14.8}$$

By linearity of differentiation, we find $\frac{\mathrm{D}\nabla_f \Gamma}{\mathrm{D}t} = \nabla_f \frac{\mathrm{D}\Gamma}{\mathrm{D}t}$. Applying this rule and integrating both sides gives

$$\boxed{\frac{\mathrm{D}\Gamma}{\mathrm{D}t} = -\frac{\sigma}{\rho}(H_1 - H_2),} \tag{14.9}$$

where we have chosen to discard the integration constant, recalling that Γ is only defined up to the choice of a constant, equivalently the choice of a reference point. For an alternative derivation, refer to Pozrikidis [2000] who builds on an argument by Baker et al. [1982].

Note that circulation evolves in proportion to the *scalar-valued* mean curvature; this may be contrasted against other representations of the velocity field, for which the consideration of surface tension can involve the *vector-valued* mean curvature normal.

Discussion We have arrived at the temporal evolution of the circulation Γ subject to inertia and surface tension. The attendant evolution of the ambient velocity field thus accounts for **inertia**, **surface tension**, and (due to the Biot-Savart law) **incompressibility**.

The simplicity and locality of the evolution (14.9) become pronounced when one adopts a Lagrangian discretization of the soap film: the update to the state variables requires simply the evaluation of the mean curvature. To our knowledge this circulation-based temporal evolution of vortex sheets subject to surface tension has not previously been employed to model soap films.

Chapter 15

Discrete Setting

15.1 Spatial discretization

Nonmanifold triangle mesh without boundary We represent the geometry of our discrete soap film using a piecewise linear triangulated surface, i.e., a triangle mesh. The mesh contains non-manifold edges and vertices where soap films meet along *Plateau borders* [Weaire and Hutzler, 2001].

Our mesh is always watertight, i.e., it does not have open boundaries. We accommodate scenarios where the soap film has a prescribed open boundary (e.g., a wire) by ignoring a part of the mesh in the dynamics, as discussed in Section 15.6.

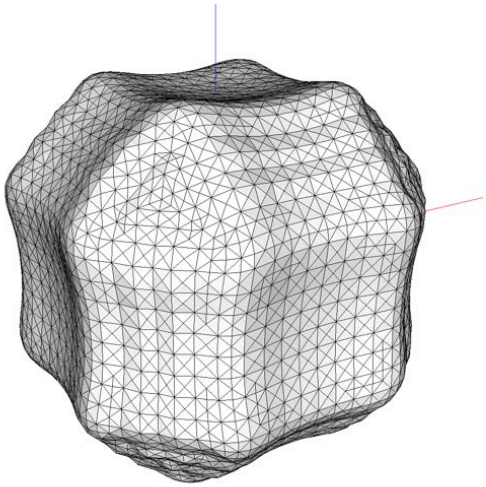
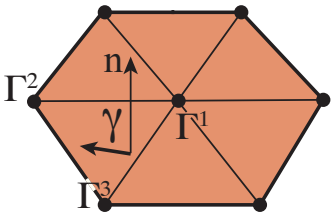


Figure 15.1: Oscillating cube.

Regions We *enumerate* each watertight region of the mesh (i.e., each bubble volume), assigning a unique integer *index* to each region. Two regions with a common interface are called a *region-pair*.

Circulation For the purpose of modeling vortex sheets, we have interpreted circulation as a pointwise quantity [Stock, 2006]; that is, it is a scalar field restricted to the surface. Recall that the orientation of the circulation gradient together with the surface normal determines the

orientation of the vortex sheet strength: We therefore sample the scalar circulation field Γ at *vertices* of the triangle mesh. This gives rise to a piecewise linear field over triangles, whose gradient (and therefore also the sheet-strength) is piecewise constant, and which coincides with the vertex-based discrete curvature measure used below. (This is to be contrasted with previous work which placed pointwise circulation on edges [Stock, 2006; Pfaff *et al.*, 2012; Brochu *et al.*, 2012].)



$\gamma = \mathbf{n} \times \nabla_f \Gamma.$

We associate a distinct scalar circulation field to each region-pair. Thus, manifold vertices store one circulation scalar, while triple-junction vertices store *three* circulation scalars, one for each region-pair incident to the vertex. Quadruple-points, which also arise frequently in stable foams, will store *six* circulations, and so forth.

Region-pair interfaces and circulations must satisfy a consistent sign convention. Our implementation chooses the interface normal as oriented from the higher index region to the lower; the recovered velocity is *independent* of this arbitrary choice, so long as circulation variables are consistently oriented relative to the surface normal, using the convention codified in Fig. 14.1.

15.2 Temporal discretization

Since we work in the Lagrangian frame of the moving surface, the time-evolution equation for circulation (14.9) indicates that we can simply advect vertices according to the ambient velocity, and integrate surface tension forces into the vertex-based circulations (§15.3). In the absence of surface tension, circulations are carried along and naturally conserved. We apply forward Euler to integrate the vertex positions under the velocity field $\mathbf{u}$ induced by circulations, as computed from the Biot-Savart law (§15.4).

15.3 Integrating Surface Tension

We integrate surface tension forces into the circulation using the (discrete, scalar, signed, pointwise) mean curvature at the liquid-air interface. Recall that since we have only surface tension forces, and no baroclinic terms, we avoid the need to convert back and forth between circulation and vorticity:

integration is performed directly in the space of circulations.

We take a forward Euler step, updating the circulation associated to each region-pair (i, j), $i < j$ via

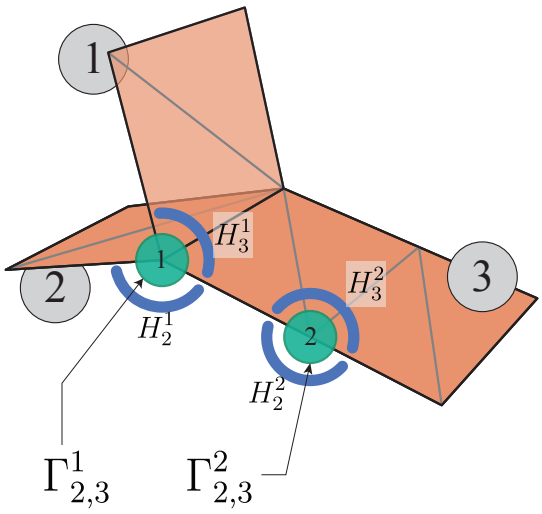
$$\Delta\Gamma^v_{i,j} = -\frac{\sigma\Delta t}{\rho A}(H^v_i - H^v_j) \tag{15.1}$$

where A is the Voronoi area of the vertex [Meyer *et al.*, 2002], used to convert from integrated to pointwise curvature, and H^v_i, H^v_j are the integrated signed scalar mean curvatures at vertex v due to the liquid-air interface at incident regions i and j, respectively. Recall from the smooth setting that H^v_i is positive when region i is outwardly convex at v.

To evaluate H^v_i, we consider only the triangles incident to vertex v and region i. We sum the signed scalar mean curvature of incident edges. For each edge e, we take $H^e_i = |e|\theta$, where $|e|$ is the edge length, and θ_i is the dihedral angle given by $\cos\theta_i = \mathbf{n}^1_i \cdot \mathbf{n}^2_i$, and $\mathbf{n}^1_i$, $\mathbf{n}^2_i$ are the outwardly oriented normals of the two incident triangles to e for region i [Cohen-Steiner and Morvan, 2003]. The sign of H^e_i is unambiguous, based on the outward orientation of region i. Then the signed scalar vertex curvature, H^v_i, is

$$H^v_i = \frac{1}{2}\sum_e H^e_i \tag{15.2}$$

where the factor of one half accounts for the fact that the integrated edge curvature is divided amongst its two vertices.



This unified treatment is applied identically whether a vertex lies on simple manifold regions, or on a complex triple, quadruple, or even higher-order junction.

Plateau borders At a triple-junction the total effect of surface tension will be zero on all of the three circulations, $\Gamma^v_{1,2}, \Gamma^v_{1,3}, \Gamma^v_{1,3}$, only when all three curvatures are equal, $H^v_1 = H^v_2 = H^v_3$. This recovers the proper 120° angle Plateau border equilibrium, as dictated by Plateau's laws, and illustrated in Fig. 15.3. A similar behavior occurs for quadruple-points, where the four curvatures become equal

with angles of $\theta \approx 109.47°$. Although higher order junctions can also balance in this manner, they are typically unstable.

Discussion For our mean curvature computation, we initially attempted to use the familiar cotan Laplace curvature discretization, which gives rise to a mean curvature normal *vector*, $H\mathbf{n}$; however, our dynamics requires *signed mean curvature scalars*, H. While the latter could theoretically be extracted from the former by an inner product with an appropriately oriented surface normal, we encountered difficulties in the use of various options for discrete vertex normals $\mathbf{n}$. Without a discrete unit normal $\mathbf{n}$ that is guaranteed to be in precise alignment with $H\mathbf{n}$, the scalar mean curvature H extracted from $H\mathbf{n}$ may have an incorrect sign.

15.4 Evaluating Biot-Savart

Given the circulation values stored on each vertex, including multi-valued circulations on non-manifold vertices, we can convert to vortex sheet strength, and use the Biot-Savart law to find the velocity at any point in the domain.

From $\gamma = \mathbf{n} \times \nabla_f \Gamma$ we uniquely recover the vortex sheet strength on a per-triangle basis. The triangle normal $\mathbf{n}$ is computed in standard fashion recalling our earlier orientation convention, and the scalar circulations on the vertices define a linear circulation field on the triangle from which the constant gradient can be readily recovered. Since the vortex sheet strength is a per-triangle quantity, no changes are needed to evaluate Biot-Savart near non-manifold geometry.

Our expression for the Biot-Savart law, with regularization per Pfaff et al. [2012], is:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_S \gamma(\mathbf{x}') \times \frac{\mathbf{x} - \mathbf{x}'}{(|\mathbf{x} - \mathbf{x}'|^2 + \alpha^2)^{3/2}} d\mathbf{x}'. \tag{15.3}$$

Thus the velocity $\mathbf{u}$ at a point $\mathbf{x}$ in space is an integral over the surface, which we discretize with one-point quadrature at triangle barycenters. We set the regularization parameter α to half the mean mesh edge length. We perform a direct $O(N^2)$ evaluation of this integral; proven FMM [Brochu *et al.*, 2012] or PPPM [Zhang and Bridson, 2014] techniques exist to drop this to essentially $O(N)$.

15.5 Integrating Position

The velocity can be evaluated at the mesh vertices in order to perform time integration of the vertex positions. We again use forward Euler, although higher order schemes could be readily incorporated (both here and when integrating surface tension) at the cost of additional Biot-Savart evaluations. Since we use forward Euler to integrate surface tension into circulation (effectively updating velocity), and then use forward Euler to advect positions based on the (updated) velocity, we are in fact applying a Symplectic Euler scheme.

15.6 Solid Interaction

To support simple solid interactions with wire loops (e.g., Fig. 15.3), we develop a projection method to constrain the motion of the vertices. Given a set of constrained vertices and their desired trajectories, our approach locally modifies the circulations of the constrained vertices to induce the necessary sheet velocities.

Since the tangential motion of the sheet does not necessarily correspond to tangential motion of the actual liquid film [Pozrikidis, 2000], we seek only to constrain the normal motion of the sheet by modifying circulations. In the tangential direction, we simply snap the vertices to their desired position in the tangent plane. This reduces the number of active constraints from $3n$ to n for a set of n constrained vertices. Conveniently, each (manifold) vertex carries a single circulation, giving us exactly n constraints for n degrees of freedom. We can recover a square, dense linear system to find the necessary circulations as follows.

We describe the computation of the Biot-Savart integral for this set of vertices as a matrix B applied to the affected circulation variables, Γ_c, yielding a relation $\mathbf{u} = B\Gamma_c$. We seek a change in circulations $\Delta\Gamma_c$ such that a vertex's normal velocity component changes by $\Delta\mathbf{v} \cdot \mathbf{n}$, where $\Delta\mathbf{v}$ represents the difference between current and target velocity. We must solve the linear system

$$\mathbf{N}^T B \Delta\Gamma_c = \mathbf{N}^T \Delta\mathbf{v} \tag{15.4}$$

where N is a matrix composed from the discrete vertex normals of the vertices. For robustness against the possibility of (near-) singularity we employ Tikhonov regularization, by adding the identity

matrix scaled such that its influence is small and independent of time step and mesh resolution:

$$(B^T\mathbf{N}\mathbf{N}^T B + \lambda^2\mathbf{I})\Delta\Gamma_c = B^T\mathbf{N}\mathbf{N}^T\Delta\mathbf{v}, \tag{15.5}$$

where $\lambda = 0.025/\bar{e}$ is the regularization parameter, and $\bar{e}$ is the average edge length. This corresponds to a mild preference for smaller changes in circulation. The computed $\Delta\Gamma_c$ is directly added to the circulation variables and the simulation continues.

At times, we would like to animate an *open* soap film bounded by a wire, although the *Los Topos* package currently requires that all regions be closed and watertight. We therefore approximate open scenarios by adding a film of inactive and non-rendered ghost triangles that do not participate in the dynamics (i.e., circulation evolution, Biot-Savart evaluation, etc.), but *do* close up the open region. Moreover, we observe that in the presence of an open boundary, having vorticity in the interior of the soap film is insufficient, since those degrees of freedom cannot adequately encode a path of air flow from below the film to above (and *vice versa*) around the film boundary. As a result the flow behaves as if the air volumes above and below the film are conserved separately, even though the two regions are well connected outside the sheet boundary. The top row of Figure 15.2 illustrates this phenomenon in 2D. We therefore place an additional vorticity degree of freedom on each open boundary edge along the edge tangent direction (Figure 15.2 bottom row), and solve for their strength using the same projection method above.

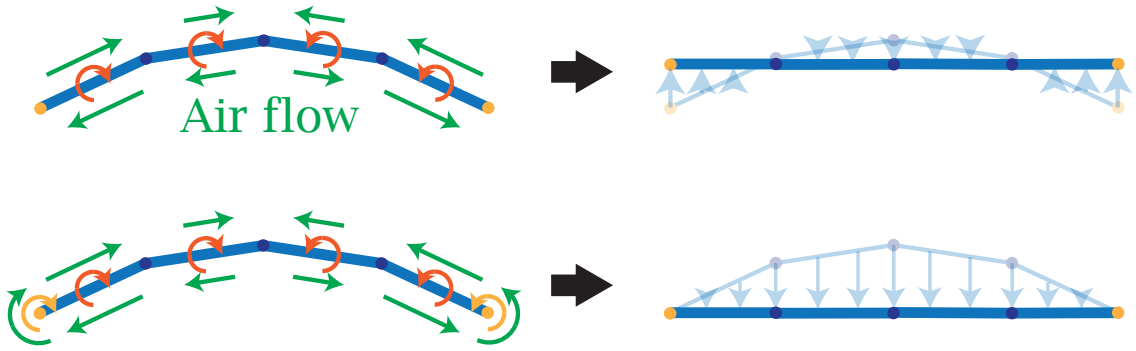


Figure 15.2: 2D illustration of the treatment of vorticity at open boundary. The thick blue lines are the vortex sheet, with the yellow dots being the boundary. Top: The interior vorticity (orange arrows) lacks the ability to represent an airflow path from below to above the sheet. Bottom: The additional vorticity degrees of freedom (yellow) provides the airflow path around the open boundary.

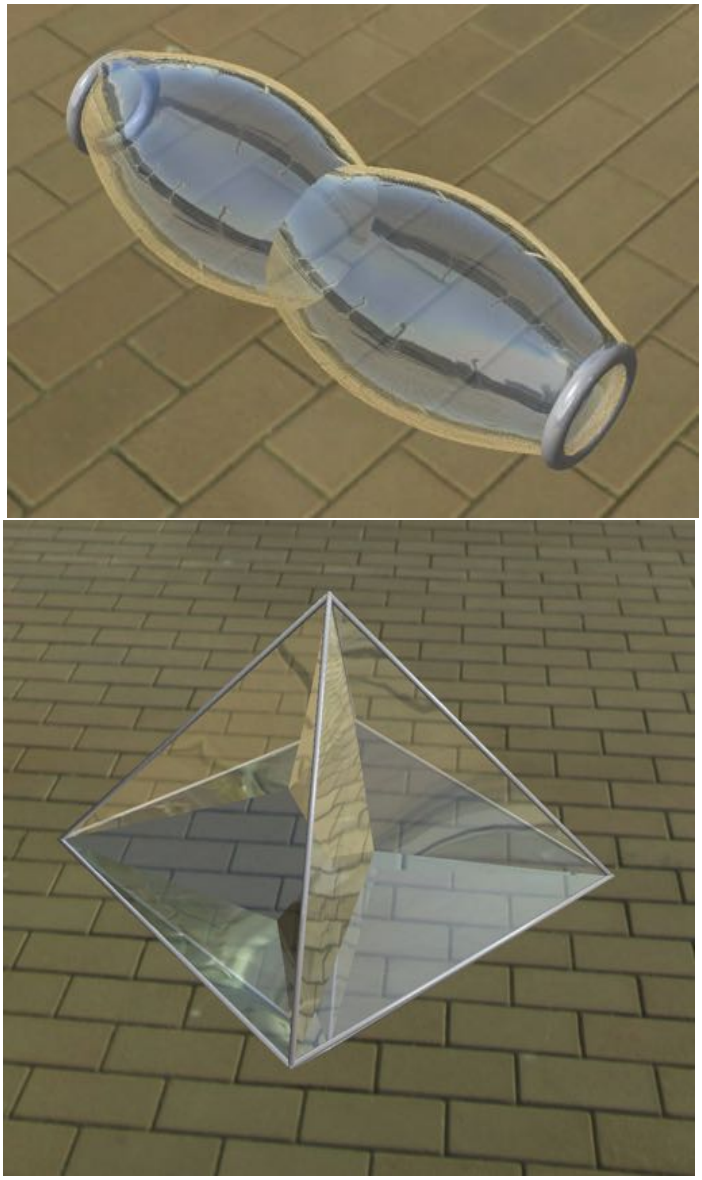


Figure 15.3: Top:A double-bubble being gradually pulled apart by two wire loop (constraints) on either end. Bottom:A stable non-manifold film structure spanning an octahedron-shaped wire.

Chapter 16

Time Integration

Our complete time integration loop is given by Algorithm 3.

Algorithm 3 Time Integration Loop

while simulating **do**

Integrate Surface Tension into Circulations (§15.3)

Recover Velocity from Circulation (Biot-Savart) (§15.4)

Integrate Vertex Positions Using Velocity (§15.5)

Enforce Constraints via Projection (§15.6)

Perform Remeshing and Topology Changes, *Los Topos* (§16.1)

end while

16.1 Discrete Mesh Evolution

Our discrete soap film model relies on the ability to represent non-manifold liquid interfaces as Lagrangian triangle meshes, and to treat the topological changes that arise as bubbles deform and rearrange. We make use of the *Los Topos* multimaterial surface tracking package introduced by Da et al. [2014b], which was tailor-made for these types of topology changes, including bubble splitting, multi-bubble merging, and so-called T1 processes (essentially foam rearrangement).

We treat *Los Topos* largely as a black box: we provide proposed vertex trajectories for the vertices of the triangle mesh, and the algorithm finds an updated watertight mesh that satisfies these trajectories as nearly as possible, after handling topological changes and remeshing. Since we rely

on storing circulation data at mesh vertices, we must also reassign circulation values to affected vertices whenever a remeshing operation is performed. We therefore use a slightly smaller set of operations to reduce resampling: edge splitting, edge collapse, T1 processes, and snapping. For edge splitting, edge collapse, or snapping (coalescing nearby unconnected vertices), we project the new vertex position onto the old mesh and compute a new circulation value by barycentric interpolation over triangles. For a T1 process, which amounts to duplicating and separating an existing vertex, the two resulting vertices are assigned the circulation of the old vertex. (We eschew edge flipping, which changes geometry without changing vertices leading to temporal discontinuities, and vertex smoothing, which moves nearly all vertices and leads to oversmoothing of circulation.)

Chapter 17

Evaluation

We evaluate the effectiveness of our method with a range of examples. The results are organized to show how our method effectively preserves circulation, handles solid boundaries, and stably simulates non-manifold foam configurations. The size of the simulation meshes and simulation times are summarized in Table 17.1. The maximum percentage deviations in the total volume of enclosed air over the entire simulations for the cube, double-bubble, and foam rearrangement examples was 2.2%, 2.4%, and 0.05% respectively.

Circulation preservation The circulation-based discretization in Section 15 allows our bubbles to preserve circulation over long simulations while being relatively insensitive to the effects of remeshing. Figure 15.1 uses an initially cube-shaped bubble to illustrate how our method maintains bubble oscillations over very long periods. The bubble remains stable and temporally smooth despite re-meshing and re-sampling the circulation variables as often as needed to preserve mesh quality. The sphere also maintains a remarkable degree of symmetry throughout.

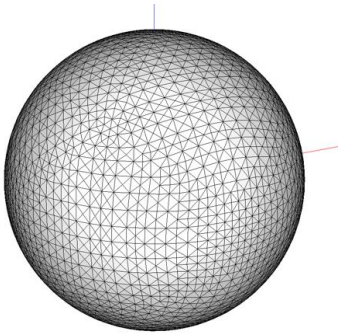


Figure 17.1: The equilibrium bubble.

Note that our explicit integration scheme is not unconditionally stable, and we must introduce some damping (by slightly diffusing Γ along the sheet) in order to maintain stability. Conveniently however, this can also provide a reasonable visual approximation of air viscosity, as we demonstrate by re-running the same oscillating

Test	Vertices	Faces	Simulation time per frame (s)	Number of frames	Total time (min)
Cube oscillation	4274	8544	1.29	6736	145
Double bubble	5204	10496	1.94	3560	115
Octahedron	854	1951	0.319	564	3
Bubble in a ring	1122	2240	0.309	1533	8
Catenoid pinching	1153	2294	0.96	1500	24
Pulling double bubble apart	1003	1998	0.276	5000	23
Bubble cluster	2734	5787	0.695	2214	25
Bubble popping	1971	4099	0.321	28000	150
Sweeping ring	3070	6129	0.504	7500	63

Table 17.1: Simulation time for various examples. A frame always corresponds to one time step of simulation.

cube bubble with a substantially increased damping coefficient. As expected it settles rapidly to a stable spherical configuration (Fig. 17.1), demonstrating that our method achieves the desired steady-state equilibrium for this canonical example.

Moving beyond the simple sphere topology, we perform a similar oscillation test on an initially stretched *double bubble*, shown inset. Here, and throughout our examples, the velocity of the ambient air flow plays the dominant role in the motion, in this case generating the interesting wobbling behavior; this velocity is compactly encoded by the triangle mesh. Furthermore, we see that surface tension behaves naturally on the triple curves and the interior interface joining the two bubbles.

Figure 17.2: A double-bubble.

Interaction with solid boundaries Our solid boundary condition explained in Section 15.6 allows us to simply and stably simulate bubbles interacting with solid obstacles like metal wires or bubble wands. Figure 12.2, center, shows how pulling apart two metal rings can correctly stimulate the catenoid soap film shape and eventual separation of a single soap film. Here we have also made use of non-simulated triangles to illustrate that our numerical method supports open films (although our underlying surface tracker does not). When the separation occurs, sharp geometric features are created near-instantaneously, but again the simulation remains stable.

Figure 15.3, left, shows a double-bubble being pulled apart by two metal rings until they undergo splitting and become fully separated. While the pinching behavior is similar to the catenoid case, here we see a triple junction curve shared by two bubbles collapsing to a point and separating as opposed to the earlier example involving the collapse of a single film.

Figure 12.2, right, illustrates a squished bubble constrained by a stationary loop. The surface tension forces lead the bubble to gradually squeeze to one side as the combined system relaxes towards a more stable configuration. Such an example would be difficult to achieve with a standard Eulerian scheme due to the complex boundary conditions created by the wire.

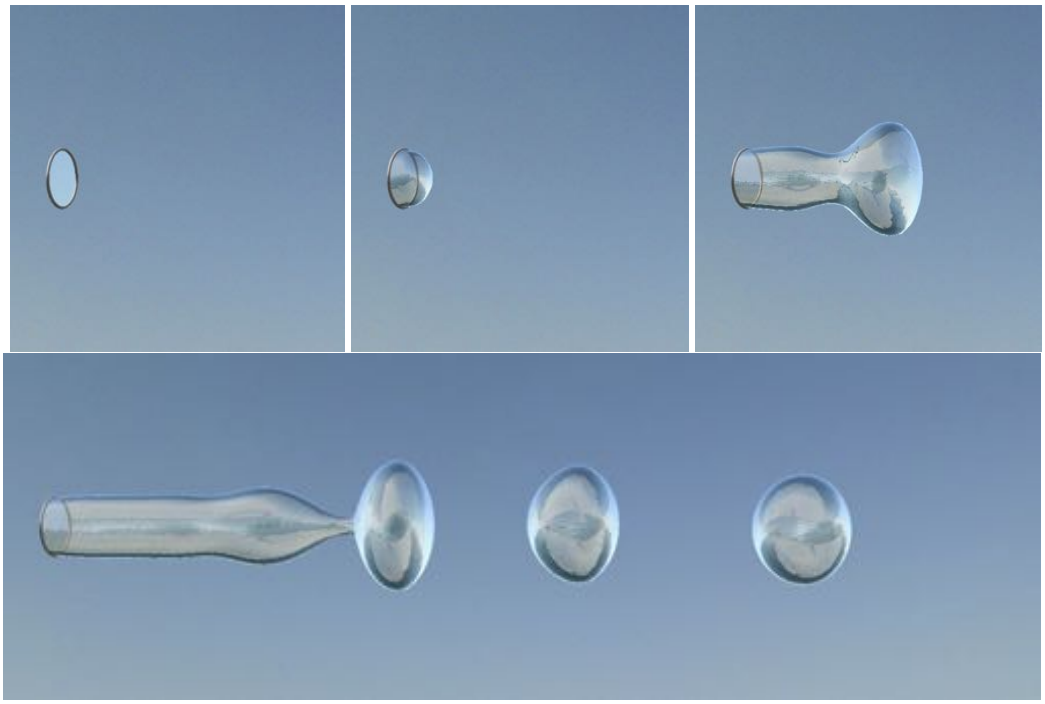


Figure 17.3: Sweeping a metal ring through space to blow bubbles.

Figure 17.3 simulates blowing bubbles from a wire ring, by sweeping the ring through space in the axial direction. The inertia of the air mass pushes the initially flat soap film to bulge out until it pinches off into individual bubbles.

Non-manifold foam dynamics Next we step from single and double film and bubble configurations up to the more general case of highly complex non-manifold film and foam structures. Figure 15.3, right, shows a network of non-manifold soap films stretched over an octahedron, a common physics

experiment for children. We initialize the geometry a short distance from its equilibrium, and observe that it quickly settles and remains stable.

Our final pair of test cases involves a free-floating foam consisting of ten bubbles combined together in a complicated, non-manifold non-equilibrium state (Fig. 12.2, left). Upon initiating the dynamics, the bubbles naturally shift and rearrange until they find an equilibrium. To up the ante, we take the same drifting foam and intermittently puncture a random bubble by directly deleting the relevant interface triangles. At each pop the foam suddenly ripples and the bubbles quickly shift into new non-manifold equilibrium configurations in accordance with the dynamical laws. In the accompanying video we can see that there appears to be net lateral motion of the bubble cluster upon bubble popping; this motion does not necessarily indicate a violation of the law of momentum conservation. As the mass of the system is in the air, popping bubbles changes the momentum that the bubble cluster 'visually' represents, therefore as the air is moved by the advancing/receding soap films, it is not surprising to see the bubble cluster's visual center shift while the total air momentum is conserved. As a simpler example, figure 17.4 shows a large bubble being deformed from the spherical shape due to the presence of a smaller bubble to its right. As the small bubble is suddenly popped, the large C-shaped bubble tries to recover the spherical steady state by pushing away the air volume at the original location of the small bubble towards the right, which in turn propels the large bubble to the left, thus creating an apparent net motion indicated by the green arrow.

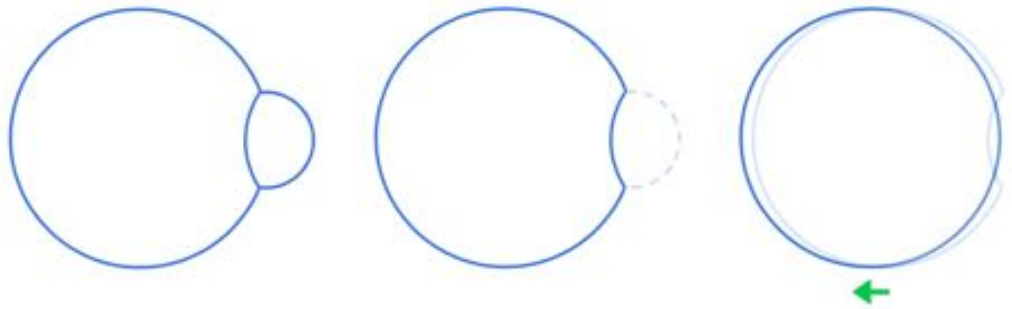


Figure 17.4: The net motion of a deformed large bubble after the small bubble next to it pops.

Chapter 18

Accelerating the Biot-Savart integral by Fast Multipole Method

The proposed method relies on a key step, which is the computation of the velocity field from the vortex sheets, using the Biot-Savart integral:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_S \frac{\gamma(\mathbf{x}') \times (\mathbf{x} - \mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|^3} d\mathbf{x}'. \tag{18.1}$$

Since this boundary integral needs to be evaluated at each vertex during mesh movement, a naive implementation that directly discretizes the integral through quadratures results in a quadratic time complexity in the mesh size. In general, fast summation techniques such as fast multipole method (FMM) have been developed to improve the scaling of such computation.

Fast Multipole Method Template Library (FMMTL) [Cecka and Layton, 2015] is a generalized framework for FMM with various kernels. It is a versatile template library that accelerates matrix-vector multiplications where the matrix is a kernel matrix resulting from integrals like Biot-Savart, while encapsulating the implementation details of FMM from the client application. We employ this library to replace the original quadratic-time implementation of the Biot-Savart integral.

18.1 Time Complexity

The implementations with and without FMMTL integration are compared on a test where a single bubble with an ellipsoidal initial shape is allowed to relax towards and then oscillate around the

spherical equilibrium configuration. The number of vertices in the mesh is varied roughly from 400 to 40,000. Each mesh resolution results in a series of frames, whose individual computation time and precise vertex counts are recorded, and plotted in log-log in Fig. 18.1 with respective linear fits.

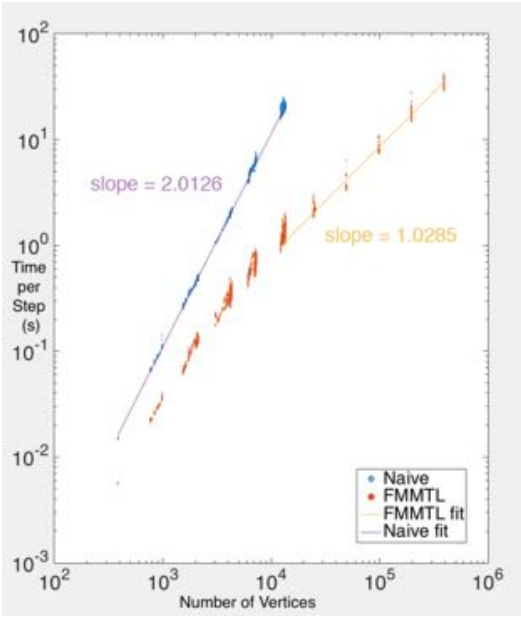


Figure 18.1: The scaling of computation time with mesh size.

Note that the computation time recorded above consists of only the time spent on Biot-Savart integral and excludes all other components of the simulation such as mean curvature computation. In this bubble oscillation example, the Biot-Savart integral is by far the performance bottleneck in the naive implementation, usually two orders of magnitude more expensive than the second component which is remeshing.

The plot confirms the quadratic time complexity of the naive implementation, evident from the slope of 2. The timing for the FMMTL version shows a slope of 1 on the right part of the curve which dips down as the vertex number decreases, consistent with a nonlinear curve $\log T = \log N + \log\log N$ resulting from the expected time complexity of $T = O(N \log N)$. Around $N = 10^4$ vertices, the FMMTL version is more than 10 times faster than the naive version.

18.2 Large Foam Tests

The accelerated Biot-Savart integral allows for simulation of foams with more bubbles. Fig. 18.2 shows a bubble cluster, starting from a regular 5 by 5 by 5 cubic grid. Fig. 18.3 shows an even bigger foam cluster with 512 bubbles. In both examples, the individual bubbles gradually slide past each other to seek a more energetically favorable arrangement, transforming the global shape of the whole cluster to a near spherical configuration in the process.

Figure 18.2: Bubble cluster rearrangement test with 125 bubbles. The bubbles start off from a regular grid and gradually rearrange into a near-spherical cluster.

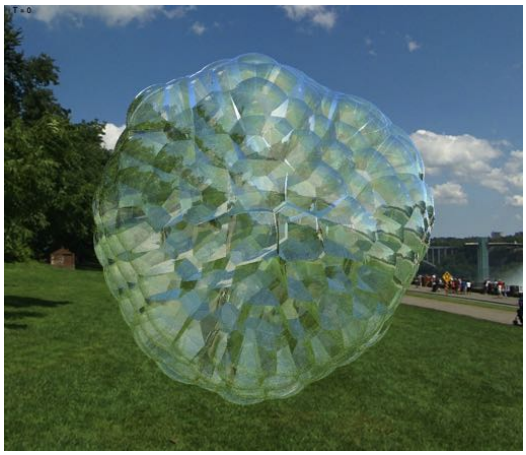


Figure 18.3: Bubble cluster rearrangement test with 512 bubbles.

Chapter 19

Discussion

The surface-only nature makes our proposed technique to foam simulation markedly different than earlier alternatives, yielding a number of trade-offs, including limitations.

From the standpoint of versatility, purely Eulerian methods currently dominate. Given their long line of investigation, Eulerian methods support a broad range of features that we have yet to explore, including accurate modeling of viscosity, application to volumetric liquids, and bidirectional coupling with complex rigid and deformable objects. On the other hand, Eulerian methods require that bubbles be significantly larger than the grid scale in order to resolve the desired velocity field and film geometry without excessive dissipation or volume loss; even then, explicit volume control may be necessary. Although circulation- or energy-preservation has been considered by a few Eulerian schemes [Elcott *et al.*, 2007; Mullen *et al.*, 2009], conserving these quantities in an Eulerian setting remains challenging in the presence of fluid interfaces. In contrast to Eulerian methods, our approach efficiently handles all the dynamics on the mesh surface itself, notably without the need for a large linear solve; furthermore, circulation is preserved in a natural and simple manner, and in particular in the presence of non-manifold fluid interfaces.

In comparison to related surface-based discretizations of thin sheets and films [Batty *et al.*, 2012; Zhu *et al.*, 2014], our focus on centimeter-scale soap films justifies neglecting certain quantities: we do not track mass or thickness of the liquid, nor *tangential* component of the dynamics. Instead, we focus on the complex geometry of non-manifold foam structures and its attendant topology changes, moreso than any existing purely triangle mesh-based scheme for foam. Our treatment of air-dominated dynamics also requires no coupling with a grid-based solver.

Due to the need of encoding the velocity field on a surface representation (also inherent to the vortex sheets method), our current derivation assumes inviscid incompressible flow, so that proper viscous boundary layers (e.g., vortex shedding) and divergent flows (e.g., inflating bubbles) remain out of reach; it may be interesting to introduce a limited form of divergence in the form of normal motion, via an operator splitting approach. The assumption of equal densities on either side of the sheet removes the baroclinic terms, which play a crucial role in previous vortex sheet smoke methods as well as potential extensions to small-scale liquid droplets.

I believe this surface-only discretization based on circulation is attractive both for its minimalist representation of complex soap film dynamics, and for the concrete numerical and computational benefits it offers. We have shown that it preserves volume effectively and exhibits minimal dissipation, while being fast and stable enough to handle a range of compelling foam and bubble behaviors. Having so far addressed the dynamics of both turbulent smoke and soap foams, we anticipate that vortex sheet techniques have a great deal yet to offer.

Part III

Surface-Only Liquids

Chapter 20

Introduction

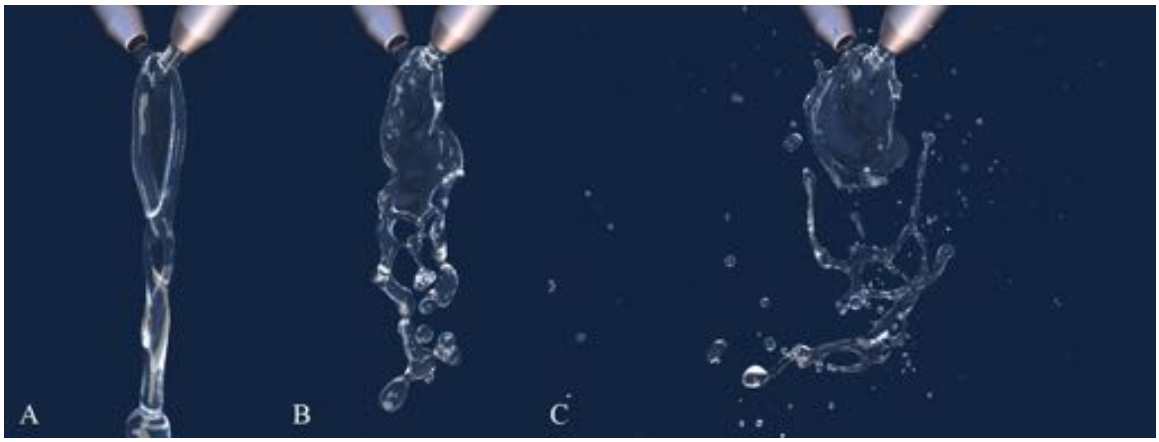


Figure 20.1: Reproduction of various patterns resulting from the collision of jets. (A) Fluid chains. (B) Disintegrating sheets. (C) Violent flapping.

Many fascinating liquid phenomena, such as the crown splash adored by artists and scientists alike and the fluid chains formed by colliding jets [Bush and Hasha, 2004], are driven by subtle balances between the surface tension force and the inertia of the liquid. The deforming free surface, a defining aspect of liquids, carries the geometric information that drives surface tension effects, which are in turn responsible for the characteristic look and feel of familiar liquids like water. Undoubtedly, capturing the free surface with sufficient detail is crucial to the successful re-creation of these liquid phenomena in computer simulations. From a computer graphics viewpoint, the free surface is also

the only visually important element of the final imagery, as it is where the most prominent optical effects such as reflection and refraction happen.

However, traditional fluid simulation techniques, including spatial grids [Foster and Fedkiw, 2001], volumetric meshes [Misztal *et al.*, 2010], and particles [Ihmsen *et al.*, 2014b], invest many of their degrees of freedom deep inside the liquid and far from the free surface. Populating the entire *volume* of the liquid is clearly uneconomical, and as we will demonstrate, it is also unnecessary for many phenomena of interest. In a traditional free surface flow solver, volumetric samples serve two roles: (a) to represent an arbitrary velocity field inside the liquid volume, and (b) to enforce incompressibility. Recent numerical experiments suggest that internal vorticity plays only a small role in the perceived dynamics of liquids [Zhang *et al.*, 2015], motivating us to disregard the first role of volumetric samples. But what about incompressibility? Discarding internal samples and working with variables only on the liquid surface, incompressibility can be enforced by projecting the velocity or position of the surface [Zhang *et al.*, 2012], or by working with a surface velocity representation that is, by construction, incompressible [Brochu *et al.*, 2012; Keeler and Bridson, 2014]. Motivated by these observations, we propose the **first surface-based treatment of general 3D liquid bodies dominated by surface tension and inertia**.

The first challenge in a surface-only numerical scheme is finding a sufficiently expressive velocity field representation that can be stored on the surface. Capturing an arbitrary three dimensional velocity field with a surface-only representation is an inherently ill-posed problem. Fortunately, many liquid animation scenarios are well approximated by a reduced space of *harmonic velocity fields* induced by two simplifying assumptions.

First, liquids are typically assumed to be incompressible, which reduces the space of velocity fields to those that are *divergence-free*.

Second, the three primary sources of vorticity in typical flows of interest are baroclinity due to density gradients, surface tension due to curvature gradients, and interaction with solid boundaries due to viscosity. However, for a liquid with uniform internal density, density gradients and surface tension forces occur only at the liquid-air surface. Furthermore, for the idealized case of inviscid flow described by the Euler equations, viscosity is absent and as a result vorticity generated at the solid or air surfaces *does not propagate into the interior* [Stock, 2006]. This motivates the assumption that the interior of the liquid volume is irrotational, reducing the space of velocity fields to those that are

curl-free.

Our work builds on these two assumptions. We develop a surface-based numerical treatment of volumetric liquids, inspired by the representations that become possible when an internal velocity field is assumed to be divergence- and curl-free. For a contractible domain, such as a liquid droplet, the *harmonic vector field* can be expressed as the gradient of a harmonic scalar function. When this integrability condition holds, it becomes possible to express the volumetric velocity field, and in turn to integrate the Euler equations, in terms of only surface position and velocity. Even when we consider scenarios beyond the scope of the integrability condition, the surface-based representation continues to produce visually compelling animations reproducing fascinating liquid behavior (see Figure 20.1).

This lower dimensional subspace of vector fields has previously been shown to possess a surprising degree of expressive power through successful applications to a variety of phenomena ranging from droplet impact [Davidson, 2000] and smoke [Brochu *et al.*, 2012; Weissmann and Pinkall, 2010], to soap films (presented in Part II of this thesis) and ocean waves [Xue *et al.*, 2001; Keeler and Bridson, 2014]. Unfortunately, existing surface-only techniques are not suited to reliable simulation of many common liquid phenomena, such as splashing and water jets. Vortex methods have difficulty treating baroclinity robustly (the Boussinesq approximation is not applicable to the liquid-air interface; see section 21.3 for more details), while potential flow approaches suffer from stability issues due to the nonlinearity of the Bernoulli equation. For most liquid phenomena, a full 3D simulation (grid-based, volumetric mesh-based, or particle-based) is currently the only feasible approach.

We propose a novel framework that is exempt from these pitfalls, for robust and flexible simulation of incompressible liquid of uniform density, with surface tension, gravity, and contact with solid objects. By making velocity the simulation state variable, we avoid the destabilizing effect of nonlinear time integration: the nonlinearity in the advective term of the Euler equations is easily handled on a Lagrangian mesh in the form of the total derivative.

Working directly with velocity in a surface-only setting requires a new set of tools. We present the first advection-projection scheme for surface-based liquids, which proceeds in three steps: a Lagrangian mesh advection phase; a surface-only projection phase based on the Helmholtz decomposition that requires only the evaluation of two boundary integrals; and a final boundary element solve

to integrate forces such as surface tension, gravity, and solid contact.

The use of an explicit triangle mesh for surface tracking allows for easy and accurate surface tension force computation, which is especially crucial at the three-phase junction where the solid, liquid, and air phases meet. We solve for the pressure and its normal derivative on the surface using a first-order boundary element method (BEM) solver, where the surface tension force and the solid contact force are conveniently described by Dirichlet and Neumann boundary conditions, respectively. The resulting numerical scheme successfully reproduces a host of surface tension-dominated liquid phenomena (see Figure 20.2 for an example).

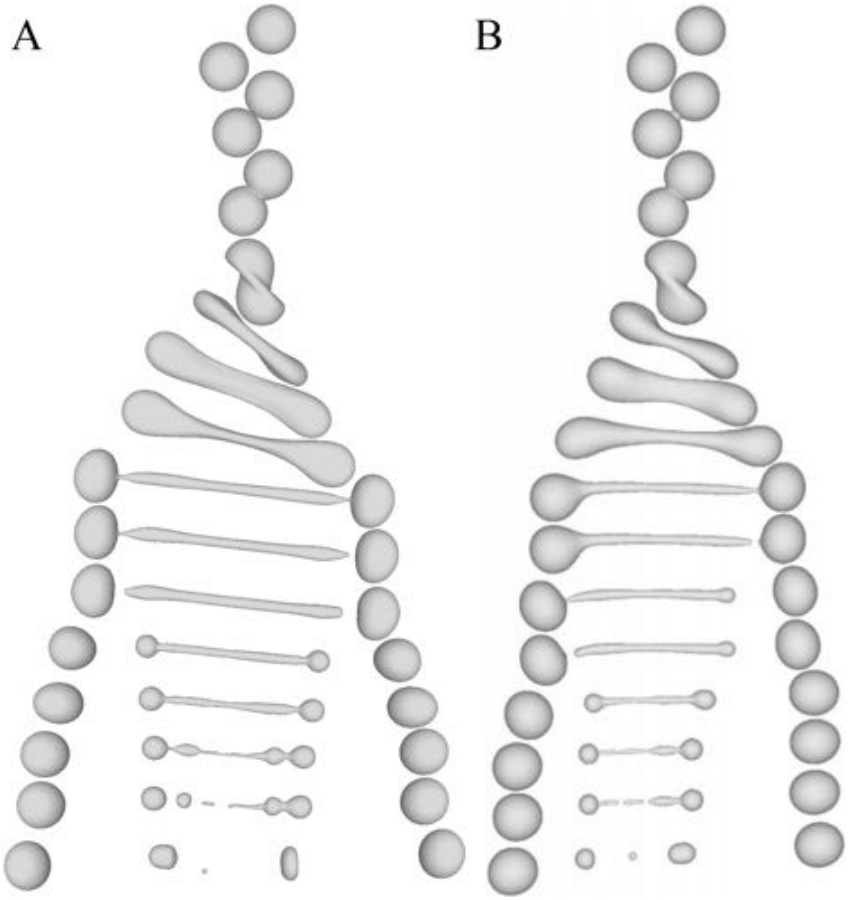


Figure 20.2: Comparison of the evolution of droplets in an off-center coalescence test between our method (A) and an unstructured tetrahedral mesh method (B) [Quan *et al.*, 2009].

Chapter 21

Related Work

Free surface flow simulation has been a central topic in the broader literature of computational fluid dynamics since the field's inception. Characterized by a free surface separating a liquid region from an exterior air region assumed to be massless, it presents various challenges in interface tracking, discretization of the governing physical equations, and treatment of liquid-solid-air interaction.

21.1 Grid and volumetric meshes

The dominant approach to liquid simulation discretizes the space in and around the liquid into a grid, and stores on it the velocity field of the fluid [Foster and Metaxas, 1996; Foster and Fedkiw, 2001]. The surface evolution is tracked using one of a range of techniques, such as the popular level set method [Osher and Sethian, 1988; Enright *et al.*, 2002b]. Alternative discretizations include conforming volumetric meshes [Chentanez *et al.*, 2007; Misztal *et al.*, 2010; Clausen *et al.*, 2013], embedded volumetric meshes [Batty *et al.*, 2010], and particle methods [Ihmsen *et al.*, 2014b]. The spatial discretization of the liquid volume is subsequently used to approximate and solve the incompressible Euler or Navier-Stokes equations.

These approaches suffer from memory and computational costs proportional to the volume of the simulation domain due to the discretization of the liquid interior. Furthermore, conforming volumetric mesh methods such as that of Clausen et al. [2013] spend a significant fraction of their computation time maintaining the quality of the mesh interior, while grid-based methods require more complex construction of differential operators to treat surface tension [Fedkiw *et al.*, 1999;

Hong and Kim, 2005] and triple junctions [Wang *et al.*, 2005] in the absence of a conforming mesh.

21.2 Mesh-based surface tracking for Eulerian fluids

In contrast to level set- or particle-based approaches for representing liquid geometry, explicit surface meshes [Wojtan *et al.*, 2011] store geometric information strictly on the surface and are gaining attention due to their ability to preserve surface details and volume. Their explicit geometry can be beneficial when coupled to an underlying volumetric discretization of the fluid dynamics, such as for the discretization of surface tension. Thürey et al. [2010] achieve a remarkable level of detail by combining a high resolution surface mesh for surface tracking and local wave simulation with a low resolution grid for coarse-scale motion and incompressibility. Brochu et al. [2010] also employ a surface mesh, using it to guide the placement of additional pressure samples in a Voronoi mesh. Bojsen-Hansen and Wojtan [2013b] introduce a surface tracking error measure which compares the normal from the mesh to the pressure gradient from the grid, enabling the use of meshes with much higher resolution than the grid. These works and others that utilize surface meshes, such as those by Schroeder et al. [2012] and Pfaff et al. [2012], all rely on a background grid to carry out the pressure solve. While they benefit from a higher resolution or the more straightforward surface description afforded by the explicit surface mesh, the physical degrees of freedom still reside on the grid and a discretization of the volume remains necessary.

21.3 Vortex sheets

While both grid- and mesh-based volumetric discretizations grow cubically in complexity when refined, surface meshes grow quadratically. Some researchers have therefore abandoned the background grid altogether and sought to formulate the necessary dynamics entirely on the surface. Brochu et al. [2012] advocate a philosophy of "as much as one sees, that much one should compute," achieving linear time complexity for smoke by using vortex sheets represented by an explicit surface mesh. Part II of this thesis proposes a circulation-preserving surface tension model for soap films also based on vortex sheets. The strength of vortex sheets lies in the Biot-Savart integral which recovers the full 3D velocity field from a surface-only representation; however, baroclinity, which is key to modeling the effects of gravity, is difficult to treat near strong density gradients. Brochu et

al. [2012] adopt the Boussinesq approximation, applicable only when the density difference across the surface is small (Atwood ratio approaches 0), while the soap films technique in this thesis does not consider baroclinity. Neither method is suitable for the simulation of sharp liquid-air interfaces in free surface flow where the density ratio is near-infinite (Atwood ratio approaches 1).

The fundamental limitation of vortex sheets is also its greatest strength: the reliance on Kelvin's circulation theorem to simplify self-advection. With this simplification the vortex sheets method completely side-steps the non-linear part of the dynamics and achieves both simplicity and efficiency; however, barotropy is a prerequisite condition to Kelvin's circulation theorem, which does not hold for phenomena containing interfaces with large density jumps, such as the liquid-air interface.

21.4 Potential flow

The simulation of liquid using potential flow formulations has been explored for various phenomena such as droplet impact [Davidson, 2000] and ocean waves [Xue *et al.*, 2001; Keeler and Bridson, 2014]. By assuming the velocity field is a gradient field and thus irrotational in addition to incompressible, the velocity field can be compactly encoded by the scalar potential function on the surface whose evolution is governed by the Bernoulli equation. Unfortunately, the nonlinear term in the Bernoulli equation [Davidson, 2000] destabilizes the time integration. Xue et al. [2001] rely on smoothing to suppress the instability, while Keeler and Bridson [2014] linearize the Bernoulli equation, sacrificing interesting motions such as breaking waves in return for stability. For the phenomena we consider, both a severe lack of stability and a loss of motion modes are undesirable.

21.5 Other non-volumetric techniques

Several additional techniques have been proposed to simulate liquid phenomena without paying the price of volumetric discretization. Brochu [2006] proposes an incompressibility enforcement technique based on a harmonic partition of the liquid domain. Zhang et al. [2012] efficiently process the motion of a surface mesh using a collection of deformation operators, including a mean curvature flow operator to approximate surface tension. In these works, the masses associated to the surface degrees of freedom only approximately reflect how much of the underlying liquid volume it influences. Batty et al. [2012] and Zhu et al. [2014] also utilize triangle meshes, but only to capture

Method	Setting	BI	BEM
[Zhang *et al.*, 2012]	Droplets		
[Brochu *et al.*, 2012]	Smoke	✓	
[Keeler and Bridson, 2014]	Waves	✓	✓
[Da *et al.*, 2015]	Bubbles	✓	
Current	3D Liquid	✓	✓

Table 21.1: A summary of related surface-based fluid methods. BI indicates boundary integrals; BEM indicates the boundary element method.

thin sheet-like structures, not as a representation of the surface of a liquid volume. For bulk bodies of liquid that are not thin in any direction, Zhu et al. [2014] use a volumetric mesh, effectively reverting to a conforming tetrahedral method.

21.6 Boundary integrals and the boundary element method

The boundary element method (BEM) has proven to be a powerful technique for a variety of phenomena including elasticity [James and Pai, 1999] and brittle fracture [Zhu *et al.*, 2015; Hahn and Wojtan, 2015], as well as the potential flow schemes above. We refer the interested readers to Sauter and Schwab [2011] for more details on the theory of boundary integrals and the boundary element method.

Despite the success of BEM in elasticity and fracture simulation, applying BEM to the Navier-Stokes equations results in an integral equation containing a domain integral [Ladyzhenskaya, 1963], corresponding to the nonlinear convective term. One proposed treatment is the Dual Reciprocity Method (DRM) [Power and Partridge, 1994; Florez and Power, 2001], which essentially approximates this term using a collection of radial basis functions. However, several issues of DRM have prevented it from gaining widespread popularity, including lack of convergence [Florez and Power, 2001] and ill-conditioned systems [Chen *et al.*, 2003].

We summarize the relationships among the surface-based fluid animation methods most relevant to the current work in Table 21.1. Note that the vortex sheets method is considered a boundary integral method, as is evident from the Biot-Savart integral.

Chapter 22

Time Integration

Our model of a 3D liquid volume represents the state—position and velocity—only on the liquid boundary surface. We do not explicitly represent state in the interior.

Throughout our method, we adopt the following assumption (which will be referenced as [HVF] in the following text whenever the derivation invokes it):

DEFINITION **22.0.1.** *The* Harmonic Velocity Field *assumption (HVF): A velocity field* $\mathbf{u} : \mathbb{R}^3 \to \mathbb{R}^3$ *fulfills the HVF assumption if:*

- $\nabla \cdot \mathbf{u} = 0$,
- $\nabla \times \mathbf{u} = 0$,

i.e., it corresponds to a flow that is both incompressible and irrotational.

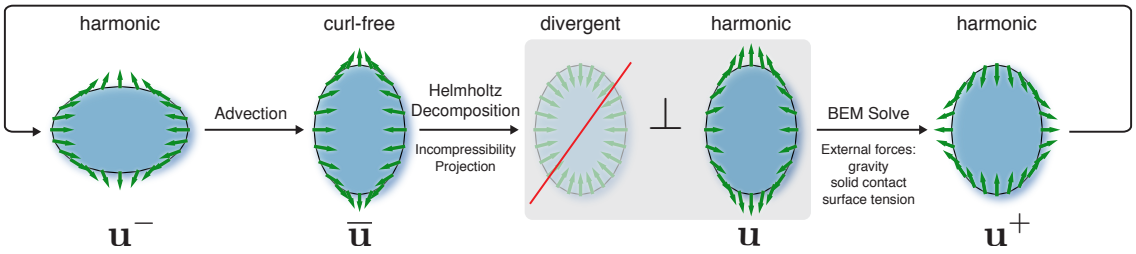


Figure 22.1: Overview of a time step. The inset text summarizes the properties on various velocity components.

We integrate forward in time by operating directly on the boundary surface, storing and updating the position and velocity of the surface. We employ operator-splitting [Stam, 1999] to solve the incompressible Euler equations over the liquid domain Ω:

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{F}, \tag{22.1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{22.2}$$

We implement this scheme in three steps (Figure 22.1): advection, projection, and integration of external forces (gravity, surface tension and solid contact). The advection step moves the position of the boundary surface and updates the velocity field absent internal and external forces ($\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = 0$). Next, the projection step removes any velocity divergence, $\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\nabla p$, introduced by advection. Finally, we integrate the effect of external forces into the velocity field while respecting the solid boundary and incompressibility constraints, $\frac{\partial \mathbf{u}}{\partial t} = \frac{1}{\rho}\mathbf{F}$. We describe each step in detail in the sections below.

22.1 Advection

At the start of time integration, we are given the position of the surface, decorated with a harmonic velocity field. In our Lagrangian approach, advection requires only to update the position of the surface; the velocity associated to each surface point is unaltered. Although the initial velocity field was divergence-free [HVF], the change in surface position can introduce velocity divergence.

22.2 Projection

Next, we remove the divergence, projecting back to the space of harmonic velocity fields. We first describe Helmholtz-based projection for a volumetric velocity field, and then present the transformation to the surface-based representation.

Helmholtz decomposition As depicted in Figure 22.1, we denote by $\mathbf{u}^-$ the harmonic 3D velocity field before advection, $\overline{\mathbf{u}}$ the (generally divergent) velocity field after passively moving the velocity field with the liquid, and $\mathbf{u}$ the (incompressible) post-advection, post-projection velocity field.

We can think of divergence as inducing a pressure, p, in a projection that brings $\overline{\mathbf{u}}$ back into the linear subspace of incompressible velocity fields to obtain $\mathbf{u}$. In other words, we can find $\mathbf{u}$ by the orthogonality condition $\mathbf{u} \perp (\overline{\mathbf{u}} - \mathbf{u})$ and the incompressibility condition $\nabla \cdot \mathbf{u} = 0$. For continuous vector fields in $\mathbb{R}^3$ over the liquid domain Ω, orthogonality $\mathbf{v} \perp \mathbf{w}$ is defined through the vanishing of the inner product $\langle \mathbf{v}, \mathbf{w} \rangle_\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{w} \mathrm{d}V$.

Unlike most previous approaches [Batty *et al.*, 2007; Clausen *et al.*, 2013; Misztal *et al.*, 2010; Ihmsen *et al.*, 2014a], which use a Poisson solve to perform the projection, we adopt a constructive Helmholtz decomposition [Phillips, 1933] on $\overline{\mathbf{u}}$,

$$\overline{\mathbf{u}} = -\nabla\Phi + \nabla \times \mathbf{A} \tag{22.3}$$

$$= \overbrace{\nabla \times \mathbf{A}_\Gamma - \nabla\Phi_\Gamma}^{\text{harmonic}} + \overbrace{\nabla\Phi_\Omega}^{\text{curl-free}} - \overbrace{\cancel{\nabla \times \mathbf{A}_\Omega}}^{\text{div-free}}, \tag{22.4}$$

$$\text{harmonic} = \mathbf{u}, \quad \text{curl-free} = \overline{\mathbf{u}} - \mathbf{u}, \quad \text{div-free} = \nabla \times \mathbf{A}_\Omega \ (\text{since } \nabla \times \overline{\mathbf{u}} = 0)$$

where the scalar potential is $\Phi(\mathbf{x}) =$

$$\underbrace{\int_\Gamma \mathbf{n}(\mathbf{y}) \cdot \overline{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) \mathrm{d}S_\mathbf{y}}_{\Phi_\Gamma} - \underbrace{\int_\Omega \nabla_\mathbf{y} \cdot \overline{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) \mathrm{d}V_\mathbf{y}}_{\Phi_\Omega}, \tag{22.5}$$

and the vector potential is $\mathbf{A}(\mathbf{x}) =$

$$\underbrace{\int_\Gamma \mathbf{n}(\mathbf{y}) \times \overline{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) \mathrm{d}S_\mathbf{y}}_{\mathbf{A}_\Gamma} - \underbrace{\int_\Omega \nabla_\mathbf{y} \times \overline{\mathbf{u}}(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) \mathrm{d}V_\mathbf{y}}_{\mathbf{A}_\Omega = 0}. \tag{22.6}$$

Here $\Gamma = \partial\Omega$ is the liquid surface, $\mathbf{n}$ is the outward unit surface normal, and $G(\mathbf{x}, \mathbf{y}) = -\frac{1}{4\pi\|\mathbf{x}-\mathbf{y}\|}$ is the free space Green's function of Laplace's equation. These formulas can be proven using the property of the Green's function $\nabla^2 G = \delta_D$, where δ_D is the Dirac delta. In the literature, boundary integrals in the form of Φ_Γ or $\mathbf{A}_\Gamma$ are known as *single layer potentials* [Sauter and Schwab, 2011].

Equation 22.4 decomposes $\overline{\mathbf{u}}$ into three *orthogonal* subspaces (see Appendix): (a) harmonic (simultaneously div- and curl-free); (b) divergent but curl-free; (c) rotational but div-free. Since by assumption $\overline{\mathbf{u}}$ is irrotational, the third term $\nabla \times \mathbf{A}_\Omega$ vanishes. Any divergent component of $\overline{\mathbf{u}}$ *must* be in $\nabla\Phi_\Omega$. These properties on the various velocity components are summarized in Table 22.1. If we equate $\mathbf{u}$ to the harmonic first component, then the divergent second component is $\overline{\mathbf{u}} - \mathbf{u}$. We have identified the solution to the simultaneous conditions $\nabla \cdot \mathbf{u} = 0$, from (a), and $\mathbf{u} \perp (\overline{\mathbf{u}} - \mathbf{u})$, by the orthogonality of the subspaces.

Component	Divergence-free	Curl-free	Zero
$\mathbf{u}^-$	✓	✓	
$\overline{\mathbf{u}}$		✓	
$\mathbf{u}$	✓	✓	
$\mathbf{u}^+$	✓	✓	
$\nabla\Phi_\Gamma$	✓	✓	
$\nabla\Phi_\Omega$		✓	
$\nabla\Phi$		✓	
$\nabla \times \mathbf{A}_\Gamma$	✓	✓	
$\nabla \times \mathbf{A}_\Omega$	✓	✓	✓
$\nabla \times \mathbf{A}$	✓	✓	

Table 22.1: A summary of various velocity components.

In summary, we can construct the divergence-free velocity, $\mathbf{u}$, using only the boundary integrals Φ_Γ and $\mathbf{A}_\Gamma$. Note that the Green's function, $G(\mathbf{x}, \mathbf{y})$, tends to infinity as the evaluation point, $\mathbf{x}$, approaches the source point, $\mathbf{y}$. The resulting singularities in the integrals require special treatment, which we describe in Section 23.

Surface-only representation Assume that the liquid domain Ω is contractible. By the Poincaré lemma, a curl-free vector field $\mathbf{u}$ over a contractible domain Ω can be always be represented as the gradient $\mathbf{u} = \nabla\phi$ of a scalar potential field, ϕ. If $\mathbf{u}$ is also divergence-free, $0 = \nabla \cdot \mathbf{u} = \nabla \cdot \nabla\phi$, then ϕ is harmonic, satisfying Laplace's equation $\nabla^2\phi = 0$.

Since our velocity field is harmonic, i.e., curl- and divergence-free [HVF], it can be exactly represented by the gradient of a harmonic potential.

Taking the boundary restriction of the velocity field yields a volume-conserving surface velocity field, whose normal component $\hat{\mathbf{n}} \cdot \mathbf{u}$ equals the normal derivative $(\hat{\mathbf{n}} \cdot \nabla)\phi$ of the harmonic potential by construction. Therefore, we have a specific instance of Laplace's equation, a boundary value problem (BVP), which can be solved to recover ϕ over Ω, using the surface velocity as the Neumann boundary condition. The gradient $\nabla\phi$ of the solution is unique.

If any two harmonic velocity fields have the same normal component in their boundary restriction, then they will both establish the same boundary data for the BVP, and both yield the same gradient $\nabla\phi$. Therefore, two harmonic velocity fields with the same normal component in their boundary restriction must be the same field. In other words, for a contractible domain, a (3D) harmonic velocity

field can be fully described by the normal component of its boundary restriction.

In addition to the construction above that recovers the harmonic volumetric velocity field from the given surface velocity field which is known to be the boundary restriction of some harmonic velocity field, we may also desire the ability to test if a given (arbitrary) surface velocity field is the boundary restriction of any harmonic velocity field at all. For this purpose we introduce the construction of a surface scalar field $\tilde{\zeta}$ for a given surface velocity field $\tilde{\mathbf{u}}$ (in this section we use tilde over symbols to denote surface fields as opposed to volumetric fields) provided certain conditions on $\tilde{\mathbf{u}}$ are met. We pick an arbitrary reference point $\mathbf{x}_0 \in \Gamma$, define $\zeta(\mathbf{x}_0) = 0$, and compute the potential of any other point $\mathbf{x} \in \Gamma$ as

$$\tilde{\zeta}(\mathbf{x}) = \int_{P_{\mathbf{x}_0\mathbf{x}}} \hat{\mathbf{t}} \cdot \tilde{\mathbf{u}} \mathrm{d}s, \tag{22.7}$$

where $P_{\mathbf{x}_0\mathbf{x}} \subset \Gamma$ is an arbitrary curve on the surface connecting $\mathbf{x}_0$ and $\mathbf{x}$, provided the choice of the curve does not affect this value; otherwise we say the $\tilde{\zeta}$ field is not well-defined for the given surface velocity $\tilde{\mathbf{u}}$. When $\tilde{\zeta}$ is indeed well-defined, we can use it as the Dirichlet boundary condition to solve a Laplace BVP problem for a volumetric field ζ over Ω. The aforementioned ϕ field over Ω can also be constructed provided the Neumann BVP problem is solvable (i.e. the Neumann boundary condition is compatible); otherwise we say the ϕ field is not well-defined for the given surface field $\tilde{\mathbf{u}}$. We then introduce the following theorem:

THEOREM **22.2.1.** *On a contractible domain, the necessary and sufficient condition for a surface velocity field $\tilde{\mathbf{u}}$ to be the boundary restriction of a harmonic velocity field is that the corresponding ζ and ϕ fields are both well-defined, and differ by a global constant.*

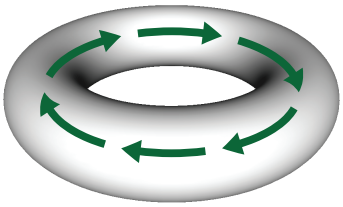
Proof. If $\tilde{\mathbf{u}}$ is indeed the boundary restriction of a harmonic velocity field $\mathbf{u}$, it clearly affords a well-defined ϕ field, as shown above, with the property $\mathbf{u} = \nabla\phi$ by construction. It follows that its boundary restriction, $\tilde{\phi}$, satisfies that $(I - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\mathbf{u} = \nabla\tilde{\phi}$.

For any closed surface loop $L \subset \Gamma$, the circulation is $\oint_L \hat{\mathbf{t}} \cdot \tilde{\mathbf{u}} \mathrm{d}s = \int_S \hat{\mathbf{n}} \cdot \nabla \times \tilde{\mathbf{u}} \mathrm{d}A$ where $S \subset \Gamma$ is the surface region enclosed by L (because the domain is assumed to be contractible, S is well-defined). The right hand side is the normal flux of vorticity through S. As the velocity field in the interior of Ω is curl free, the boundary limit of vorticity is zero (assuming the velocity is sufficiently smooth and well-behaved), so the circulation is zero for any L, and thus the choice of curve $P_{\mathbf{x}_0\mathbf{x}}$ does not affect the value of $\tilde{\zeta}(\mathbf{x})$. In other words, $\tilde{\zeta}$ is well defined, and so is ζ. By

construction $\tilde{\zeta}$ satisfies that $(I - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\mathbf{u} = \nabla\tilde{\zeta}$ as well. Therefore, $\tilde{\zeta}$ and $\tilde{\phi}$ differ only by a constant. Since both ϕ and ζ are harmonic functions over Ω and differ by a constant in the boundary value, they differ by the same constant over the entire Ω domain. In other words, the condition in question is a necessary condition.

On the other hand, if the ζ and ϕ fields corresponding to a given surface velocity field $\tilde{\mathbf{u}}$ are both well defined and differ by a constant, then $\mathbf{u} = \nabla\phi = \nabla\zeta$ is the desired velocity field. First, this velocity field $\mathbf{u}$ is harmonic by construction. Second, the boundary restriction of this velocity field $\mathbf{u}$ has the correct normal component because $\hat{\mathbf{n}} \cdot \mathbf{u} = \hat{\mathbf{n}} \cdot \nabla\phi = \hat{\mathbf{n}} \cdot \tilde{\mathbf{u}}$, as well as the correct tangential component because $(I - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\mathbf{u} = (I - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\nabla\zeta = (I - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\tilde{\mathbf{u}}$. Therefore, $\tilde{\mathbf{u}}$ is indeed the boundary restriction of $\mathbf{u}$. In other words, the condition in question is a sufficient condition. □

On a non-contractible domain, e.g., in 3D, for a genus $k \neq 0$ liquid body, the normal component of the boundary restriction of a harmonic velocity field still forms a compatible Neumann boundary condition, and the gradient of the BVP's solution is still unique. Furthermore, the velocity field obtained from the BVP is still harmonic by construction. However, the Poincaré lemma no longer holds on such a domain: there exists an "extra" *finite-dimensional vector space of harmonic vector fields* that cannot be expressed as gradients of continuous potentials. These correspond to a flow circulating around a finite number of non-contractible loops, e.g., around the handle of the solid donut (see incident figure). The analysis in our approach simply ignores these "extra" modes.



Surface-only projection We are now ready to see that the Helmholtz projection can be directly computed on the surface representation, taking as input and producing as output only surface data. Equation 22.4 allows us to evaluate the post-projection velocity

$$\mathbf{u} = \nabla \times \mathbf{A}_\Gamma - \nabla\Phi_\Gamma \tag{22.8}$$

at any location in Ω using only the surface velocity field on Γ, since both Φ_Γ and $\mathbf{A}_\Gamma$ are boundary integrals. The resulting $\mathbf{u}$ is a harmonic velocity field. Therefore, the boundary restriction of (22.4) allows us to operate directly on surface velocity fields, without representing the interior velocity field. Henceforth, we denote the boundary restriction of $\mathbf{u}$, $\overline{\mathbf{u}}$, etc. by the same symbols.

Tangential velocity Although (22.8) already gives us the post-projection velocity, we can simplify it further. For the scenarios we consider, air density is negligible compared to liquid density. Imagine that the pressure gradient ∇p has a tangential component: it would effect an infinite tangential air velocity, in turn instantly restoring equilibrium. This argument leads to the familiar free-surface condition $p = 0$, from which it follows that the pressure gradient in (22.1) is always normal to the surface Γ. We therefore resolve (22.8) using only the normal component of $\mathbf{u}$, denoted u^n, while copying the tangential component directly from $\overline{\mathbf{u}}$ using the tangential projection operator $P = I - \mathbf{n}\mathbf{n}^T$:

$$\mathbf{u} = P\overline{\mathbf{u}} + u^n\mathbf{n}, \tag{22.9}$$

$$u^n = \mathbf{n} \cdot (\nabla \times \mathbf{A}_\Gamma - \nabla\Phi_\Gamma)\,. \tag{22.10}$$

Apart from being more efficient, the main advantage of using the Helmholtz decomposition only for the normal component, u^n, is that it simplifies the treatment of the $\nabla \times \mathbf{A}_\Gamma$-term in (23.4)–(23.6).

22.3 External force integration

In the final stage of time integration, we consider external forces $\mathbf{F}$. The input to this final stage is the post-projection harmonic velocity field, and the output is the end of time step harmonic velocity field [HVF]. In particular, we emphasize that external forces are added in such a way that the velocity field remains incompressible, in contrast with typical volumetric approaches.

Unlike with volumetric discretizations such as a grid or a tetrahedral mesh, in our surface-only representation the mass associated with each degree of freedom cannot be easily computed; when a surface vertex moves it is not immediately clear how much liquid volume moves with it. We therefore circumvent the computation of the nodal masses altogether.

By harmonicity [HVF], the velocity change induced by external forces can be expressed as the gradient of a harmonic pressure field $p_\mathbf{F}$. Therefore, given appropriate boundary conditions, we can specify $p_\mathbf{F}$ as the solution to Laplace's equation.

We model external forces that act (a) normal to the boundary surface, e.g., surface tension, free-slip solid contact, or (b) uniformly throughout the liquid body, e.g., gravity. Both of these categories allow for straightforward modeling of boundary conditions for Laplace's equation.

The boundary consists of the liquid-air interface Γ_A and the liquid-solid interface Γ_S, satisfying $\Gamma_A \cup \Gamma_S = \Gamma$. The surfaces Γ_A and Γ_S intersect only along a curve T which we refer to as the triple junction. Neglecting air pressure, the surface tension force creates a Dirichlet boundary condition at the liquid-air interface. The solid contact force and gravity body force, on the other hand, create a Neumann boundary condition, i.e., a pressure derivative that prevents interpenetration and separation. Consequently, we can write a well-posed boundary value problem to find the pressure due to external forces p_F:

$$\nabla \cdot \nabla p_{\mathbf{F}} = 0\ , \quad \text{subject to} \tag{22.11}$$

$$p_{\mathbf{F}} = \sigma H, \qquad \text{on } \Gamma_A, \tag{22.12}$$

$$\frac{\partial p_{\mathbf{F}}}{\partial \mathbf{n}} = \frac{\rho}{\Delta t}(\mathbf{u} + \mathbf{g}\Delta t - \mathbf{u}_{\text{solid}}) \cdot \mathbf{n}, \qquad \text{on } \Gamma_S, \tag{22.13}$$

where σ is the surface tension coefficient, H is the (signed) mean curvature, and $\mathbf{g}$ is the gravitational acceleration. We explicitly integrate acceleration due to gravity into the vertex velocities; the constraint on the solid contact region ensures that these vertices cannot move into the solid.

A vertex is considered to be in contact with the solid only if at least one of its incident faces is in Γ_S. When a droplet peels off a solid surface, the contact area shrinks as the droplet moves away. Finally, Γ_S consists of only one triangle; once it gets too small, remeshing turns this triangle into a single vertex. This last vertex is then *not* constrained any further and can move away from the solid freely.

We solve this system using the Boundary Element Method [Sauter and Schwab, 2011], *without the need to discretize the liquid volume*. The BEM solve yields pressure values and normal derivatives. The resulting pressure gradient accelerates the input velocity field $\mathbf{u}$ to the end-of-step velocity $\mathbf{u}^+$,

$$\mathbf{u}^+ = \mathbf{u} - \frac{\Delta t}{\rho}\nabla p_{\mathbf{F}}. \tag{22.14}$$

Chapter 23

Spatial Discretization

23.1 Surface mesh

To track the liquid domain boundary Γ, we employ the mesh-based surface tracking library *Los Topos* [Da *et al.*, 2014b], taking advantage of its gentle mesh-merging operation, robust collision resolution, and remeshing capabilities. However, in principle any surface tracking technique that maintains a manifold surface mesh can be used. Γ is represented as a triangle mesh (V, F) consisting of vertices V and triangular faces F. Each connected component of liquid volume is represented by a closed manifold triangle mesh. We use $V(i)$ and $F(i)$ to denote the set of vertices neighboring vertex i and the set of faces incident to vertex i, respectively. The area of face i is denoted a_i, while the *vertex area* is defined as $\hat{a}_i = \frac{1}{3}\sum_{j \in F(i)} a_j$. We store a velocity vector $\mathbf{u}_i$ (the subscript distinguishes the discrete quantity $\mathbf{u}_i$ from its continuous counterpart $\mathbf{u}$) on each vertex $i \in V$. A linear basis function θ_i is used to interpolate the velocity at any point on the surface: $\mathbf{u} = \sum_{i \in V} \theta_i \mathbf{u}_i$.

We implement isotropic spatial adaptivity into the *Los Topos* library by allowing different elements to have different minimum and maximum edge length bounds. Each vertex is endowed with a *target edge length*, from which the its minimum and maximum edge lengths are calculated. The remeshing code makes edge collapsing and splitting decisions based on the smaller of the bounds on the two vertices of the edge. Loosely inspired by Narain et al. [2012], we first determine the vertex target edge lengths individually according to the local curvature (as the largest deviation of the dihedral angles from π on incident edges) and the local velocity variation (as the magnitude of the velocity Laplacian). We then reduce the target edge lengths where necessary, in order to ensure that

the target edge lengths on neighboring vertex do not differ by more than a constant factor (we use 1.2 for all tests).

The stored surface velocity is linearly interpolated onto new vertices during edge-split operations, and averaged onto the replacement vertex during edge-collapse operations.

23.2 Vertex-neighborhood integration

Using the piecewise-linear triangle element greatly simplifies the geometric evolution of the surface as compared to higher order representations, but makes evaluation of boundary integrals tricky. For example, the normal derivative $\mathbf{n} \cdot \nabla \Phi_\Gamma$ has a strongly singular kernel [Sauter and Schwab, 2011] and is not well defined at non-smooth points on the surface (such as a vertex) even in the Cauchy principal value sense. Appropriate measures must be taken to manage the order of singularity.

We discretize (22.10) by integrating it with the basis function θ_i, in a style similar to the Galerkin BEM:

$$\hat{a}_i u_i^n = \int_\Gamma \theta_i u^n \mathrm{d}S = \int_\Gamma \theta_i \left(\mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma - \mathbf{n} \cdot \nabla \Phi_\Gamma \right) \mathrm{d}S. \tag{23.1}$$

The integrals can then be evaluated on each face individually, over which the normal $\mathbf{n}$ is a constant:

$$\int_\Gamma \theta_i \mathbf{n} \cdot \nabla \Phi_\Gamma \mathrm{d}S = \sum_{j \in F(i)} \int_j \theta_i \mathbf{n} \cdot \nabla \Phi_\Gamma \mathrm{d}S, \tag{23.2}$$

$$\int_\Gamma \theta_i \mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma \mathrm{d}S = \sum_{j \in F(i)} \int_j \theta_i \mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma \mathrm{d}S. \tag{23.3}$$

The right hand sides of (23.2) and (23.3) are double integrals, since Φ_Γ and $\mathbf{A}_\Gamma$ are boundary integrals themselves. The inner integrals diverge to infinity near vertex i in a weakly singular fashion, thus we discretize the outer integral using a four-point Gaussian quadrature with Duffy transform [Duffy, 1982; Keeler and Bridson, 2014]; this analytically removes weak singularities.

Next, we describe the discretization of the inner integral. For the Φ_Γ term, the integrand in (23.2) requires evaluating $\mathbf{n} \cdot \nabla \Phi_\Gamma = \frac{\partial}{\partial \mathbf{n}} \Phi_\Gamma = \int \mathbf{n}(\mathbf{y}) \cdot \overline{\mathbf{u}}(\mathbf{y}) \frac{\partial}{\partial \mathbf{n}_\mathbf{x}} G(\mathbf{x}, \mathbf{y}) dS_\mathbf{y}$. Although it is divergent on vertices as discussed above, the integral is not singular when evaluated at a quadrature point in the interior of face j, because the kernel $\frac{\partial}{\partial \mathbf{n}_\mathbf{x}} G(\mathbf{x}, \mathbf{y})$ evaluates to zero on face j. This means that within a particular face, this face itself does not contribute to the value of the integral. Therefore,

this integral can be discretized by Gaussian quadrature directly. In practice we find using only one quadrature point per face sufficient.

The $\mathbf{A}_\Gamma$ term in (23.3), on the other hand, is vector valued and needs a different treatment. We first transform it via integration by parts:

$$\int_j \theta_i \mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma \mathrm{d}S \tag{23.4}$$

$$= \int_j \mathbf{n} \cdot \nabla \times (\theta_i \mathbf{A}_\Gamma) \mathrm{d}S - \int_j \mathbf{n} \cdot (\nabla \theta_i \times \mathbf{A}_\Gamma)\, \mathrm{d}S \tag{23.5}$$

$$= \oint_{\partial j} \theta_i \mathbf{A}_\Gamma \cdot \mathbf{t} \mathrm{d}s - \int_j \mathbf{n} \cdot (\nabla \theta_i \times \mathbf{A}_\Gamma)\, \mathrm{d}S, \tag{23.6}$$

where ∂j is the boundary of triangle j, and $\mathbf{t}$ is its unit tangent. Since the integration by parts moved the derivative from $\mathbf{A}_\Gamma$ to the shape function θ_i, the inner integrals now have a weakly singular kernel instead of a strongly singular one. For a given face $j \in F(j)$, the first term consists of three line integrals, one along each edge. The integrals along the two edges incident to vertex i cancel with those from neighboring faces, while the integral along the third edge is zero because θ_i vanishes there (Figure 23.1). Therefore, the first term can be dropped, resulting in:

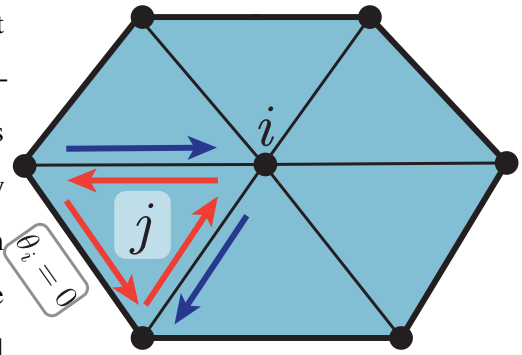


Figure 23.1: The loop integral involving $\mathbf{A}_\Gamma$.

$$\int_j \theta_i \mathbf{n} \cdot \nabla \times \mathbf{A}_\Gamma \mathrm{d}S = -\int_j \mathbf{n} \cdot (\nabla \theta_i \times \mathbf{A}_\Gamma)\, \mathrm{d}S. \tag{23.7}$$

As with the preceding weakly singular integrals, this too can be discretized directly by a four-point Gaussian quadrature with Duffy transform. Consequently, we now have all the required tools to evaluate the normal component of the velocity field in (23.1).

Eliminating numerical smoothing The Galerkin-style outer integral (23.1) is employed to suppress the singularity, but at the same time it introduces a smoothing of the velocity field which can damp the motion significantly at low mesh resolutions. Fortunately, as we show below, the negative effects of this numerical dissipation can be largely counteracted.

For any discrete field ϕ_i on vertices, whose interpolated continuous field is $\phi = \sum_{i \in V} \theta_i \phi_i$, discretizing it back to vertices again via an integral in the form of (23.1) will result in a discrete field $\tilde{\phi}_i$ that appears smoothed compared to ϕ_i:

$$\tilde{\phi}_i = \frac{1}{\hat{a}_i} \int_\Gamma \theta_i \phi \mathrm{d}S = \frac{1}{\hat{a}_i} \int_\Gamma \theta_i \sum_{j \in V} \theta_j \phi_j \mathrm{d}S \tag{23.8}$$

$$= \frac{1}{\hat{a}_i} \phi_i \int_\Gamma \theta_i^2 \mathrm{d}S + \frac{1}{\hat{a}_i} \sum_{i \neq j} \phi_j \int_\Gamma \theta_i \theta_j \mathrm{d}S \tag{23.9}$$

$$= \frac{1}{2} \phi_i + \frac{1}{2} \sum_{j \in V(i)} \omega_j \phi_j, \text{where} \tag{23.10}$$

$$\omega_j = \frac{\sum\limits_{k \in F(i) \cap F(j)} a_k}{2 \sum\limits_{k \in F(i)} a_k}. \tag{23.11}$$

The weights ω_j can be computed by evaluating the integrals over the linear basis functions θ on triangles containing node i and collecting the resulting terms. It is easy to verify that these weights form a partition of unity: each term in the sum in the nomiator adds the areas of the two triangles containing edge (i, j), the result is scaled by twice the area of the neighborhood of node i, as illustrated in Figure 23.1. Guided by this analysis, we apply Laplacian smoothing after evaluating the velocity field by (23.1) and before the pressure solve, with the negative coefficient $-\frac{1}{2}$ and the same weights as above, which effectively sharpens the velocity field just enough to remove the numerical smoothing. In practice we use a slightly smaller coefficient (e.g. -0.4 or -0.45) to retain a small amount of smoothing, so that the explicit time integration is stable.

23.3 Boundary Element solve

In order to solve (22.11) for pressure induced by external forces, we use a direct boundary integral equation formulation [Atkinson, 1997], due to the mixed nature of our boundary condition:

$$\Theta(\mathbf{x}) p_{\mathbf{F}}(\mathbf{x}) = \int_\Gamma \left(p_{\mathbf{F}}(\mathbf{y}) \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} - \frac{\partial p_{\mathbf{F}}(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} G(\mathbf{x}, \mathbf{y}) \right) \mathrm{d}S_\mathbf{y}, \tag{23.12}$$

where $\Theta(\mathbf{x})$ is the internal solid angle of the liquid domain Ω at $\mathbf{x} \in \Gamma$, discretized according to Mantic [1993] (below, we will drop the subscript from $p_\mathbf{F}$ in derivatives for legibility). We then use a vertex-collocated linear BEM formulation as in Keeler and Bridson [2014].

Special care must be taken at the triple junctions, where the surface Γ is expected to have a sharp crease. Such geometric non-smoothness often plagues collocated BEM solvers, as quantities like $\frac{\partial p}{\partial \mathbf{n}}$ may take on different values on either side of the crease, creating a discontinuity. We use a simplified version of the *double nodes* technique [Hartmann, 2012], taking advantage of the fact that the pressure p is always continuous, and that the $\frac{\partial p}{\partial \mathbf{n}}$ discontinuity only occurs at the triple junctions. The local geometry around a triple-junction vertex can be seen as the union of the liquid-air interface (part of Γ_A) and the liquid-solid interface (part of Γ_S), each a smooth patch of surface isomorphic to a half-plane. Therefore, each vertex on a triple junction holds two $\frac{\partial p}{\partial \mathbf{n}}$ values, one for the liquid-air interface, which is an unknown, and the other for the liquid-solid interface, which is given by the Neumann boundary condition in (22.13). The pressure p, on the other hand, is not duplicated, and its value on the triple junction is given by the Dirichlet boundary condition in (22.12). This guarantees that there is still one unknown and one collocation equation on each triple junction vertex, as for all other vertices, and thus the resulting linear system is not over- or underdetermined.

23.4 Pressure and discrete mean curvature

Equation 22.12 requires the evaluation of mean curvature H at a vertex. We discretize it as the (signed) scalar mean curvature integral over the vertex local neighborhood (the *discrete mean curvature measure*) [Cohen-Steiner and Morvan, 2003]. The integral mean curvature is then divided by the vertex area $\hat{a}_i$ to obtain the pointwise mean curvature H in (22.12). This discretization scheme has been found to work well for implementing surface tension as a pressure jump condition in the Lagrangian setting [Da *et al.*, 2015].

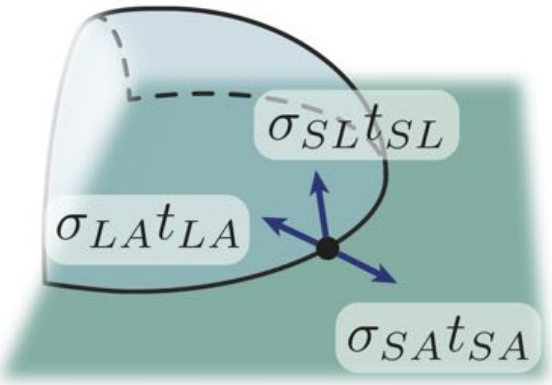


Figure 23.2: Surface tension force balance at the triple junction.

Again, the triple junctions must be treated differently, as one of the principal curvatures of the liquid surface is essentially infinite, making the mean curvature not well-defined. Examining the

force balance at a triple junction point (Figure 23.2), we find the pressure induced by the pairwise surface tension forces between the three phases as an integral over a strip of infinitesimal width covering both sides of the triple junction,

$$\int p_{\text{phases}} \mathrm{d}s = (\sigma_{LA}\mathbf{t}_{LA} + \sigma_{SA}\mathbf{t}_{SA} + \sigma_{SL}\mathbf{t}_{SL}) \cdot \mathbf{t}_{SA}, \tag{23.13}$$

where subscripts LA, SA and SL stand for liquid-air, solid-air and solid-liquid respectively, and the $\mathbf{t}$ vectors with various subscripts are the unit tangent vectors in the corresponding interface, pointing away from and orthogonal to the triple junction. The width of this strip over which the surface tension force between three phases is exerted is a numerical parameter, chosen so that the motion of non-equilibrium triple junction configurations can be properly resolved by the explicit time integration, while making the equilibrium configuration sufficiently accurate. We find a value of 0.25 times the average edge length works well.

23.5 Velocity update

However, the BEM solve for $p_{\mathbf{F}}$ yields pressure values and normal derivatives, not the gradients required by the update (22.14). For smooth regions of the surface, we interpolate the vertex pressure linearly per face to obtain a piecewise-constant tangential pressure gradient, and average them from faces to vertices. This tangential gradient is then combined with the normal pressure derivative $\frac{\partial p}{\partial \mathbf{n}}$ from the BEM solve to produce the full pressure gradient. On triple junctions, across which the surface normal and tangent vectors are different, we solve the following overdetermined system via its normal equations for each triple junction vertex i, essentially least-square fitting the various directional derivatives known to us:

$$\begin{pmatrix} \mathbf{n}_S^T \\ \mathbf{n}_A^T \\ \mathbf{t}_{SL}^T \\ \mathbf{t}_{LA}^T \\ \mathbf{t}_{TP}^T \end{pmatrix} \nabla p = \begin{pmatrix} \frac{\partial p}{\partial \mathbf{n}_S} \\ \frac{\partial p}{\partial \mathbf{n}_A} \\ \frac{\partial p}{\partial \mathbf{t}_{SL}} \\ \frac{\partial p}{\partial \mathbf{t}_{LA}} \\ \frac{\partial p}{\partial \mathbf{t}_{TP}} \end{pmatrix}, \tag{23.14}$$

where $\mathbf{t}_{TP}$ is the unit tangent of the triple junction curve, and the partial derivatives on the right hand side are obtained either from the BEM solve (for normal derivatives) or from tangentially

Vertices	Faces	FMM solve	Single iteration of FMM solve	Factor
162	320	128.4ms	1.42ms	-
642	1280	1838.9ms	36.86ms	24x
2562	5120	39.365s	403.3ms	11x
10242	20480	268.23s	1759ms	4.4x
40962	81920	1884.31s	8533ms	4.9x

Table 23.1: Time cost and mesh complexity for various refinement levels.

differentiating the pressure field (for tangential derivatives) as before. The first four equations of this overdetermined system establish the pressure gradient in the plane orthogonal to the triple junction. Although redundant, only together can they span this plane reliably regardless of the contact angle at the triple junction; removing any of them makes the system prone to ill-conditioning.

23.6 Fast summation for boundary integrals

Both the boundary integral evaluation in the projection step and the boundary element solve in the force integration step can be accelerated by fast summation techniques like FMM [Greengard and Rokhlin, 1987]. As in Part II, we employ FMMTL [Cecka and Layton, 2015], a template library that provides an efficient implementation of FMM for various kernels, for this task. Following Keeler and Bridson [2014], the BEM solve is performed by a BiCGStab solver that relies on the FMM summation for matrix-vector multiplication.

23.6.1 Scaling

Table 23.1 shows preliminary results on the scaling of the FMM implementation compared to the naïve direct summation implementation. The test geometry is a deformed spherical droplet with successively refined tessellation. The right most column shows the factor of cost increase from the previous row due to the mesh complexity increase; a factor of 4 is consistent with linear scaling. The factor is is significantly higher for the early rows (which is not surprising from the $O(N \log N)$ scaling), and approaches 4 for the later rows.

Chapter 24

Evaluation

This section evaluates the proposed method with the results of some simulation experiments. The purpose of these experiments is to explore what phenomena the method can produce, as well as to investigate how restrictive the HVF assumption is and how expressive the harmonic velocity fields can be.

24.1 Dripping

The familiar phenomenon of water dripping from a sink tap is driven by surface tension in a process known as the Rayleigh-Plateau instability (Figure 24.1). Our method successfully captures the entire process from the thinning of the neck to the pinch-off of the droplet.

24.2 Water jets collision

Bush and Hasha [2004] systematically studied the flow patterns resulting from laminar jets colliding at an angle, identifying several regimes in a phase diagram parameterized by the jet flow rate, surface tension coefficient, nozzle radius, etc. In Figure 20.1, the most representative patterns are reproduced. Lower flow rates and higher surface tension coefficients cause the liquid jets shooting away from the original collision to bend back and meet again for a second collision (and subsequently, a third, and so on). Each collision flips the orientation of the liquid sheet plane by 90°, forming a so-called "fluid chain" structure (Figure 20.1 A). At higher flow rates, the liquid sheet oscillates in the normal

direction until it disintegrates (Figure 20.1 B) or even ruptures violently (Figure 20.1 C), as a result of the liquid sheet flapping instability (highlighted in Figure 24.2).

24.3 Splash on a hydrophobic surface

The behavior of a liquid droplet on a solid surface is greatly affected by the relative strength of the surface tension at the liquid-solid interface and at the air-solid interface. Figure 24.3 shows the splash created by the impact of a water droplet on a hydrophobic solid surface.

24.4 Crown splash

When a droplet impacts on a water surface, the resulting splash takes a distinctive "crown" shape as a result of the Rayleigh-Plateau instability (Figure 24.4). The top of the crown usually atomizes into many small droplets shooting off in different directions. Level set based surface tracking methods are known to lose these small droplets quickly. Our method was able to resolve and maintain these droplets, recreating the splashes at a previously unseen level of detail.

24.5 Droplet collision

Figure 24.5 shows the splash patterns resulting from the head-on collision of two droplets in zero gravity. The extent of the splash and the atomization of the sheet is dictated by the relative strength of surface tension and liquid inertia, which is described by the dimensionless Weber number $We = \rho v^2 l/\sigma$. Our results qualitatively agree with the experimental observations reported by Pan et al. [2009], successfully reproducing the "fingering and separation" regime and the "breakup" regime (Figure 24.6) in the correct parameter ranges.

In applications like ink-jet printing, off-center collisions are of more practical interest since they are statistically more likely. Quan et al. [2009] studied the intermediate configurations during the coalescence and subsequent separation of two droplets using an unstructured tetrahedral mesh solver. Figure 20.2 shows a side-by-side comparison of our simulation and the results reported by Quan et al. [2009] in Figure 16.

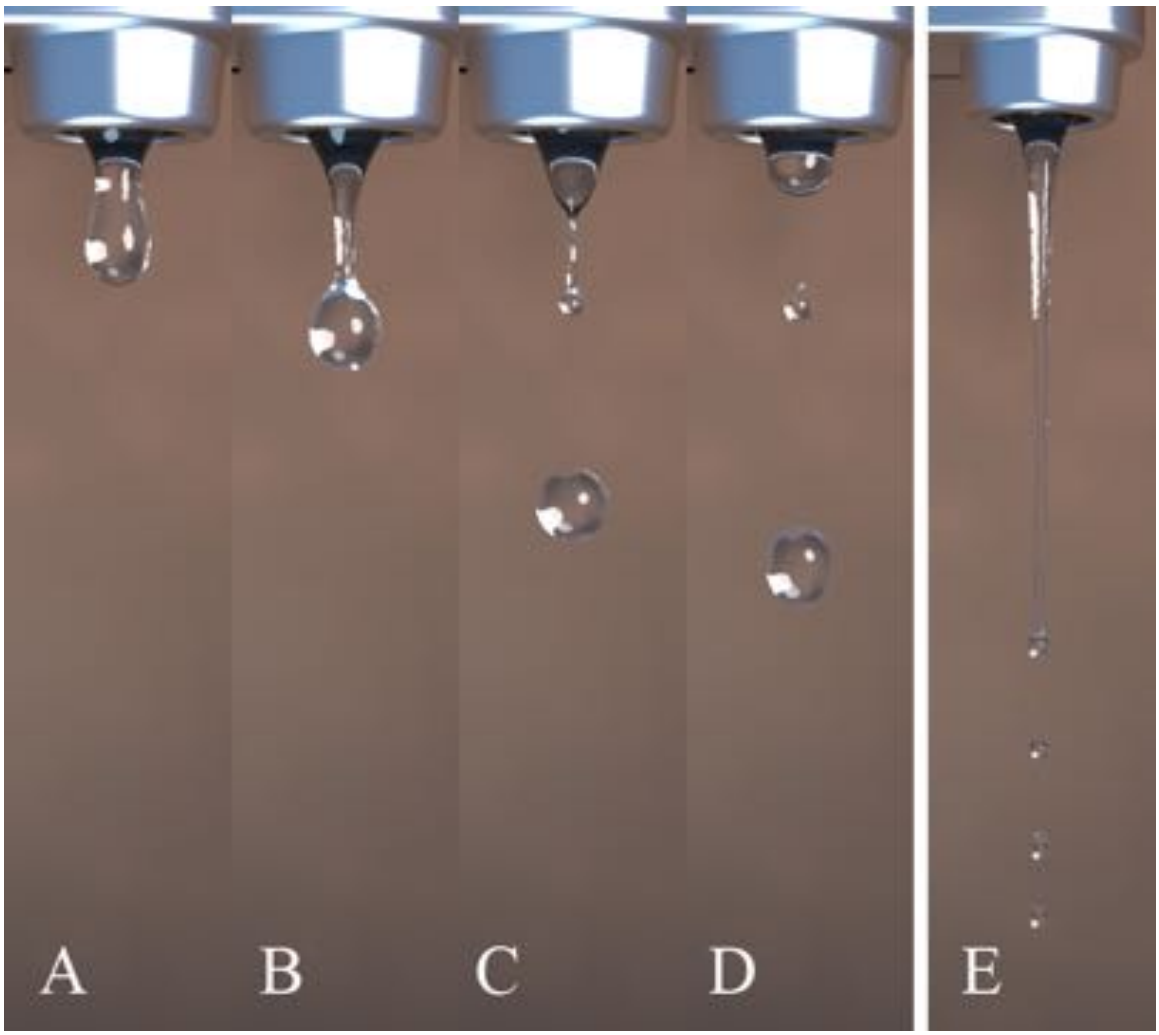


Figure 24.1: Dripping water. (A) – (D) A droplet pinches off near the tap at a low flow rate. (E) An elongated stream is created before droplet formation at a high flow rate.

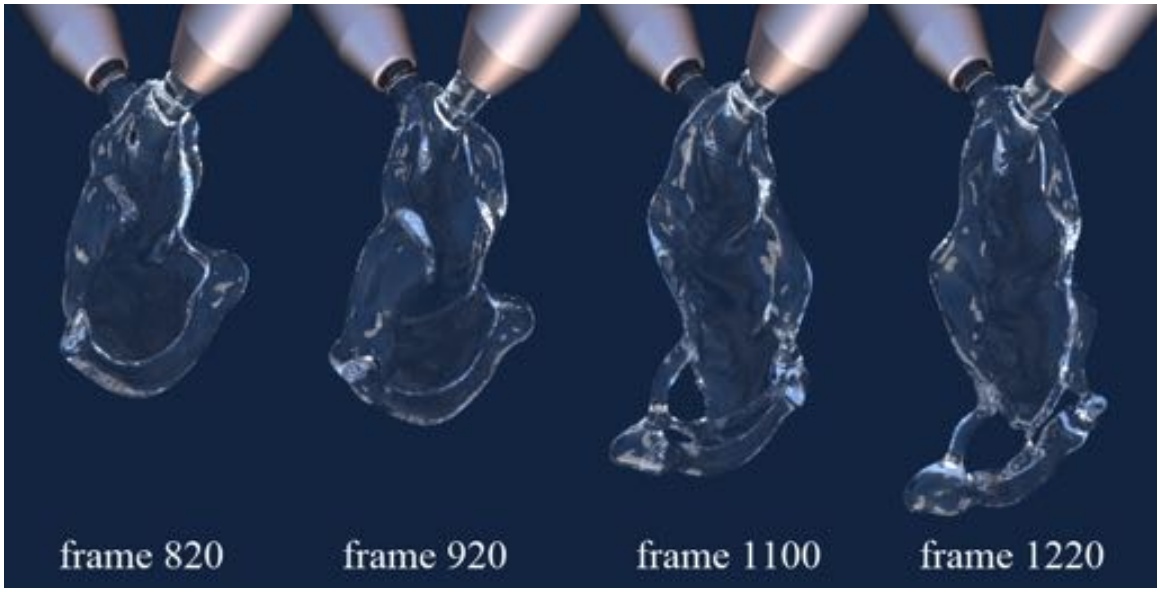


Figure 24.2: Sheet flapping instability following a jet collision.

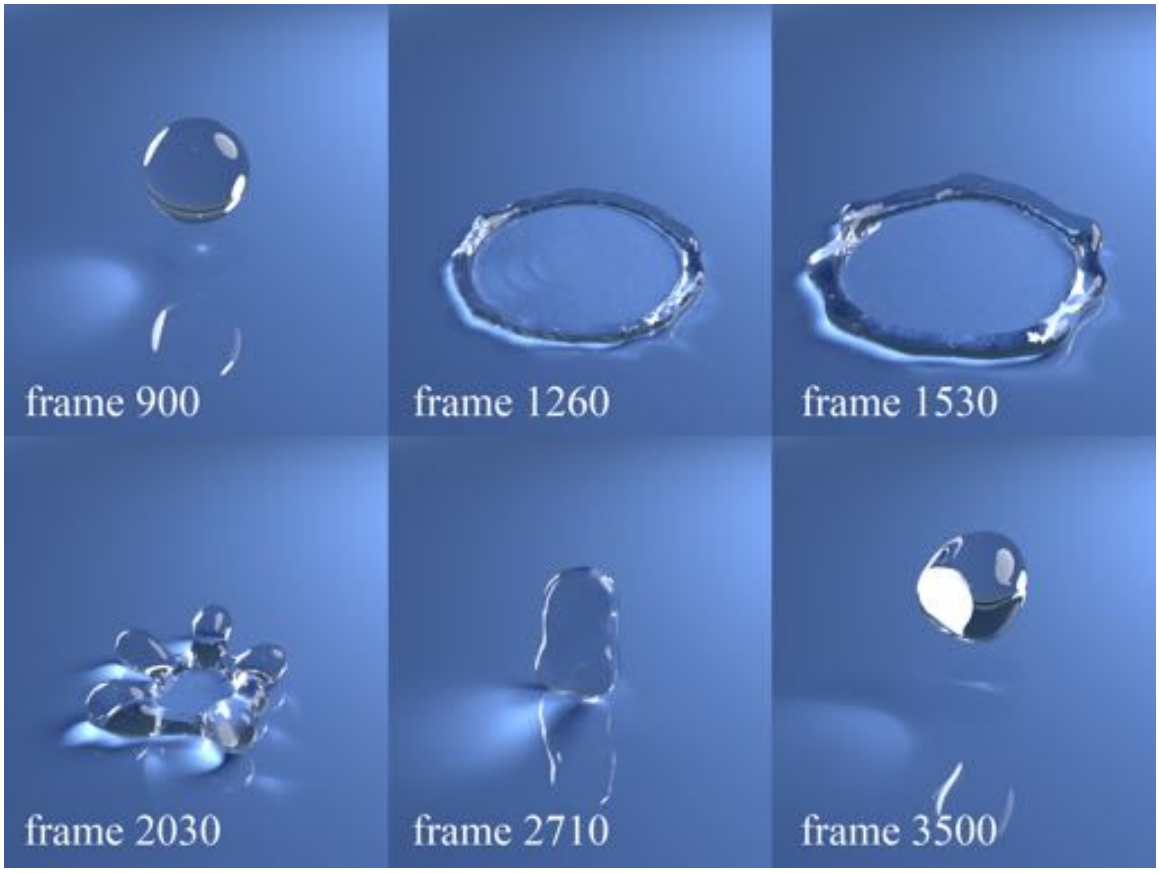


Figure 24.3: A water droplet impacting, splashing on, and rebounding off a hydrophobic surface.

Figure 24.4: Crown splash.

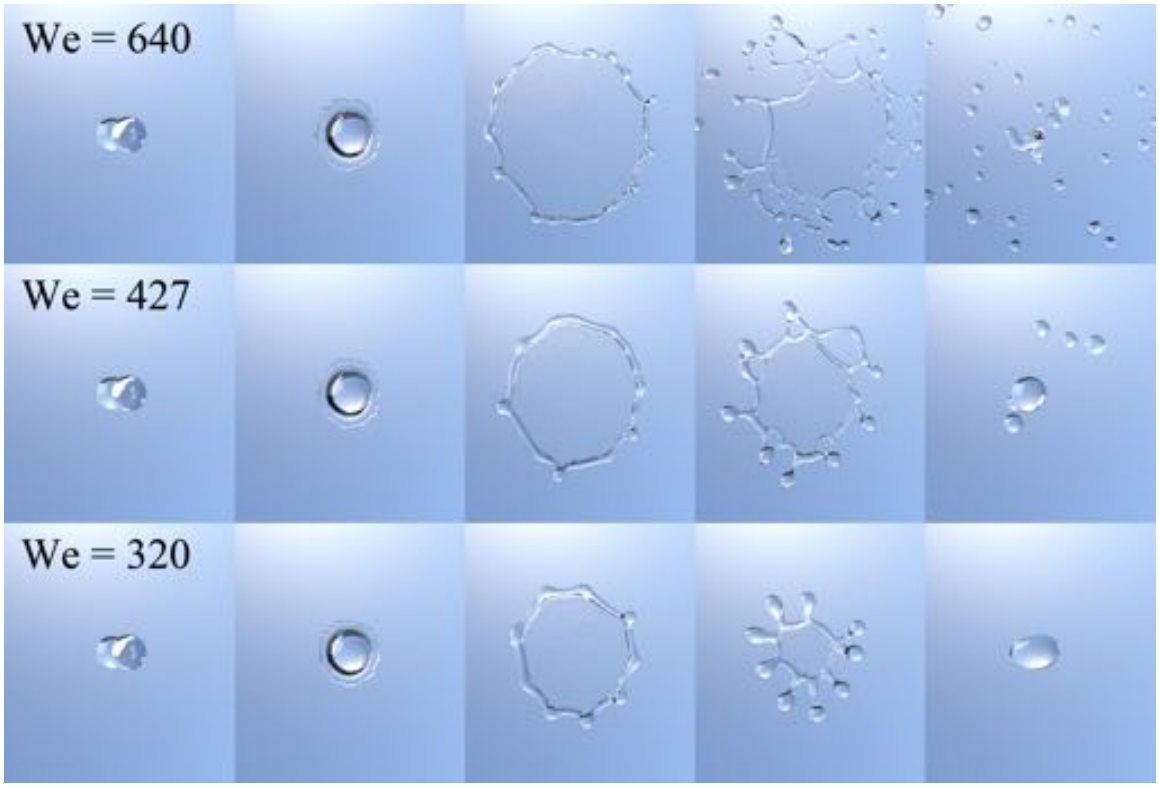


Figure 24.5: Collision of two droplets, with various surface tension coefficients.

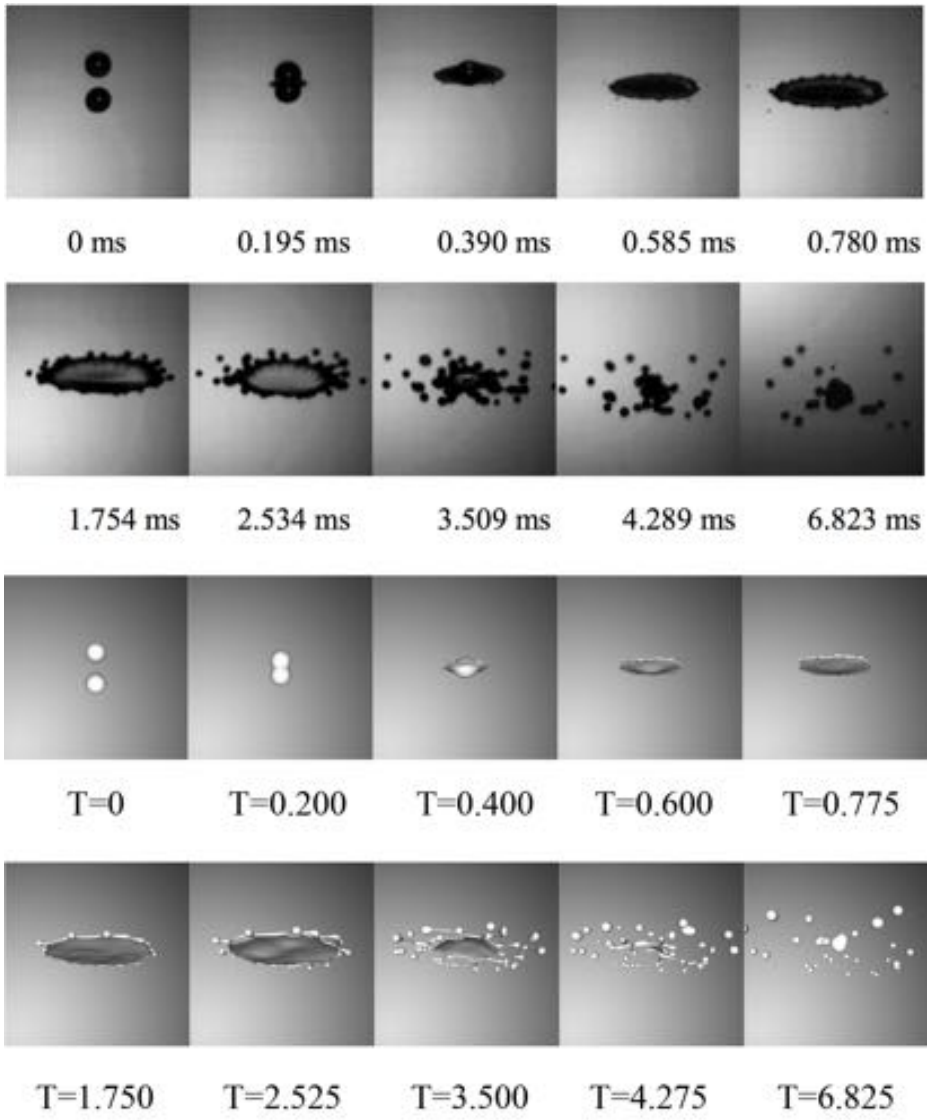


Figure 24.6: Droplet collision simulated by our technique (bottom) compared to physical experiment photos (top) by Pan et al. [2009] in Figure 3 (c).

Chapter 25

Discussion

Our method represents the first surface-only numerical scheme practical for simulation of liquid phenomena involving large, splashy motions, and therefore opens up a number of new questions and future opportunities. Most importantly, as a surface-only method, the asymptotic complexity compares favorably against the most widely-used Eulerian solvers employing regular grids, and thus provides a more tractable solution when scaling to larger and more complex scenarios.

We would like to clearly point out two major limitations in terms of applicability of the proposed technique, as a result of our assumptions: the technique is only applicable to irrotational flows, and is only applicable to contractible domains (detailed below).

First, our formulation—both the projection step and the boundary element solve—assumes that the liquid domain is irrotational on the interior. This excludes certain motions like swirling and vortices. However, for the kinds of external forces and boundary contact conditions we consider, vorticity cannot originate spontaneously in the interior of an initially-irrotational fluid of uniform density, therefore we have no risk of creating erroneous behavior, if the initial state is irrotational. On the other hand, a no-slip boundary contact condition, such as that needed for modeling the *rolling* of a droplet down an inclined plane, *can* induce rotational motion, and is therefore outside the scope of our current work. It would be interesting to superpose additional rotational degrees of freedom, such as vortex particles [Selle *et al.*, 2005; Park and Kim, 2005], vortex rings [Angelidis and Neyret, 2005; Weissmann and Pinkall, 2010], and vortex sheets [Pfaff *et al.*, 2012; Brochu *et al.*, 2012].

Second, our formulation also ignores the finite number of harmonic vector fields that cannot be represented as the gradient of harmonic potentials. It would be interesting to study in detail whether

accurately tracking the modes can reproduce a broader range of liquid phenomena. As a preliminary exploration, we initialize a solid-donut shaped liquid body with a rotational velocity (see Figure 25.1). Such a velocity cannot be represented by the gradient of a harmonic potential; note, however, that it can be represented by our surface-based vector field. We observe that the donut spins freely before the Rayleigh-Plateau instability sets in and atomizes the domain. We have measured the momentum magnitude (P), angular momentum magnitude (J), kinetic energy (T) and surface tension energy (V) of the system and plotted them in Figure 25.2; note that the system momentum and angular momentum are roughly conserved (with the angular momentum showing about 30% error over the entire simulation), and kinetic energy dissipates, as expected. If we consider a stronger

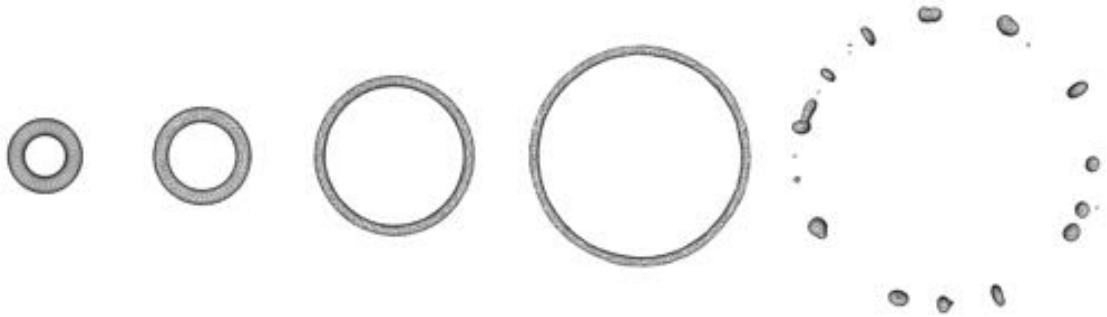


Figure 25.1: Donut-shaped liquid body with a rotational velocity.

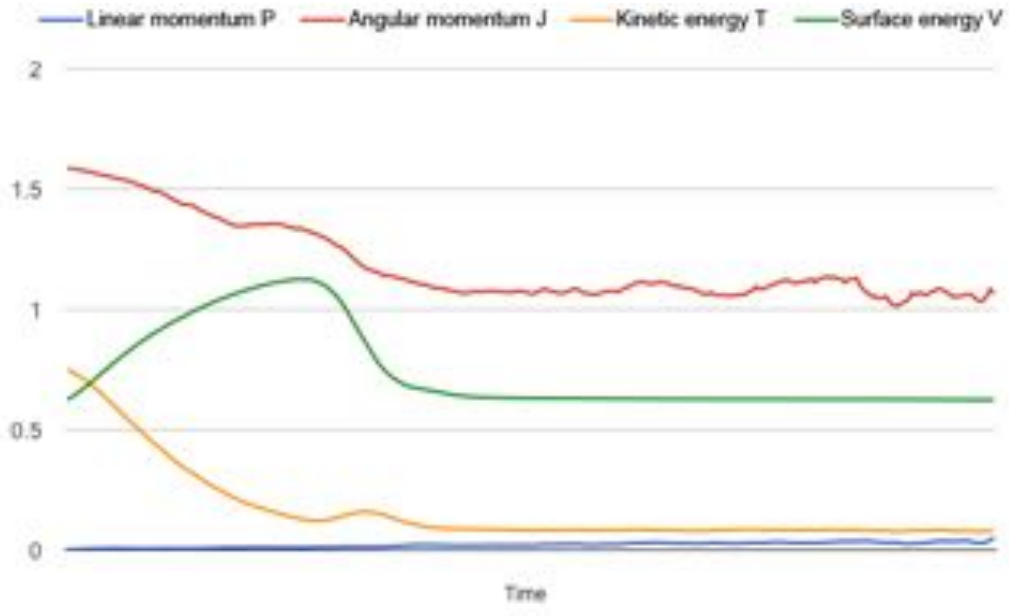


Figure 25.2: Measurement of momenta and energies.

surface tension coefficient, the donut collapses into a droplet, which also continues to spin. As stated previously, for the forces that we consider, Euler's equations specify that the modes in the kernel (such as the rotational mode here) should be conserved over time; hence at least in this scenario, qualitatively the correct behavior is observed.

Furthermore, as a result of the liquid-centric formulation, our technique is not aware of the air incompressibility (in fact, it is not aware of anything in the air volume, since air velocity is not even represented at all). This could pose problems when the desired simulation involves bubble entrainment. However, collapsing bubbles are not the first problem we would have in such a scenario; since any liquid domain with interior bubbles would not be contractible by definition, our basic assumption breaks down anyway.

Although not required by our solver, it may be desirable to evaluate the velocity at certain locations in the interior of the liquid domain in some situations, such as for one-way solid coupling (an example being entrained bubbles). This information is not directly available on our surface-only representation, but can be obtained by solving for a potential flow using the surface normal velocity as the Neumann boundary condition, as described in Section 22.2 (the "Surface-only representation" paragraph).

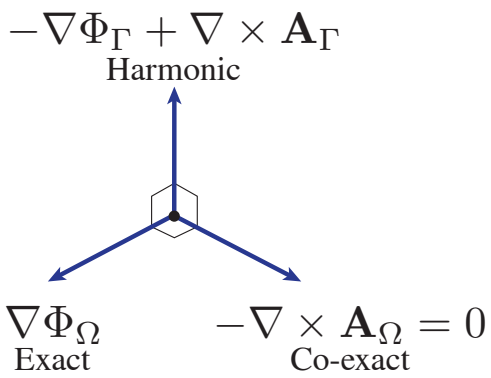
Our surface-based approach has the potential to scale favorably with scene resolution compared to volumetric-based approaches. With the fast summation techniques implemented, the complexity of the entire method (both the Helmholtz decomposition and the BEM solve) is practically linear with the number of elements in the surface mesh, which in turn scales quadratically with the inverse of the spatial resolution.

In our numerical experiments with asymmetrically colliding jets, we also attempted to reproduce the parameter regime corresponding to the "fishbone" observed by Bush and Hasha [2004]. However our simulation slowed down to ten minutes per frame as we approached the formation of the fishbone, due to the number of triangle mesh elements and the quadratic scaling of the boundary

integral quadrature. This suggests that the fishbone is an ideal testbed for these fast summation methods. A 3D grid, on the other hand, would scale cubically, due to its volumetric nature. As the space of potential geometric features also scales quadratically with the inverse resolution, our method, combined with fast summation techniques, would achieve an asymptotically optimal complexity.

Our Helmholtz decomposition-based incompressibility projection requires evaluation of two boundary integrals, and does not require a linear solve. It could be adapted for other applications, such as volume-conserving elastic deformation to remove divergence, or in other surface-only models like that of Zhang et al. [2012] as an alternative to local volume correction.

Appendix: Orthogonality of Helmholtz Decomposition



We dissect (22.3) from the perspective of the Hodge decomposition [Abraham *et al.*, 1988], which decomposes a differential k-form ω into the sum of an exact component $d\alpha$, a co-exact component $\delta\beta$, and a harmonic (closed and co-closed) component γ, while guaranteeing the components are mutually orthogonal provided certain boundary conditions are met. Applying it to our scenario, we decompose the 1-form $\overline{\mathbf{u}}^\flat$, in which case d and δ correspond to gradient and curl respectively. Based on the observation on the divergence and curl of each term above, we find that $(\nabla\Phi_\Omega)^\flat = d\Phi_\Omega$ is the exact component, $(-\nabla \times \mathbf{A}_\Omega)^\flat = -\delta(*\mathbf{A}_\Omega^\flat)$ is the co-exact component, and the two boundary integral terms $(-\nabla\Phi_\Gamma + \nabla \times \mathbf{A}_\Gamma)^\flat$ together become the harmonic component. For this scenario, Abraham et al. [1988] prove in Theorem 8.5.5 that the decomposition is indeed orthogonal, and in particular, the exact component is orthogonal to the harmonic component (summarized in the incident figure). Converting back to vector fields, this means that the velocity constructed from the two boundary integral terms,

$$\mathbf{u} = -\nabla\Phi_\Gamma + \nabla \times \mathbf{A}_\Gamma, \tag{25.1}$$

satisfies

$$\nabla \cdot \mathbf{u} = 0, \tag{25.2}$$

$$(\mathbf{u} - \overline{\mathbf{u}}) \perp \mathbf{u}, \tag{25.3}$$

and we have found the desired post-projection velocity $\mathbf{u}$.

Part IV

Conclusions

Chapter 26

Summary and Future Outlook of Surface-Only Techniques

The methods proposed in Part II and Part III add to the class of surface-only simulation techniques, which avoids any volumetric discretization, and its associated disadvantages. The superior asymptotic complexity of the surface-only techniques is a significant benefit, and its importance will only grow in the future as the hardware becomes more powerful and the demand in resolution and detail increases accordingly.

As the foundation of surface-only formulations, one must adopt certain simplifying assumptions so that the interior velocity field (not directly represented) is constrained from having too much freedom for the surface-only representation to capture. It is important to note that such an assumption is critical in the formulation of surface-only methods. A fully general velocity field cannot be adequately captured by a surface-only representation in a meaningful way (of course it is possible to construct a one-to-one mapping between $\mathbb{R}^3$ and $\mathbb{R}^2$, but such mapping cannot be continuous and thus is of little use in this context).

In both Part II and Part III, the critical assumption is that the interior velocity field is both incompressible and irrotational, which corresponds to the inviscid limit of the fluid material property. In other words, treating the viscous effect in a principled way is a challenging problem in the proposed frameworks above. Other assumptions are possible and may result in a different surface-only formulation. For example, the fully-viscous limit is an interesting case where the interior

velocity field can be constructed from the surface velocity field by solving a vector Laplace equation.

While the promises of the class of surface-only simulation techniques are exciting, this is a research area still in its infancy, and many questions remain to be answered before the techniques are ready for general application.

On the theoretical side, topology changes are currently not treated robustly in most surface-only approaches. When originally-disjoint fluid volumes are merged (e.g., when the topology changes occur in the vortex sheets smoke by Brochu [Brochu *et al.*, 2012], when two droplets collide, or when the separating film in the middle of a double-bubble pops), the two original potential fields that are possibly discontinuous must be replaced by a single continuous potential field, which may cause loss of motion modes if not handled properly. In these cases it should be recognized that the first cohomology groups of the newly formed fluid domain are altered, which may result in new harmonic modes that need to be explicitly represented.

Boundary integrals and the boundary elements method also have issues on non-manifold surfaces. Most of the boundary elements textbooks do not discuss non-manifold surfaces at all. Although a more solid theoretical footing is yet to be found, the results presented in Part II suggest that a straightforward extension of the manifold formulation to the non-manifold case is possible.

On the application side, an important issue to address is coupling surface-only solvers with traditional volumetric (especially Eulerian grid) solvers, due to the rich collection of features available to the latter, accumulated over the past decades. For example, the vortex sheets bubbles solver coupled to a traditional liquid solver can be used to simulate a foaming jug of beer, where neither is sufficient on its own. It is also conceivable that the surface-only liquids solver can be used on top of a background low-resolution Eulerian grid to simulate a large-scale waterfall, where the surface-only solver preserves all the high frequency details while the background grid cuts off long-range communication between different parts of the connected body of water and reduces the boundary integral cost significantly. Although the benefits of such coupling is readily obvious, it is challenging to implement due to the distinct representation employed by each approach.

Part V

Bibliography

Bibliography

[Abraham *et al.*, 1988] R Abraham, Jerrold E Marsden, and Tudor Ratiu. *Manifolds, Tensor Analysis, and Applications*. Springer, 1988.

[Agishtein and Migdal, 1989] Michael E Agishtein and Alexander A Migdal. Dynamics of vortex surfaces in three dimensions: Theory and simulations. *Physica D: Nonlinear Phenomena*, 40(1):91–118, 1989.

[Alexa, 2002] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–198, 2002.

[Alliez *et al.*, 2008] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. In Leila Floriani and Michela Spagnuolo, editors, *Shape Analysis and Structuring*, pages 53–82. Springer, Berlin, 2008.

[Anderson *et al.*, 2010] John C. Anderson, Christoph Garth, Mark A. Duchaineau, and Kenneth I. Joy. Smooth, volume-accurate material interface reconstruction. *IEEE TVCG*, 16(5):802–814, 2010.

[Angelidis and Neyret, 2005] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *Symposium on Computer Animation*, pages 87 – 96, 2005.

[Angelidis *et al.*, 2006] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *Symposium on Computer Animation*, pages 25–32, 2006.

[Atkinson, 1997] Kendall E Atkinson. The numerical solution of boundary integral equations. In *The State of the Art in Numerical Analysis, pp. 223–259*, 1997.

[Baker *et al.*, 1982] Gregory R Baker, Daniel I Mirron, and Steven A. Orszag. Generalized vortex methods for free-surface flow problems. *J. Fluid Mech.*, 123:477–501, 1982.

[Bargteil *et al.*, 2006] Adam W. Bargteil, James F. O'Brien, Tolga G. Goktekin, and John A. Strain. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1):19–38, January 2006.

[Barnat and Pollard, 2012] Alfred Barnat and Nancy S Pollard. Smoke sheets for graph-structured vortex filaments. In *Symposium on Computer Animation*, pages 77–86, 2012.

[Batty and Bridson, 2008] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*, pages 219–228, 2008.

[Batty *et al.*, 2007] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 26(3):100, 2007.

[Batty *et al.*, 2010] Christopher Batty, Stefan Xenos, and Ben Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics)*, 29(2):695–704, 2010.

[Batty *et al.*, 2012] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. Discrete viscous sheets. *ACM Trans. Graph. (SIGGRAPH)*, 31(4):113, 2012.

[Bernstein and Wojtan, 2013] Gilbert Bernstein and Chris Wojtan. Putting holes in holey geometry: Topology change for arbitrary surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 32(4):34, 2013.

[Bojsen-Hansen and Wojtan, 2013a] Morten Bojsen-Hansen and Chris Wojtan. Liquid surface tracking with error compensation. *ACM Trans. Graph. (SIGGRAPH)*, 32(4):79:1–79:10, 2013.

[Bojsen-Hansen and Wojtan, 2013b] Morten Bojsen-Hansen and Chris Wojtan. Liquid surface tracking with error compensation. *ACM Trans. Graph.*, 32(4):68, 2013.

[Botsch and Kobbelt, 2004] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 185–192, New York, 2004. ACM.

[Brakke, 1992] Ken Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[Bridson *et al.*, 2002] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):594–603, 2002.

[Brochu and Bridson, 2009] Tyson Brochu and Robert Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput.*, 31(4):2472–2493, 2009.

[Brochu *et al.*, 2010] Tyson Brochu, Christopher Batty, and Robert Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):47, 2010.

[Brochu *et al.*, 2012] Tyson Brochu, Todd Keeler, and Robert Bridson. Linear-time smoke animation with vortex sheets. In *Symposium on Computer Animation*, pages 87–95, 2012.

[Brochu, 2006] Tyson Brochu. Fluid animation with explicit surface meshes and boundary-only dynamics. Master's thesis, Citeseer, 2006.

[Bronson *et al.*, 2012] Jonathan R. Bronson, Joshua A. Levine, and Ross T. Whitaker. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In Xiangmin Jiao and Jean-Christophe Weill, editors, *International Meshing Roundtable*, pages 191–209, Berlin, 2012. Springer.

[Busaryev *et al.*, 2012] Oleksiy Busaryev, Tamal K. Dey, Huamin Wang, and Zhong Ren. Animating bubble interactions in a liquid foam. *ACM Trans. Graph. (SIGGRAPH)*, 31(4):63, 2012.

[Bush and Hasha, 2004] John WM Bush and Alexander E Hasha. On the collision of laminar jets: fluid chains and fishbones. *Journal of fluid mechanics*, 511:285–310, 2004.

[Campen and Kobbelt, 2010] Marcel Campen and Leif Kobbelt. Exact and robust (self-)intersections for polygonal meshes. *Computer Graphics Forum (Eurographics)*, 29(2):397–406, June 2010.

[Cecka and Layton, 2015] Cris Cecka and Simon Layton. FMMTL: FMM template library a generalized framework for kernel matrices. In *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, pages 611–620. Springer, 2015.

[Chen *et al.*, 2003] CS Chen, MA Golberg, and RS Schaback. Recent developments in the dual reciprocity method using compactly supported radial basis functions. *Transformation of Domain Effects to the Boundary (YF Rashed and CA Brebbia, eds), WITPress, Southampton, Boston*, pages 138–225, 2003.

[Chentanez *et al.*, 2007] Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228. Eurographics Association, 2007.

[Clark *et al.*, 2012] Bryan Clark, Navamita Ray, and Xiangmin Jiao. Surface mesh optimization, adaption, and untangling with high-order accuracy. In Xiangmin Jiao and Jean-Christophe Weill, editors, *International Meshing Roundtable*, pages 385–402, Berlin, 2012. Springer.

[Clausen *et al.*, 2013] Pascal Clausen, Martin Wicke, Jonathan Richard Shewchuk, and James F. O'Brien. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph.*, 32(2):17, 2013.

[Cohen-Steiner and Morvan, 2003] David Cohen-Steiner and Jean-Marie Morvan. Restricted Delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321. ACM, 2003.

[Cottet and Koumoutsakos, 2000] George-Henri Cottet and Petros Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[Crane *et al.*, 2013] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Trans. Graph. (SIGGRAPH)*, 32(4):61, 2013.

[Da *et al.*, 2014a] Fang Da, Christopher Batty, and Eitan Grinspun. A convergence study of multimaterial mesh-based surface tracking. Technical report, Columbia University, 2014.

[Da *et al.*, 2014b] Fang Da, Christopher Batty, and Eitan Grinspun. Multimaterial mesh-based surface tracking. *ACM Trans. Graph. (SIGGRAPH)*, 33(4):112:1–112:11, 2014.

[Da *et al.*, 2015] Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Double bubbles sans toil and trouble: discrete circulation-preserving vortex sheets for soap films and foams. *ACM Trans. Graph. (SIGGRAPH)*, 34(4):149, 2015.

[Da *et al.*, 2016] Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Surface-only liquids. *ACM Trans. Graph. (SIGGRAPH)*, 35(4), July 2016.

[Davidson, 2000] Malcolm R Davidson. Boundary integral prediction of the spreading of an inviscid drop impacting on a solid surface. *Chemical Engineering Science*, 55(6):1159–1170, 2000.

[de Goes *et al.*, 2008] Fernando de Goes, Siome Goldstein, and Luis Velho. A simple and flexible framework to adapt dynamic meshes. *Computers and Graphics*, 32(2):141–148, 2008.

[de Sousa *et al.*, 2004] F. S. de Sousa, Norberto Mangiavacchi, L. G. Nonato, Antonio Castelo, Murilo F. Tomé, and Sean McKee. A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces. *J. Comp. Phys.*, 198(2):469–499, 2004.

[Derby, 2010] Brian Derby. Inkjet printing of functional and structural materials: fluid property requirements, feature stability, and resolution. *Annual Review of Materials Research*, 40:395–414, 2010.

[Du *et al.*, 2006] Jian Du, Brian Fix, James Glimm, Xicheng Jia, Xiaolin Li, Yuanhua Li, and Lingling Wu. A simple package for front tracking. *J. Comp. Phys.*, 213(2):613–628, 2006.

[Duffy, 1982] Michael G Duffy. Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM journal on Numerical Analysis*, 19(6):1260–1262, 1982.

[Dunyach *et al.*, 2013] Marion Dunyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. Adaptive remeshing for real-time mesh deformation. In *Eurographics short papers*, pages 29–32, Girona, Spain, 2013. Eurographics Association.

[Durikovic, 2001] Roman Durikovic. Animation of soap bubble dynamics, cluster formation and collision. *Computer Graphics Forum (Eurographics)*, 20(3):67–76, 2001.

[Dyadechko and Shashkov, 2008] Vadim Dyadechko and M. Shashkov. Reconstruction of multi-material interfaces from moment data. *J. Comp. Phys.*, 227(11):5361–5384, 2008.

[Eckstein *et al.*, 2007] I. Eckstein, J.-P. Pons, Y. Tong, C.-C. J. Kuo, and Mathieu Desbrun. Generalized surface flows for mesh processing. In *Symposium on Geometry Processing*, pages 183–192, 2007.

[Elcott *et al.*, 2007] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[Enright *et al.*, 2002a] Doug Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.*, 183(1):83–116, 2002.

[Enright *et al.*, 2002b] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[Faeth, 1977] GM Faeth. Current status of droplet and liquid combustion. *Progress in Energy and Combustion Science*, 3(4):191–224, 1977.

[Fedkiw *et al.*, 1999] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2):457–492, 1999.

[Florez and Power, 2001] WF Florez and H Power. Comparison between continuous and discontinuous boundary elements in the multidomain dual reciprocity method for the solution of the two-dimensional Navier–Stokes equations. *Engineering analysis with boundary elements*, 25(1):57–69, 2001.

[Foster and Fedkiw, 2001] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30. ACM, 2001.

[Foster and Metaxas, 1996] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical models and image processing*, 58(5):471–483, 1996.

[Glassner, 2000a] Andrew Glassner. Soap bubbles: Part 1. *IEEE Computer Graphics and Applications*, 20(5):76–84, 2000.

[Glassner, 2000b] Andrew Glassner. Soap bubbles: Part 2. *IEEE Computer Graphics and Applications*, 20(6):99–109, 2000.

[Glimm *et al.*, 1998] James Glimm, John W. Grove, Xiaolin Li, Keh-ming Shyue, Yanni Zeng, and Qiang Zhang. Three-dimensional front tracking. *SIAM J. Sci. Comput.*, 19(3):703–727, 1998.

[Glimm *et al.*, 2000] James Glimm, John W. Grove, X. L. Li, and D. C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comput.*, 21(6):2240–2256, 2000.

[Golas *et al.*, 2012] Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming C. Lin. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. (SIGGRAPH Asia)*, 31(6):148, 2012.

[Greengard and Rokhlin, 1987] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[Greenwood and House, 2004] S. T. Greenwood and Donald H. House. Better with bubbles. In *Symposium on Computer Animation*, 2004.

[Hahn and Wojtan, 2015] David Hahn and Chris Wojtan. High-resolution brittle fracture simulation with boundary elements. *ACM Transactions on Graphics (TOG)*, 34(4):151, 2015.

[Harmon *et al.*, 2008] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):23, 2008.

[Hartmann, 2012] Friedel Hartmann. *Introduction to boundary elements: theory and applications*. Springer Science & Business Media, 2012.

[Hirt and Nichols, 1981] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comp. Phys.*, 39(1):201–225, 1981.

[Hong and Kim, 2003] Jeong-Mo Hong and Chang-Hun Kim. Animation of bubbles in liquid. *Computer Graphics Forum*, 22(3):253–262, 2003.

[Hong and Kim, 2005] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):915–920, July 2005.

[Hong *et al.*, 2008] Jeong-Mo Hong, Ho-Young Lee, Jong-Chul Yoon, and Chang-Hun Kim. Bubbles Alive. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):48, 2008.

[Hubeli and Gross, 2000] Andreas Hubeli and Markus Gross. Fairing of non-manifold models for visualization. In *Proceedings of Visualization '00*, pages 407–414, Salt Lake City, Utah, USA, 2000. IEEE Computer Society Press.

[Ihmsen *et al.*, 2014a] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible SPH. *Visualization and Computer Graphics, IEEE Transactions on*, 20(3):426–435, 2014.

[Ihmsen *et al.*, 2014b] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. *Eurographics 2014 - State of the Art Reports*, 2014.

[James and Pai, 1999] Doug L James and Dinesh K Pai. ArtDefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72. ACM Press/Addison-Wesley Publishing Co., 1999.

[Jiao and Alexander, 2005] Xiangmin Jiao and Phillip J. Alexander. Parallel feature-preserving mesh smoothing. In *Proceedings of the 2005 international conference on Computational Science and Its Applications - Volume Part IV*, pages 1180–1189, Singapore, 2005. Springer-Verlag.

[Jiao and Bayyana, 2008] Xiangmin Jiao and Narasimha Bayyana. Identification of C1 and C2 discontinuities for surface meshes in CAD. *Computer Aided Design*, 40(2):160–175, 2008.

[Jiao *et al.*, 2010] Xiangmin Jiao, Andrew Colombi, Xinlai Ni, and John Hart. Anisotropic mesh adaptation for evolving triangulated surfaces. *Engineering with Computers*, 26(4):363–376, 2010.

[Jiao, 2007] Xiangmin Jiao. Face offsetting: A unified approach for explicit moving interfaces. *J. Comp. Phys.*, 220(2):612–625, 2007.

[Kang *et al.*, 2000] Myungjoo Kang, Ron Fedkiw, and Xu-Dong Liu. A boundary condition capturing method for multiphase incompressible flow. *SIAM J. Sci. Comput.*, 15(3):323–360, 2000.

[Keeler and Bridson, 2014] Todd Keeler and Robert Bridson. Ocean waves animation using boundary integral equations and explicit mesh tracking. In *Symposium on Computer Animation*, 2014.

[Kim *et al.*, 2007] Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. (SIGGRAPH)*, 26(3):98, 2007.

[Kim *et al.*, 2009] Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. Stretching and wiggling liquids. *ACM Trans. Graph. (SIGGRAPH Asia)*, 28(5):120, 2009.

[Kim *et al.*, 2010] Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. A practical simulation of dispersed bubble flow. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):70, 2010.

[Kim, 2010] Byungmoon Kim. Multiphase fluid simulation using regional level sets. *ACM Trans. Graph. (SIGGRAPH Asia)*, 29(6):175, 2010.

[Kuck *et al.*, 2002] Henrick Kuck, Christian Vogelsang, and Günter Greiner. Simulation and rendering of liquid foams. In *Graphics Interface*, pages 81–88, 2002.

[Kuprat *et al.*, 2003] Andrew Kuprat, Denise George, Galen Straub, and Melik C. Demirel. Modeling microstructure evolution in three dimensions with Grain3D and LaGriT. *Computational Materials Science*, 28(1):199–208, 2003.

[Ladyzhenskaya, 1963] Olga A Ladyzhenskaya. *The mathematical theory of viscous incompressible flow*. Gordon and Breach New York, 1963.

[Lazar, 2011] Emanuel Lazar. *The evolution of cellular structures via curvature flow*. PhD thesis, Columbia University, 2011.

[Losasso *et al.*, 2006] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):812–819, 2006.

[Mantič, 1993] Vladislav Mantič. A new formula for the c-matrix in the somigliana identity. *Journal of Elasticity*, 33(3):191–201, 1993.

[McKee *et al.*, 2008] Sean McKee, Murilo F. Tomé, Valdemir G. Ferreira, José A. Cuminato, Antonio Castelo, F. S. de Sousa, and Norberto Mangiavacchi. The MAC method. *Computers & Fluids*, 37(8):907–930, September 2008.

[Meyer *et al.*, 2002] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, pages 35–54, Berlin, Germany, 2002. Springer-Verlag.

[Meyer *et al.*, 2008] Miriah Meyer, Ross T. Whitaker, Robert M. Kirby, Christian Ledergerber, and Hanspeter Pfister. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE TVCG*, 14(6):1539–1546, 2008.

[Mihalef *et al.*, 2006] Viorel Mihalef, B. Unlusu, Dimitris Metaxas, Mark Sussman, and M. Y. Hussaini. Physics based boiling simulation. In *Symposium on Computer Animation*, pages 317–324, 2006.

[Misztal *et al.*, 2010] Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. Optimization-based fluid simulation on unstructured meshes. In *VRIPHYS*, pages 11–20, 2010.

[Misztal *et al.*, 2012] Marek Misztal, Kenny Erleben, Adam W. Bargteil, B. Bunch Christensen, Andreas Baerentzen, and Robert Bridson. Multiphase flow of immiscible fluids on unstructured moving meshes. In *Symposium on Computer Animation*, pages 97–106, Lausanne, Switzerland, 2012. Eurographics Association.

[Mora *et al.*, 2008] L.A. Barrales Mora, G. Gottstein, and L. S. Schvindlerman. Three-dimensional grain growth: Analytical approaches and computer simulations. *Acta Materialia*, 56(1):5915–5926, 2008.

[Mullen *et al.*, 2009] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):38, 2009.

[Müller *et al.*, 2005] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. Particle-based fluid-fluid interaction. In *Symposium on Computer Animation*, pages 237–244, Los Angeles, CA, USA, 2005. ACM.

[Müller, 2009] Matthias Müller. Fast and robust tracking of fluid surfaces. In *Symposium on Computer Animation*, pages 237–245, New York, NY, USA, 2009. ACM.

[Narain *et al.*, 2012] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. (SIGGRAPH Asia)*, 31(6):147, 2012.

[Osher and Fedkiw, 2002] Stanley Osher and Ron Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2002.

[Osher and Sethian, 1988] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[Pan *et al.*, 2009] Kuo-Long Pan, Ping-Chung Chou, and Yu-Jen Tseng. Binary droplet collision at high Weber number. *Phys. Rev. E*, 80:036301, Sep 2009.

[Pan *et al.*, 2012] Hao Pan, Yi-King Choi, Yang Liu, Wenchao Hu, Qiang Du, Konrad Polthier, Caiming Zhang, and Wenping Wang. Robust modeling of constant mean curvature surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 31(4):85, 2012.

[Park and Blair, 1975] Jin Yong Park and L Mo Blair. The effect of coalescence on drop size distribution in an agitated liquid-liquid dispersion. *Chemical Engineering Science*, 30(9):1057–1064, 1975.

[Park and Kim, 2005] Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *Symposium on Computer Animation*, pages 261–270, 2005.

[Patkar *et al.*, 2013] Saket Patkar, Mridul Aanjaneya, Dmitry Karpman, and Ronald Fedkiw. A Hybrid Lagrangian-Eulerian Formulation for Bubble Generation and Dynamics. In *Symposium on Computer Animation*, pages 105–114, 2013.

[Pellerin *et al.*, 2011] Jeanne Pellerin, Bruno Lévy, and Guillaume Caumon. Topological control for isotropic remeshing of nonmanifold surfaces with varying resolution: application to 3D structural models. In *IAMG*, pages 678–688, Salzburg, Austria, 2011. International Association of Mathematical Geosciences.

[Pfaff *et al.*, 2009] Tobias Pfaff, Nils Thuerey, Andrew Selle, and Markus Gross. Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph. (SIGGRAPH Asia)*, 28(5):121, 2009.

[Pfaff *et al.*, 2012] Tobias Pfaff, Nils Thuerey, and Markus Gross. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph. (SIGGRAPH)*, 31(4):112:1–112:8, 2012.

[Phillips, 1933] Henry Bayard Phillips. *Vector analysis*. Wiley New York, 1933.

[Pinkall and Polthier, 1993] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[Pons and Boissonnat, 2007a] J.-P. Pons and J.-D. Boissonnat. A Lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space. *Computer Graphics Forum*, 26(2):227–239, 2007.

[Pons and Boissonnat, 2007b] J.-P. Pons and J.-D. Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *CVPR*, pages 1–8, Minneapolis, Minnesota, USA, 2007. IEEE.

[Power and Partridge, 1994] Henry Power and Paul W Partridge. The use of Stokes' fundamental solution for the boundary only element formulation of the three-dimensional Navier-Stokes equations for moderate Reynolds numbers. *International journal for numerical methods in engineering*, 37(11):1825–1840, 1994.

[Pozrikidis, 2000] C. Pozrikidis. Theoretical and computational aspects of the self-induced motion of three-dimensional vortex sheets. *J. Fluid Mech.*, 425:335–366, 2000.

[Quan and Schmidt, 2007] Shaoping Quan and David P. Schmidt. A moving mesh interface tracking method for 3D incompressible two-phase flows. *Journal of Computational Physics*, 221(2):761–780, 2007.

[Quan *et al.*, 2009] Shaoping Quan, Jing Lou, and David P. Schmidt. Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations. *Journal of Computational Physics*, 228(7):2660–2675, 2009.

[Sauter and Schwab, 2011] Stefan A Sauter and Christoph Schwab. *Boundary element methods*. Springer, 2011.

[Saye and Sethian, 2012] Robert Saye and James Sethian. Analysis and applications of the Voronoi Implicit Interface Method. *J. Comp. Phys.*, 231(18):6051–6085, 2012.

[Saye and Sethian, 2013] Robert Saye and James Sethian. Multiscale Modeling of Membrane Rearrangement, Drainage, and Rupture in Evolving Foams. *Science*, 340(6133):720–724, 2013.

[Schroeder *et al.*, 2012] Craig Schroeder, Wen Zheng, and Ronald Fedkiw. Semi-implicit surface tension formulation with a Lagrangian surface mesh on an Eulerian simulation grid. *Journal of Computational Physics*, 231(4):2092–2115, 2012.

[Selle *et al.*, 2005] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):910–914, 2005.

[Sethian, 1999] James Sethian. *Level set methods and fast marching methods*. Cambridge University Press, 1999.

[Solenthaler and Pajarola, 2008] Barbara Solenthaler and Renato Pajarola. Density contrast SPH interfaces. In *Symposium on Computer Animation*, pages 211–218, Dublin, 2008. Eurographics Association.

[Stam, 1999] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[Stanculescu *et al.*, 2011] Lucian Stanculescu, Raphaelle Chaine, and Marie-Paule Cani. Freestyle: Sculpting meshes with self-adaptive topology. *Computers and Graphics*, 35(3):614–622, 2011.

[Starinshak *et al.*, 2014] David P. Starinshak, Smadar Karni, and Philip L. Roe. A new level set model for multimaterial flows. *J. Comp. Phys.*, In press, 2014.

[Stock *et al.*, 2008] Mark J. Stock, Werner J A Dahm, and Gretar Tryggvason. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheetmethod. *J. Comp. Phys.*, 227(21):9021–9043, 2008.

[Stock, 2006] Mark Stock. *A regularized inviscid vortex sheet method for three-dimensional flows with density interfaces*. PhD thesis, University of Michigan, 2006.

[Syha and Weygand, 2010] M. Syha and D. Weygand. A generalized vertex dynamics model for grain growth in three dimensions. *Modelling Simul. Mater. Sci. Eng.*, 18(1):015010, 2010.

[Taylor, 1976] Jean E. Taylor. The structure of singularities in soap-bubble-like and soap-film-like minimal surfaces. *Annals of Mathematics*, 103:489–539, 1976.

[Thürey *et al.*, 2010] Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics (TOG)*, 29(4):48, 2010.

[Toutant *et al.*, 2012] A. Toutant, B. Mathieu, and O. Lebaigue. Volume-conserving smoothing for front tracking methods. *Computers & Fluids*, 67:16–25, 2012.

[Tryggvason, 1988] Gretar Tryggvason. Numerical simulations of the Rayleigh-Taylor instability. *J. Comp. Phys.*, 75(2):253–282, 1988.

[Vines *et al.*, 2014] Mauricio Vines, Ben Houston, Jochen Lang, and Won-Sook Lee. Vortical inviscid flows with two-way solid-fluid coupling. *IEEE TVCG*, 20(2):303–315, 2014.

[Wakai *et al.*, 2000] Fumihiro Wakai, Naoya Enomoto, and Hirosh Ogawa. Three-dimensional microstructural evolution in ideal grain growth - general statistics. *Acta Materialia*, 48(1):1297–1311, 2000.

[Wang *et al.*, 2005] Huamin Wang, Peter J Mucha, and Greg Turk. Water drops on surfaces. *ACM Transactions on Graphics (TOG)*, 24(3):921–929, 2005.

[Weaire and Hutzler, 2001] Denis Weaire and Stefan Hutzler. *Physics of Foams*. Oxford University Press, New York, 2001.

[Weaire, 2013] Denis Weaire. A fresh start for foam physics. *Science*, 340(6133):693–694, 2013.

[Weißmann and Pinkall, 2009] Steffen Weißmann and Ulrich Pinkall. Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *VRIPHYS*, pages 1–10, 2009.

[Weissmann and Pinkall, 2010] Steffen Weissmann and Ulrich Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):115:1–115:12, 2010.

[Weygand and Brechet, 1999] D. Weygand and Y. Brechet. Three-dimensional grain growth: a vertex dynamics simulation. *Philosophical Magazine B*, 79(5):703–716, 1999.

[Wicke *et al.*, 2010] Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):49, 2010.

[Wojtan *et al.*, 2009] Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH)*, 28(3):76, 2009.

[Wojtan *et al.*, 2010] Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics (TOG)*, 29(4):50, 2010.

[Wojtan *et al.*, 2011] Chris Wojtan, Matthias Muller-Fischer, and Tyson Brochu. Liquid simulation with mesh-based surface tracking. In *SIGGRAPH Courses*, page 8, Vancouver, 2011. ACM.

[Wu and Sullivan, 2003] Ziji Wu and John M. Sullivan. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, 2003.

[Xue *et al.*, 2001] Ming Xue, Hongbo Xü, Yuming Liu, and Dick KP Yue. Computations of fully nonlinear three-dimensional wave–wave and wave–body interactions. Part 1. Dynamics of steep three-dimensional waves. *Journal of Fluid Mechanics*, 438:11–39, 2001.

[Ying and Zorin, 2001] Lexing Ying and Denis Zorin. Nonmanifold subdivision. In *Proceedings of Visualization '01*, pages 325 – 332, San Diego, CA, USA, 2001. IEEE.

[Yu *et al.*, 2012] Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. Explicit mesh surfaces for particle based fluids. *Computer Graphics Forum (Eurographics)*, 31(2):815–824, 2012.

[Yuan *et al.*, 2012] Zhan Yuan, Yizhou Yu, and Wenping Wang. Object-space multiphase implicit functions. *ACM Trans. Graph. (SIGGRAPH)*, 31(4):114, 2012.

[Zaharescu *et al.*, 2011] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE TPAMI*, 33(4):823–837, 2011.

[Zhang and Bridson, 2014] Xinxin Zhang and Robert Bridson. A PPPM fast summation method for fluids and beyond. *ACM Trans. Graph. (SIGGRAPH Asia)*, 33(6):206, 2014.

[Zhang *et al.*, 2012] Yizhong Zhang, Huamin Wang, Shuai Wang, Yiying Tong, and Kun Zhou. A deformable surface model for real-time water drop animation. *IEEE TVCG*, 18(8):1281–1289, 2012.

[Zhang *et al.*, 2015] Xinxin Zhang, Robert Bridson, and Chen Greif. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG)*, 34(4), 2015.

[Zhao *et al.*, 1996] Hong-Kai Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comp. Phys.*, 127(1):179–195, 1996.

[Zheng *et al.*, 2006] Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. Simulation of bubbles. In *Symposium on Computer Animation*, pages 325–333, Vienna, 2006. Eurographics Association.

[Zhu *et al.*, 2014] Bo Zhu, Ed Quigley, Matthew Cong, Justin Solomon, and Ronald Fedkiw. Codimensional surface tension flow on simplicial complexes. *ACM Trans. Graph. (SIGGRAPH)*, 33(4):111, 2014.

[Zhu *et al.*, 2015] Yufeng Zhu, Robert Bridson, and Chen Greif. Simulating rigid body fracture with surface meshes. *ACM Transactions on Graphics (TOG)*, 34(4):150, 2015.

[Zilske *et al.*, 2008] Michael Zilske, Hans Lamecker, and Stefan Zachow. Adaptive remeshing of non-manifold surfaces. In *Eurographics short papers*, Crete, Greece, 2008. Eurographics Association.

[Zorin *et al.*, 1996] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 1996*, pages 189–192, New Orleans, 1996. ACM.