Topology Change Mechanisms in Explicit Interface Tracking: An Interdisciplinary Survey

Christopher Batty

School of Computer Science, University of Waterloo, 200 University Ave W, Waterloo, N2L 3G1, Ontario, Canada

Abstract

Explicit interface tracking algorithms are used to track the evolving geometry and topology of deforming material interfaces, most often using triangle meshes. In work spanning more than forty years, this powerful approach has been applied extensively across computational physics, materials science, engineering, computer graphics, computer vision, and more. As a result, many different techniques have been proposed to carry out topological changes (merging, splitting, etc.) of explicit interfaces. However, the interdisciplinary nature of this topic has also led to a relatively fragmented body of literature. The present work therefore provides a thorough survey of topology change mechanisms used in explicit interface tracking, describing and categorizing the existing methods and tracing their evolution across domains. We consider both the two-phase case and generalizations to three or more phases, and conclude with a discussion of potential directions for future investigation.

Keywords:
explicit interface tracking, front tracking, topological change, merging, splitting, multiphase flow

1. Introduction

The time evolution of deforming interfaces—curves in two dimensions or surfaces in three—is central to a wide range of geometric, physical, and creative problems. Such interfaces arise, for example, as the boundary between two immiscible fluids in a multiphase flow, the delicate film of a soap bubble, the surface of a tumor in medical image data, or the intricate geometry of a digitally sculpted 3D model. Because these interfaces delineate distinct materials, phases, or spatial regions of practical importance, many fundamental tasks in science, engineering, and entertainment depend on being able to computationally simulate their motion and deformation over time. This process is known as *interface tracking*. An essential aspect of interface tracking is the handling of *topological changes*, such as merging (coalescence) and splitting (breakup) (see Figure 1 for instance).

Two broad families of numerical interface tracking approaches exist. *Implicit* (or interface capturing) methods represent the interface indirectly: for example, using signed distance fields (SDF) (Sethian et al., 1999; Osher and Fedkiw, 2003), phase fields (Boettinger et al., 2002; Sun and Beckermann, 2007), volume fractions (Hirt and Nichols, 1981; Rider and Kothe, 1998), or sets of smoothed particles (Monaghan, 1994). Their significant success and widespread adoption is due in no small part to their ability to handle

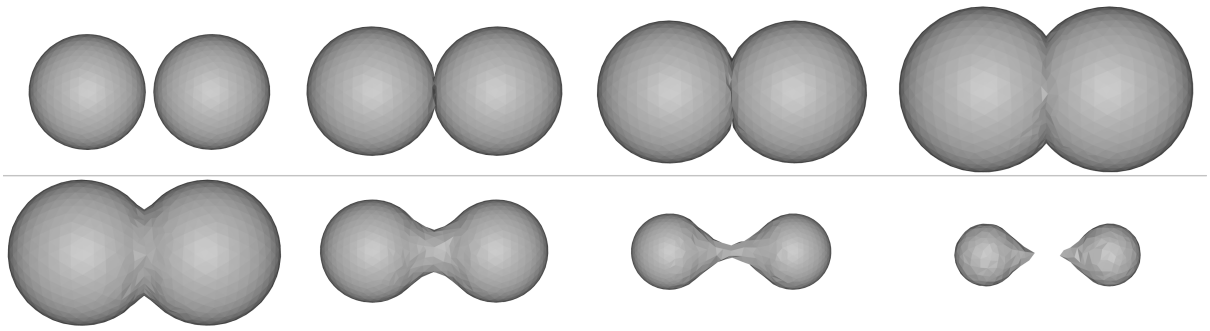

Figure 1: A sequence of snapshots from a simple geometric flow of an explicit triangle mesh in which two spheres expand uniformly outward in the surface normal direction (via *face offsetting* (Jiao, 2006)) and merge (coalesce) once they come in contact (top row, left to right). The direction of normal flow is then reversed, causing the spheres to shrink inwards and then undergo splitting (breakup) once the thread joining them becomes thin enough (bottom row, left to right). The computation was performed using the *Los Topos* code (Da et al., 2014). (To better visualize individual mesh triangles, a deliberately low resolution mesh is used and the surface is displayed with simple flat shading.)

topological changes of the interface automatically during the evolution of the associated Eulerian data fields or particle positions. However, each such scheme has strengths and weaknesses; depending on the particular implicit method chosen, it may tend to lose volume or accuracy, smooth away high-frequency details or thin features, offer low quality estimates of differential geometric quantities, or require an interface reconstruction process that can introduce further errors.

We focus instead on *explicit* interface tracking methods. They represent the interface directly, usually as a triangulated mesh whose vertices are advected forward according to the vector field prescribing the interface's evolution. Among other benefits, this approach offers ready access to many geometric quantities, can accurately preserve slender and fine-scale features, and integrates naturally with mesh-based discretizations used in embedded boundary PDE methods (Unverdi and Tryggvason, 1992), boundary integral solvers (Pozrikidis, 1992), surface PDE solvers (Dziuk and Elliott, 2013), and many animation and modeling techniques (Wojtan et al., 2011; Botsch et al., 2010). However, in scenarios where a topology change is expected to occur, if no other action is taken, separate explicit interfaces can come arbitrarily close to one another, or they may collide and interpenetrate. In the former case, the desired topology change will remain unresolved, leaving a narrow gap; in the latter case, the interface is left in a geometrically (and often physically) invalid intersecting state, that is, the interface mesh is no longer properly embedded in $\mathbb{R}^n$. Therefore, in contrast to implicit interface tracking methods, explicit interface tracking methods require deliberate treatment of topological changes. Designing such operations to be robust, accurate, and efficient is a longstanding challenge.

Over the past four decades or so, researchers in computational physics, computer vision, computer graphics, and materials science, among other disciplines, have developed a diverse range of mechanisms to perform topology changes on explicit meshes. Implicit topology change methods temporarily convert the surface to a convenient implicit representation to resolve the topology and then reconstruct from it a new mesh (or mesh patch); by contrast, explicit topology change methods perform surgery directly on the mesh itself, either via a global, Boolean-like, in-place topology correction or incremental, localized mesh edits that slightly deform the surface shape. However, while many connections exist between the methods proposed across disciplines, the resulting literature

remains relatively fragmented, with varying terminology being used and similar ideas often arising independently.

This survey therefore aims to review and unify the literature on topology-change methods for explicit interface tracking. We categorize existing techniques by their underlying mechanisms, describe and relate strategies used across disciplines, and trace the historical development of both implicit and explicit approaches to topology change. A key objective is to highlight and clarify connections among the many existing approaches and thereby establish a consistent foundation for future investigations.

We set the stage for our review by summarizing basic interface tracking concepts (§2). With this background established, we review the three overarching families of topology change mechanisms used in explicit interface tracking: implicit topology change (§3), global explicit topology change (§4), and local explicit topology change (§5). Next, we widen our scope to consider methods for multimaterial and multicellular domains involving more than two distinguished regions, for which additional types of topological change arise (§6). We then briefly highlight connections to the closely related topics of mesh repair and mesh Booleans (§7). Finally, we conclude by offering additional perspectives on the methods discussed (§8) and outlining some potential directions for further research (§9).

2. Background

We begin with a brief review of the key concepts in explicit interface tracking that are most relevant to topology change. For a rigorous explication and analysis of the mathematical theory underpinning interface tracking more broadly, we refer the reader to the work of Zhang and Fogelson (2016). The course notes by Wojtan et al. (2011) provide another useful entry point to general topology-adaptive explicit interface tracking from a computer animation perspective.

2.1. Interface Tracking Basics

Consider two nonempty disjoint regions of space (open sets), $\Omega^+, \Omega^- \subset \mathbb{R}^n$ for $n = 2$ or $n = 3$, with a shared boundary or *interface* I, so that

$$\Omega^+ \sqcup \Omega^- \sqcup I = \mathbb{R}^n, \qquad I := \overline{\Omega^+} \cap \overline{\Omega^-}. \tag{1}$$

That is, space is divided into an interior Ω^- and exterior Ω^+ separated by the interface I, which has dimension $n-1$. The domains Ω^- and Ω^+ need not each be connected, i.e., each may possess holes or multiple disjoint (and possibly nested) components. Thus, in two dimensions, I consists of one or more closed curves, and in three dimensions, I consists of one or more closed surfaces (Figure 2). We will usually refer to the domains Ω^- and Ω^+ as *phases* or *materials*, interchangeably, and use *region* to mean a particular component or patch of a domain or interface (in cases where the distinction matters).

It is often useful to consider the signed distance function $\phi(\mathbf{x})$ of the interface, for $\mathbf{x} \in \mathbb{R}^n$, which satisfies the following properties. The zero level set of ϕ coincides with I, that is,

$$\phi(x) = 0, \quad \forall \mathbf{x} \in I. \tag{2}$$

The magnitude of ϕ is the Euclidean distance to the interface,

$$|\phi(\mathbf{x})| = \|\mathbf{x} - \mathrm{cp}_I(\mathbf{x})\|, \tag{3}$$

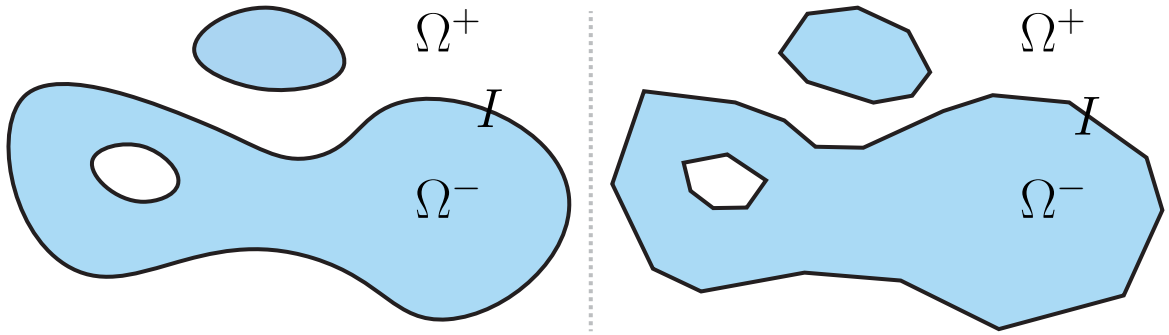

Figure 2: Left: A smooth interface I (black) in 2D dividing space into an interior Ω^- (cyan) and an exterior Ω^+ (white). Each has multiple components. Right: A coarse discrete polygonal approximation of the same interface in 2D.

where $\text{cp}_I(\mathbf{x})$ returns the closest point to $\mathbf{x}$ contained in I,

$$\text{cp}_I(\mathbf{x}) = \underset{\mathbf{y} \in I}{\arg\min} \|\mathbf{x} - \mathbf{y}\|. \tag{4}$$

By convention the sign of $\phi(\mathbf{x})$ is defined to be positive for $\mathbf{x} \in \Omega^+$ and negative for $\mathbf{x} \in \Omega^-$. This representation forms the basis of level set methods (Sethian et al., 1999; Osher and Fedkiw, 2003).

A corresponding indicator function can be defined as

$$\mathbf{1}_{\Omega^-}(\mathbf{x}) = \begin{cases} 1, & \phi(\mathbf{x}) \leq 0, \\ 0, & \phi(\mathbf{x}) > 0, \end{cases} \tag{5}$$

where the interface itself has been arbitrarily assigned the value 1 for convenience. This function has connections to phase field methods (Boettinger et al., 2002; Sun and Beckermann, 2007) and to the color fields or volume fraction fields often used in volume of fluid (VOF) methods (Hirt and Nichols, 1981; Rider and Kothe, 1998).

The motion of points $\mathbf{x}$ on the interface is assumed to be driven by a velocity field $\mathbf{V}$ according to an ordinary differential equation,

$$\frac{d\mathbf{x}}{dt} = \mathbf{V}(\mathbf{x}). \tag{6}$$

In a physical setting, such as fluid dynamics, $\mathbf{V}$ may be given by the flow velocities. In a geometric setting, such as mean curvature flow, $\mathbf{V}$ may be a function of the interface geometry itself (and hence may be well-defined only directly on the interface). In other settings, $\mathbf{V}$ may be determined by the gradient of some energy functional or defined mathematically or specified procedurally by an artist, possibly supported by 3D modeling tools. It is often convenient to design interface tracking software to be relatively agnostic to the specifics of $\mathbf{V}$; unless otherwise stated, our discussion will therefore treat $\mathbf{V}$ as a user-provided black box function, with the expectation that topology changes will be applied whenever interfaces either collide or come within some distance tolerance, depending on the particular method.

We emphasize that such automatic and purely geometry-induced topological change is not universally appropriate, as other authors have observed (Zhang and Fogelson, 2014; Chirco et al., 2022); fine-scale physical processes may act on certain types of interfaces to hasten, delay, or altogether prevent topology change when they enter into close proximity.

The method of Rajkotwala et al. (2018), for example, applies a film drainage model (Zhang and Law, 2011) to predict whether bouncing or coalescence should occur when two droplets collide.

2.2. Mesh Representation

Explicit interface tracking in three dimensions most often employs a triangle mesh data structure to discretize the interface, although some authors have proposed using higher-order surfaces, such as cubic splines or NURBS (Elgeti et al., 2012; Zhang, 2018). The connectivity of a triangle mesh can be mathematically represented using a graph structure as follows. The mesh consists of a set of vertices,

$$\mathcal{V} = \{v_1, v_2, ..., v_V\} \tag{7}$$

and a set of triangular faces (triangles), described by vertex triplets,

$$\mathcal{F} = \{f_1, f_2, ..., f_F\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}, \tag{8}$$

where we have adopted the notation of Botsch et al. (2010). Also useful is the set of edges, where each edge is defined by a pair of vertices appearing in (at least) one of the triangles:

$$\mathcal{E} = \{e_1, e_2, ..., e_E\}, \quad e_i \in \mathcal{V} \times \mathcal{V}. \tag{9}$$

These ideas can naturally be generalized to polygonal surface meshes wherein faces consist of three *or more* vertices. However, polygonal meshes are less common than triangle meshes in interface tracking, since polygonal faces are not guaranteed to be planar.

To meaningfully approximate a three-dimensional interface, the above graph structure must be immersed into $\mathbb{R}^3$ by assigning to each vertex v_i positional coordinates $\mathbf{p}_i = [x_i, y_i, z_i]^\intercal$; see Figure 3, left. The set of these vertex positions is $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_V\}$. Some authors refer to surface vertices as *markers*, in the sense that they are passive marker points following the flow, much like the particles of the classic marker-and-cell method (Harlow et al., 1965). Any point within a triangle can be described by linear (barycentric) interpolation of the triangle's vertex coordinates. The graph structure describing a mesh can be expressed computationally using a variety of data structures, the choice of which heavily impacts the efficiency of storage, traversal, querying, and modifications (Botsch et al., 2010). We will not delve into these details, keeping our discussion agnostic to these underlying implementation choices. In two dimensional settings the interface is similarly represented by vertices connected by straight edges into one or more closed polygonal curves, leading to simpler but generally analogous concepts and algorithms. We will often emphasize the three dimensional setting, but will also make frequent use of two-dimensional examples to illustrate some concepts.

A few key properties of a triangle mesh will be important for understanding topology change in interface tracking. A surface in three dimensions is described as *(2-)manifold* (without boundary) if each point on it possesses a surrounding neighborhood that is homeomorphic to a disk; that is, loosely speaking, it behaves locally like $\mathbb{R}^2$. This stipulation rules out surfaces that self-intersect, branch, or possess punctures. For triangle meshes, manifoldness implies each edge must be shared by exactly two incident triangles, each vertex must be surrounded by a closed loop of incident triangles sharing that vertex with no gaps, and coincident triangles are not allowed. Violations of the 2-manifold

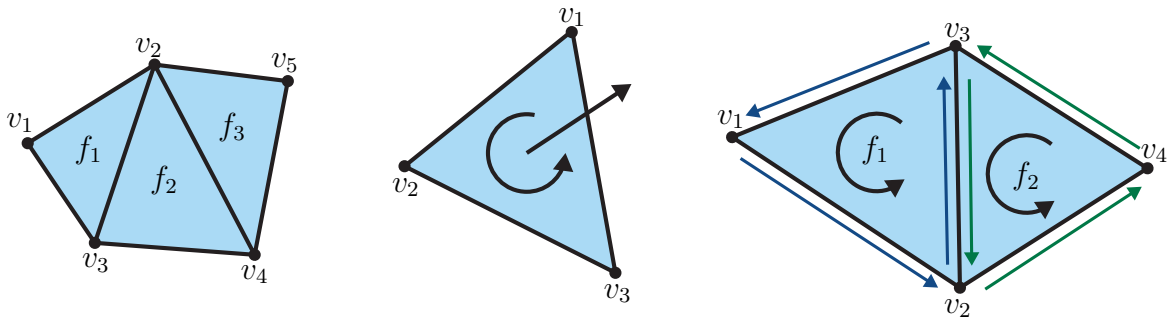

Figure 3: Left: A patch of a triangle mesh consisting of three triangles, $f_1 = (v_1, v_3, v_2)$, $f_2 = (v_2, v_3, v_4)$, and $f_3 = (v_2, v_4, v_5)$. Center: The orientation and normal vector of a triangle are determined by the ordering of its vertices. Right: Triangle pairs sharing an edge in a valid (manifold) mesh must be consistently oriented. In this instance, this condition implies that the pair of vertices $\{v_2, v_3\}$ for the shared edge must appear in the opposite order in the (cyclical) vertex ordering of the two triangles $f_1 = (v_1, v_2, v_3)$ and $f_2 = (v_3, v_2, v_4)$.

property are called non-manifold (or sometimes singular) geometries. Typical examples of manifold and non-manifold mesh geometry are shown in Figure 4. The discrete stipulations above focus on the mesh connectivity alone, i.e., whether the mesh is *intrinsically* (non-)manifold. Non-manifoldness can also arise extrinsically, when the mesh is immersed into $\mathbb{R}^n$ through the assignment of vertex coordinates: interface patches that have flowed into an intersecting state are non-manifold, as the interface is no longer properly embedded in $\mathbb{R}^n$. However, our use of the term manifold will generally refer to the intrinsic meaning.

Another important concept is *watertightness*. A triangle mesh is considered watertight if it fully divides the space into disjoint interior and exterior regions separated by the mesh, consistent with our definitions in (1) and the intuition that interfaces are borders between closed, non-overlapping volumes. Put another way, any path from the interior to the exterior of a watertight mesh must cross through a point on the mesh itself. Watertightness is therefore a critical requirement for nearly all explicit interface tracking methods we consider. A clear exception is if one wishes to instead consider interfaces with holes or borders, such as a soap film spanning a wire loop, where the notion of interior and exterior is no longer well-defined. Importantly, a triangle mesh can be simultaneously non-manifold *and* watertight: an interface consisting of two tetrahedra sharing only a single non-manifold vertex (like the one in Figure 4, lower left) remains watertight.

The surface (unit) normal vector of a triangle is dictated by the ordering and positions of its three vertices and a right-hand-rule convention: $\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)/||(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)||$ for a triangle with vertices ordered as (v_0, v_1, v_2) (Figure 3, center). The triangles of a valid mesh are also typically expected to be consistently oriented, such that all triangle normals point "outward" from the interface (i.e., from the interior Ω^- towards the exterior Ω^+); locally consistent orientation implies that an edge sharing two triangles has opposing orientation within each triangle (Figure 3, right). For a more in-depth discussion of the topological properties of triangle mesh surfaces, we refer the reader to the recent work of Bærentzen (2025).

While many interfaces are predominantly manifold, some of the methods we will discuss require support for meshes with non-manifold features, including two interfaces connected by a single vertex, an edge shared by more than two triangles, or a coincident triangle pair (Figure 4, bottom); however, they must still be properly oriented and watertight. In addition, as we discuss in §6, support for multimaterial generalizations of

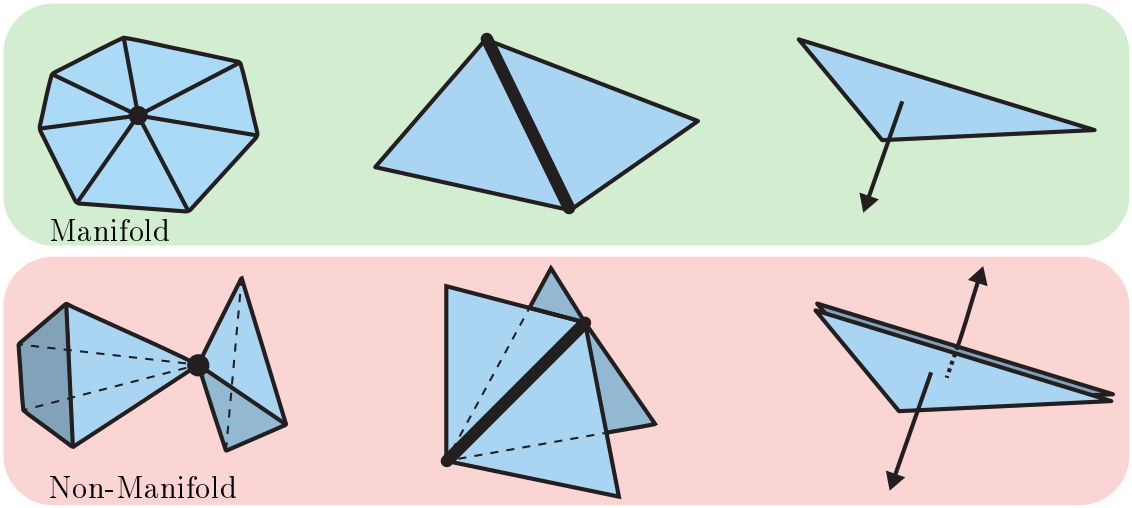

Figure 4: Examples of manifold and non-manifold configurations in triangle mesh geometry. Top, from left to right: a manifold vertex, a manifold edge, and a manifold triangle. Bottom: A non-manifold vertex (shared by two surface patches), a non-manifold edge (shared by more than two triangles), and a non-manifold pair of exactly coincident triangles.

explicit interface tracking inherently necessitates non-manifoldness, for example to allow three or more phases or regions to be in mutual contact along a curve or at a point.

2.3. Explicit Interface Tracking: Classification and Terminology

Most explicit interface tracking methods deform the interface by simply advecting points on it forward through time in a Lagrangian fashion, i.e., numerically integrating the ODE (6) for each vertex of the interface mesh, using for example the forward Euler method, Runge-Kutta schemes, etc. There are exceptions: *face offsetting* (Jiao, 2007) determines updated vertex positions indirectly from the motions in the normal direction of the incident triangular faces (in turn derived from $\mathbf{V}$), while another recent scheme projects out the tangential components of $\mathbf{V}$ to reduce vertex clustering (Gorges et al., 2023). Regardless, the advection process yields a new interface configuration; depending on the mesh resolution, the time step size, and $\mathbf{V}$ itself, this new configuration can possess interfaces in close mutual proximity or that have penetrated one another, indicating the need for topological change. Advection can also reduce the quality of mesh elements, necessitating an additional remeshing process in some schemes. Thus, explicit interface tracking algorithms can be thought of as executing three high-level stages at each time step: advecting the mesh, improving its geometric quality, and correcting its topology. (In some methods, the latter two stages are closely integrated.) It is the topology change treatment that is the focus of the present survey.

There are many different explicit, implicit, and hybrid interface tracking (or capturing) methods, so we will use the above description to characterize explicit interface tracking methods and thereby choose which schemes fall within the scope of our review. Specifically, we will consider methods that (1) employ an explicit mesh of the interface, including knowledge of its triangles (i.e., not just isolated vertices or points), and (2) advance the mesh configuration through some type of forward advection or time integration process applied to mesh vertices. This choice intentionally excludes schemes that advect (only) disconnected interfacial point samples (Szeliski et al., 1993; Torres and Brackbill, 2000), as well as various volume of fluid, phase field, level set, and or other implicit or

hybrid methods that may (perhaps optionally) construct an explicit mesh during every step, but perform advection through other means than forward vertex transport. There remains some subtlety in our definition: in some schemes, connectivity *between* adjacent triangles is not stored (and therefore "shared" vertices are computationally duplicated), or the triangle adjacency relationships may be known only indirectly from a secondary grid or mesh structure. We will include such schemes, provided they still correspond to some consistent, well-defined triangulation.

We have adopted the term *explicit interface tracking* because we believe it is relatively clear and unambiguous, but the reader should note that it is not universally adopted across disciplines. Within computational fluid dynamics, the more concise term "front tracking" is often used (Unverdi and Tryggvason, 1992), although some authors implicitly use "front tracking" to refer to the combination of explicit interface tracking with an Eulerian solver for the underlying fluid physics. The specific process of resolving topological changes or errors is also sometimes termed "untangling" (Glimm et al., 2000) (although in common parlance, untangling something usually does not imply modifying its topology). Computer graphics researchers use the term "mesh-based surface tracking" (Wojtan et al., 2011) in place of explicit interface tracking. Unfortunately surface tracking is an overloaded term: by analogy with object tracking, computer vision researchers have used surface tracking to mean reconstructing a deforming surface from sequences of imaging data and estimating correspondences between surface points through time (Ramey et al., 2004; Wang et al., 2019). To further complicate matters, surface tracking in this computer vision sense has also been performed *using* explicit interface tracking (Varanasi et al., 2008). In image processing and segmentation, especially medical imaging, two-dimensional methods fitting our definition of explicit interface tracking (called "active contour models" or "snakes") have been augmented to support topology change, leading to "T-snakes" (Kass et al., 1988; McInerney and Terzopoulos, 1995); in three dimensions, terms have included "topologically adaptive deformable surfaces" or "T-surfaces" (McInerney and Terzopoulos, 1999). In early computational methods for photolithography, explicit interface tracking methods are referred to as "ray-string methods", since the two-dimensional case involves a connected string of vertices, where at each timestep a given vertex moves some distance along a ray (Toh, 1990). They use the term "delooping" (rather than untangling) to describe resolving topological errors, since erroneous loop formation (in two dimensions) is a common error arising in that domain. In the context of foam and crystal grain dynamics in materials science and cellular dynamics in biology, the term "vertex models" (Lazar et al., 2011) is popular for explicit interface tracking-like schemes (to distinguish them from cell-based models in those domains). However, some authors use "vertex model" to refer more specifically to schemes that store and evolve only the non-manifold junction vertices where three or more materials meet (with interfaces assumed to be single arcs, lines, planes, or curved patches in between) and reserve "front tracking" for methods where the manifold interfaces are subdivided with additional vertices (e.g., allowing for more flexibility in the interface shape).

2.4. Interface Remeshing

In addition to causing topological changes, significant deformation of triangulated interfaces quickly leads to triangles becoming too large or distorted (poor aspect ratio) to ensure sufficient accuracy or too small and numerous for a simulation to efficiently continue. These situations can be treated through interface remeshing (also known as

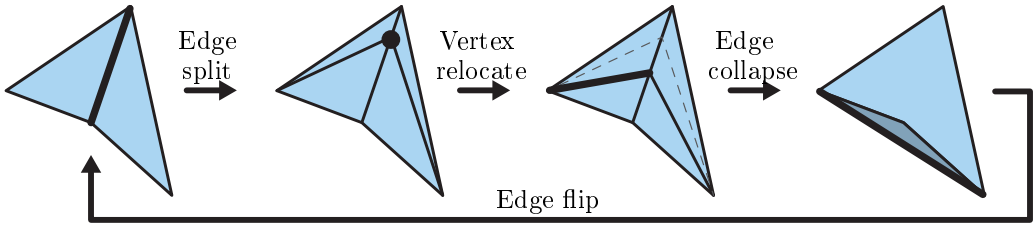

Figure 5: Four standard local triangle remeshing operations. The element (vertex or edge) to be operated on in the next operation is shown thickened.

mesh improvement or adaptation), where the goal is to improve the triangle mesh density and quality, according to some metric(s), while minimizing changes to its extrinsic shape and topology. Triangle surface remeshing (in the absence of topology change) is a major research challenge in its own right, so we will not cover it extensively; surveys by Alliez et al. (2008), Khan et al. (2020), and Panchal and Jayaswal (2021) provide excellent background.

However, we will briefly summarize the most common set of local (or incremental) remeshing operations, diagrammed in Figure 5, as they are shared by many interface tracking codes (Glimm et al., 2000; Botsch and Kobbelt, 2004; Brochu and Bridson, 2009; Botsch et al., 2010; Dunyach et al., 2013; Regnault, 2023). *Vertex smoothing* (or vertex relocation) involves moving vertices to improve the quality of each vertex's set of surrounding triangles, often using variants of Laplacian smoothing (i.e., a diffusion-like operation applied to vertex coordinates). An *edge collapse* shrinks an edge to zero length so it can be deleted, together with the two incident triangles, reducing the local mesh resolution. An *edge split* divides an edge into two by adding a new vertex, which introduces two new triangles incident to the edge and thus increases the local mesh resolution. Lastly, an *edge flip* replaces an adjacent pair of triangles covering the same quadrilateral patch in such a way that the shared diagonal edge is replaced with its opposite diagonal; this operation can be used to improve mesh angles or regularize vertex valences. Brochu and Bridson (2009) discuss how to perform such operations while guaranteeing no (self-)intersections are introduced; others have considered how to perform remeshing while preserving volume (Regnault, 2023; Gorges et al., 2022; Jiao, 2006) or sharp features (e.g., peaks and creases) (Vorsatz et al., 2003; Botsch and Kobbelt, 2004; Jiao, 2007).

2.5. Meshing of Implicit Surfaces

Given an implicit representation of an interface, one often needs to (re)construct a corresponding explicit mesh. This *isosurfacing* operation (also sometimes called isocontouring, polygonization, or mesh reconstruction) is classically performed by meshing a particular level set (isosurface) of the implicit function (e.g., all points $\mathbf{x}$ where $\phi(\mathbf{x}) = 0$ for a signed distance field or $\mathbf{1}^{\varepsilon}_{\Omega^-}(\mathbf{x}) = 0.5$ for a smoothed indicator function with interface thickness ε). Where such methods find application in explicit interface tracking is in implicit topology change mechanisms (§3): some or all of a mesh is converted to an implicit representation, the implicit data is isosurfaced, and the resulting mesh (or mesh patch), now free of topological errors, can be used to continue the interface evolution.

The most popular classic isosurfacing algorithm is marching cubes (Lorensen and Cline, 1987; Wyvill et al., 1986). By labeling the eight corners of a cubic grid cell as interior or exterior based on the implicit surface value and designing a corresponding triangulated surface patch for every possible configuration (exploiting symmetries

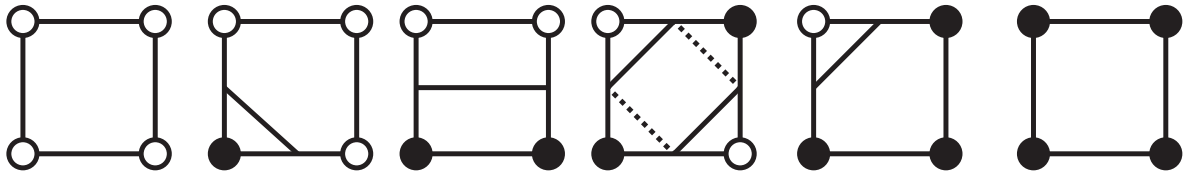

Figure 6: The marching squares (2D) cases determine an appropriate interface patch (or patches) per cell to approximate an isosurface of a scalar field stored at grid nodes. Black nodes are interior and white nodes are exterior. All other cases can be produced through rotations or reflections. (Dashed lines show a case where the correct topology is ambiguous.)

to eliminate redundancy), an entire mesh can be assembled by marching over a grid of implicit data and applying the appropriate case to every cell. The new vertices for the patches lie on grid edges where a transition from interior to exterior occurred, with the new vertex positions along the edge determined by interpolation or root-finding on the implicit data. We will refer to these points as *edge-crossings*. The cases for the analogous marching squares algorithm in two dimensions are illustrated in Figure 6. Provided that the sampled implicit data is sufficiently well-resolved, this approach yields a mesh with the correct topology. For undersampled implicit data, a valid closed surface will be generated, but its topology may not match the underlying implicit surface. An important modification is the marching tetrahedra algorithm (Doi and Koide, 1991), which uses a tetrahedrization of space rather than a Cartesian grid, offering greater flexibility and fewer cases to consider at the cost of higher triangle counts.

Isosurfacing has been deeply investigated with many improvements and alternative approaches having been suggested; we refer the reader to a useful survey of this topic (De Araújo et al., 2015). As a notable example, dual contouring is known to better recover fine details and sharp features, but requires that additional gradient/normal information be available (Ju et al., 2002). Among explicit interface tracking methods that rely on isosurfacing (i.e., handle topology change implicitly), most use variations of classical marching cubes or marching tetrahedra.

When the implicit representation consists of per-cell volumes, such as in volume-tracking schemes (§3.6), it is often natural to try to build an explicit geometry that precisely contains the target cell volumes, rather than naively isosurfacing the interpolated zero isocontour. These approaches find a line (2D), plane (3D), or mesh patch per cell whose enclosed volume precisely matches the prescribed cell volumes while encouraging consistency with neighboring cells and minimizing the presence of isolated fragments. Because they prioritize recovering the volumes, such methods do not necessarily ensure continuity of the interface between neighboring cells (Figure 7), which can introduce challenges for domains where smooth surface geometry is important (e.g., surface tension effects, visual applications, etc.).

Classical volume reconstruction approaches (illustrated in Figure 7) are the Simple Linear Interface Calculation (SLIC) method (Noh and Woodward, 1976) which assumes an axis-aligned interface per cell, and the more accurate Piecewise Linear Interface Calculation (PLIC) (DeBar, 1974; Youngs, 1982, 1984), which represents the cell's interface with a general line or plane per cell. In the latter case, given an estimated normal direction, clipping the line/plane against the grid cell determines the contained volume; the plane's offset in the normal direction can be adjusted until the correct volume is achieved. Scardovelli and Zaleski (2000) showed how to determine this offset value ana-

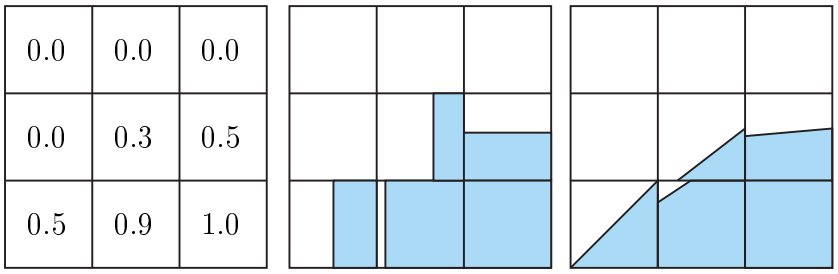

Figure 7: Left: A small grid containing per-cell volume fraction data. Center: A possible reconstruction using a Simple Linear Interface Calculation (SLIC), which uses axis-aligned lines for the interface. Right: A possible reconstruction using a Piecewise Linear Interface Calculation (PLIC). Note that the interface is not necessarily continuous between cells.

lytically. Numerous subsequent advancements and variations on this approach have also been proposed; Marić et al. (2020) provide a useful summary.

2.6. Types of Topological Changes

Let us consider the types of topological changes that can occur naturally during interface tracking under a flow $\mathbf{V}$. The process of two locally disconnected patches of the interface I joining together is variously referred to as merging or coalescence, depending on the discipline or authors. Conversely, one connected patch of I dividing into two is referred to as splitting, separation, breakup, rupture, or pinch-off. These two operations are shown in two dimensions in the top rows of Figure 8. For consistency, we will only use the terms merging and splitting going forward. Notably, such operations do not necessarily change the *global* number of disjoint components, e.g., a three-dimensional torus can undergo splitting while remaining a single connected component.

Merging and splitting are inverses of each other in the sense that, topologically, a split can undo a merge and vice versa. However, as highlighted by Lachaud and Montanvert (1996), merging and splitting also exhibit duality relationships, because the interior and exterior domains do not geometrically differ in any fundamental way. For example, two ballistic liquid droplets immersed in air can merge together, and so too can two air bubbles immersed in liquid. From a topological perspective these scenarios differ only in the material labeling of the volumes.

In two dimensions, this duality means that merging of Ω^- occurs through a (dual) splitting of Ω^+, and vice versa, as can be observed in Figure 8: when the blue material merges, the white region simultaneously splits. Two-dimensional merging and splitting operations can therefore be considered a single operation, and numerical implementations can exploit this fact.

In three dimensions, however, the extra dimension causes merging and splitting to become two fully distinct operations (Figure 9). Merging will occur when there is a thin layer of Ω^+ (or in the dual case, Ω^-) sandwiched between two separate regions of Ω^- (resp. Ω^+), i.e., two close interfaces. Merging causes the outer material regions to locally join together by effectively puncturing through the thin middle layer, which entails the two initially (locally) disjoint interfaces becoming connected. Notice that, unlike in 2D, this merging process does *not* imply that the complementary region (middle layer) has undergone splitting into two disjoint regions—it simply now has a hole through it.

Three dimensional splitting is the inverse operation. Initially a slender tube of material

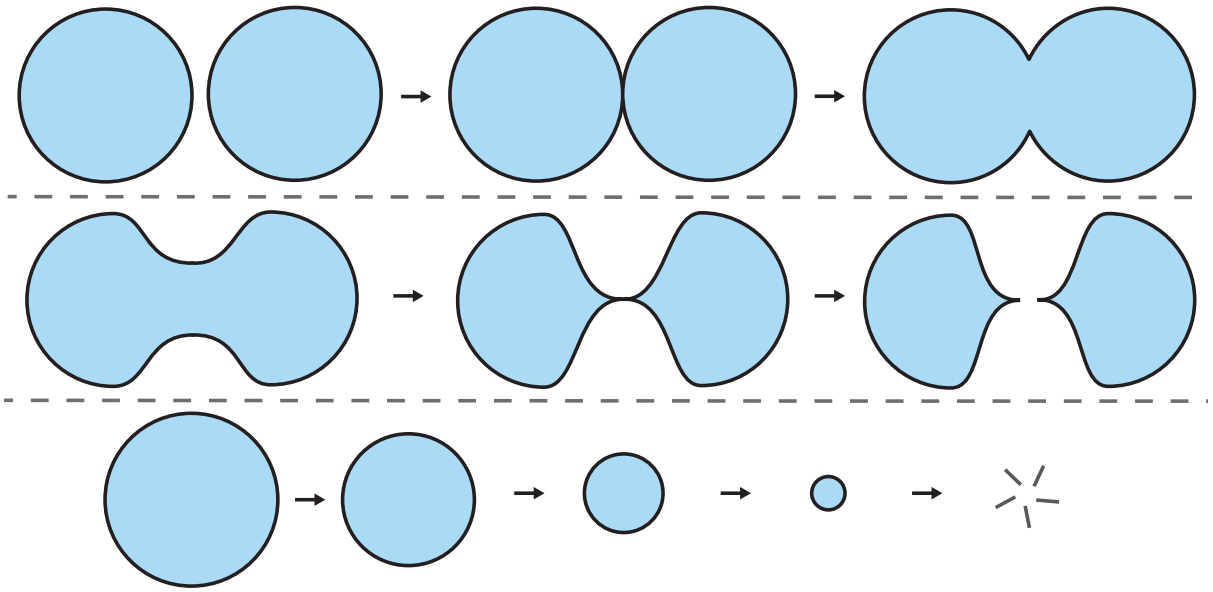

Figure 8: Topology change types in 2D, with time ordered from left to right. Top: Merging of Ω^- (i.e., the blue regions). Middle: Splitting of Ω^-. Bottom: Deletion of a region of Ω^-.

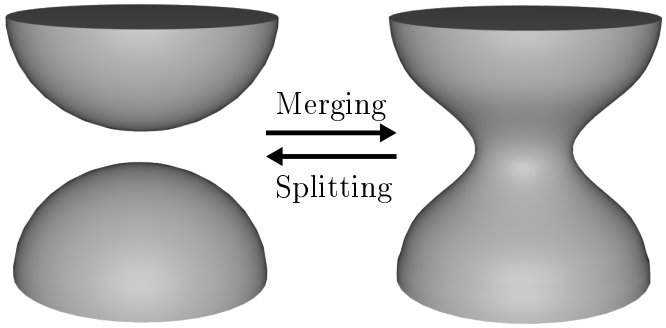

Figure 9: Merging and splitting in 3D, where the interface shown in gray contains the interior, Ω^-, with the surrounding exterior Ω^+ in white. Swapping interior Ω^- and exterior Ω^+ labelings gives the dual merging and splitting operations.

Ω^- (resp. Ω^+) is surrounded on all sides by its complement Ω^+ (resp. Ω^-). The tube then narrows to a point and the two sides separate. This splitting process likewise does *not* imply that the surrounding material has undergone a merging of two previously disconnected regions—it consisted of only one connected region to begin with.

Thus, if one considers the local behavior of the interfaces near such a three-dimensional topological change, the relevant geometric operations are two interfaces connecting (i.e., a merge) or a single interface disconnecting (i.e., a split), irrespective of which material is labeled as interior or exterior. (In the context of computational fluid dynamics, Tryggvason et al. (2001) similarly describe the two cases as "films that rupture" and "threads that break".) The array of explicit interface tracking methods we discuss differ principally in how they execute these two operations on the triangulated interface.

Another pair of comparatively simple (and less commonly discussed) topological changes are the creation and deletion of small components (Lachaud and Montanvert, 1996). Component creation corresponds to inserting a minimal new volume of material, e.g., as a single tetrahedron. Creation cannot occur on its own through the deformation of

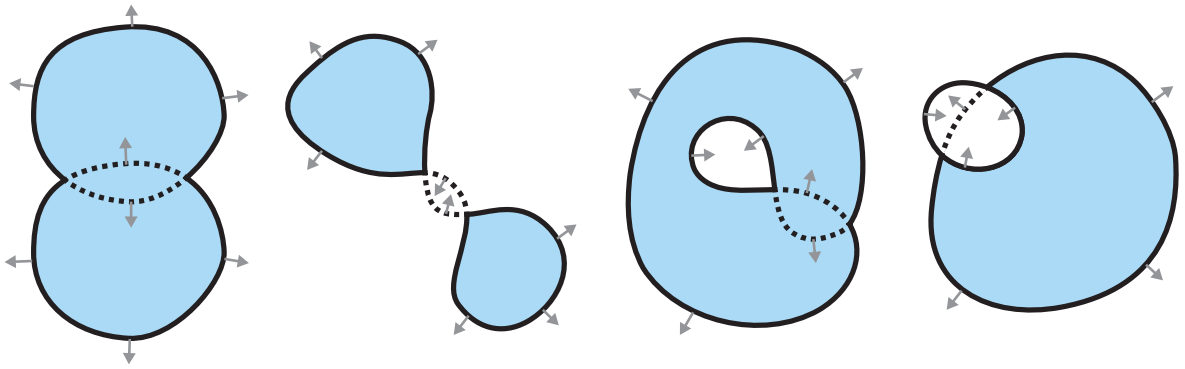

Figure 10: An appropriate point-in-polygon test (and associated winding number rule) should be able to disambiguate interior from exterior even in scenarios with (self-)intersections. The interior coloring of the shapes indicates the desired result. From left to right: merging of two colliding regions; splitting of one region due to self-intersection in a narrow region; merging of one self-colliding region; and loss of a hole, due to an interior void breaching the surface of its containing region. Grey arrows indicate the outward normals of the (input) surfaces.

existing surfaces, so it must instead be purposefully initiated through some user-defined physical or geometric mechanism, e.g., the seeding of bubbles in a multiphase flow (Patkar et al., 2013) or the insertion of isolated holes in a topology optimization task (Christiansen et al., 2014). We therefore do not consider this type of change in our review. By contrast, component deletion (Figure 8, third row) does arise naturally in convergent flows, such as in the limit of a spherical volume shrinking under mean curvature flow. Component deletion can be handled by simply removing any isolated tetrahedron that inverts or falls below some sizing threshold.

For some application domains, interfaces can also change topology in other ways besides continuous motion or deformation. For example, the instantaneous or incremental creation of sharp cuts inside or through a material can clearly change its topology. Because they do not arise naturally under a flow $\mathbf{V}$, we omit such operations from our review. Nevertheless, such cutting can be useful in computational fracture modeling (Chiaruttini et al., 2013), animations of arbitrary cutting effects (Sifakis et al., 2007), or virtual surgery simulation (Mor and Kanade, 2000), among other applications.

In summary, the complete set of topology operations we consider in the two-phase setting consists of merging, splitting, and deletion. Additional important types of topology change arise in interface tracking when one considers multimaterial problems involving three or more materials, but we defer discussion of this issue to §6.

2.7. Distinguishing Interior and Exterior

It is often necessary in interface tracking to carefully distinguish between interior and exterior regions in a mathematically consistent fashion. That is, we must be able to robustly answer point-in-polygon or point-in-polyhedron queries, a classical problem in computational geometry and computer graphics (O'Rourke, 1998; Hormann and Agathos, 2001). Furthermore, for some algorithms we discuss, this procedure must also properly handle configurations in which interfaces have become (self-)intersected, as illustrated in Figure 10. This challenge can be naturally addressed through the concept of the winding number, w, which measures how many times an interface "wraps around" a given point, as nicely summarized (in the interface tracking context) by Zaharescu et al. (2010). In

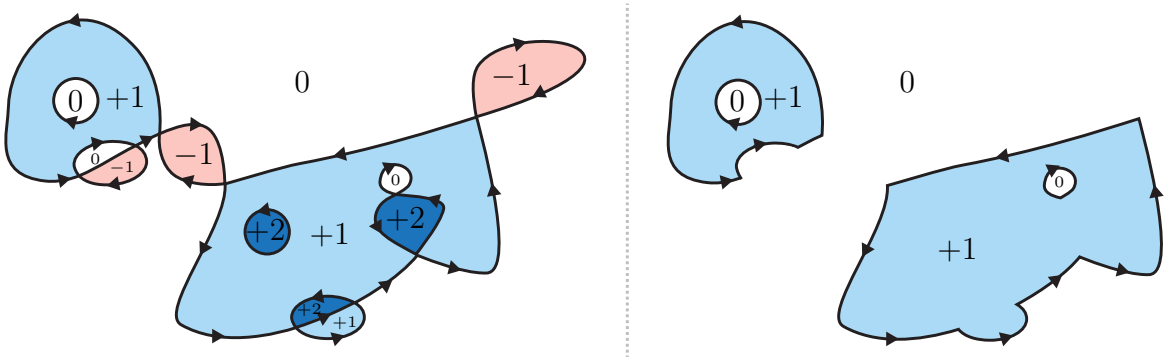

Figure 11: An illustration of the winding number for a set of intersecting and self-intersecting curves. Arrows indicate the curve orientation. Left: An initial self-intersecting set of curves. Right: The same set of curves after applying the positive winding number rule to resolve inconsistencies and achieve a valid state.

2D, assuming (without loss of generality) that the query point lies at the origin, $\mathbf{p} = 0$, we evaluate the winding number by integrating the total angle the interface curve makes around the point as

$$w_{2D}(\mathbf{p}) = \frac{1}{2\pi} \int_I d\theta, \tag{10}$$

which can be discretized naturally for polygons. In 3D, we similarly integrate the differential solid angle $d\Omega$ over the interface using

$$w_{3D}(\mathbf{p}) = \frac{1}{4\pi} \iint_I d\Omega, \tag{11}$$

using spherical coordinates; this expression can likewise be straightforwardly discretized for triangles (Jacobson et al., 2013). For closed surfaces (i.e., those of greatest relevance to interface tracking), the result will be an integer. The winding number is illustrated for a complex shape in two dimensions in Figure 11 (left).

Equivalently, the winding number can be computed using raycasting. One shoots a ray in any direction from the query point, and, for every intersection of the ray with a mesh triangle, tests whether the ray vector and triangle normal have the same direction (the 2D analog is illustrated in Figure 12). Assuming by convention that the mesh has outward oriented normals, a positive dot-product implies the ray is leaving Ω^-, and a negative dot-product implies it is entering. By incrementing a counter for every crossing (-1 for entry, $+1$ for exit), one recovers the winding number. (One can alternatively cast a ray from far *outside* the domain *to* the query point to compute the same result, adjusting the orientations and counting appropriately.) When this raytracing approach is computed under floating-point arithmetic, small errors or inconsistencies can lead to an inaccurate calculation of the winding number (e.g., when a ray passes exactly through an edge or a vertex, or tangent to an edge or triangle), so it should be treated with care (Woop et al., 2013).

Given the notion of the winding number for closed regions, we distinguish interior from exterior by choosing an appropriate labeling rule. For interface tracking, we consider regions with positive winding number to belong to Ω^- and regions with nonpositive winding number to belong to Ω^+. This rule can alternatively be viewed as relabeling (or normalizing) all negative regions to $w = 0$ (exterior) and all positive regions to

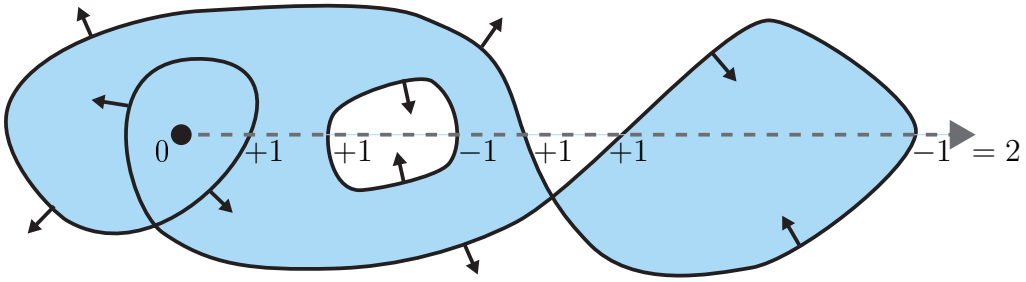

Figure 12: The winding number for a query point $\mathbf{p}$ (black disk) can also be computed by casting a ray (gray dashed arrow) in any direction and tallying its oriented crossings of the surface starting from zero: $+1$ if the surface normal and ray direction have a positive dot product, otherwise -1. The net value at infinity is the winding number for the query point. In this case, $w(\mathbf{p}) = 0 + 1 + 1 - 1 + 1 + 1 - 1 = 2$, consistent with the point $\mathbf{p}$ being doubly nested inside the curve.

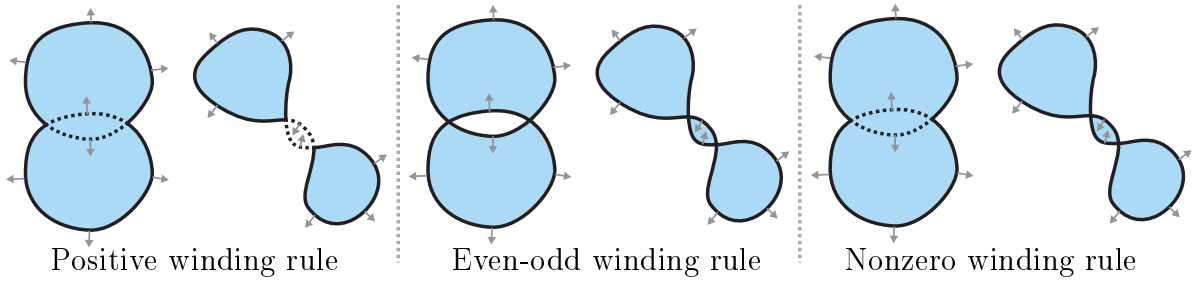

Figure 13: A comparison of common winding number rules for inside/outside determination of intersecting interfaces, applied to two of the basic merging and splitting cases from Figure 10. The cyan coloring indicates the regions labeled as interior. Dashed lines indicate curves deleted by the given rule. Both the even-odd (parity) rule and the nonzero rule erroneously fill in the inverted region ($w = -1$) during splitting. The even-odd rule additionally mislabels the doubly nested ($w = 2$) merging region as exterior. Only the positive winding rule yields the desired result.

$w = 1$ (interior), and preserving only those interfaces that represent transitions between zero and one. All other interfaces are considered redundant, and are therefore ignored or discarded; see Figure 11 (right). This rule leaves properly oriented interior holes untouched, treats overlapped interior regions as also inside (which occurs during collision-induced merging), and deletes negative regions where the material appears to have flipped "inside out" (which occurs during collision-induced splitting). Thus, this rule successfully distinguishes interior from exterior as desired, and yields a consistent way of resolving topological errors, like those in Figure 10. This approach is used by several of the methods we describe later.

Alternative rules, such as the nonzero winding number rule or the even-odd (parity) rule (Hormann and Agathos, 2001), are common in other applications in computational geometry or computer graphics, but do not play a useful role in explicit interface tracking. We illustrate this fact in Figure 13. Authors in geometry processing have also discussed a visibility-based labeling they call the outer hull (Campen and Kobbelt, 2010); however, this rule undesirably deletes interior holes, so we discard it.

A related concept is that of geometric Boolean operations, also known as Constructive Solid Geometry (CSG) (Laidlaw et al., 1986). Given two solid shapes, one can define a new shape using the union, intersection, or difference of those shapes. Much like our setting, these operations divide $\mathbb{R}^n$ into closed regions that are labeled as interior or exterior based on logical rules related to the winding number (Zhou et al., 2016). In

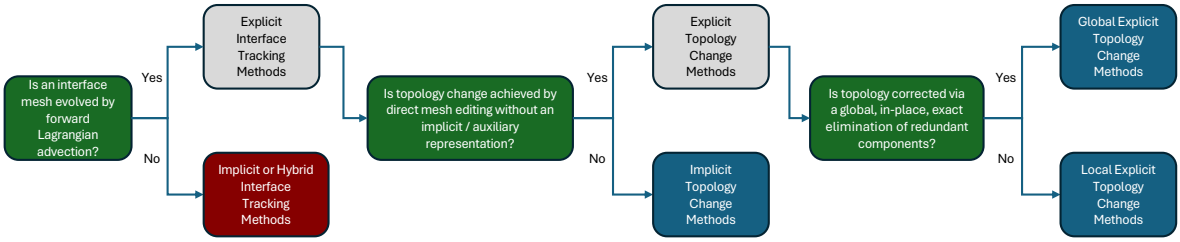

Figure 14: A flow chart summarizing our classification of interface tracking and topology change methods (§2.3 and §2.8). Our review considers only explicit interface tracking methods, ruling out implicit or other hybrid interface tracking methods (red box). Within explicit interface tracking methods, we focus on their methods for topology change (blue boxes), categorizing each as implicit, global explicit, or local explicit.

simple cases, such as two disjoint droplets colliding, a union operation can suffice to resolve intersections occurring during interface tracking. Generalizing to a notion of *self-union* (Zhou et al., 2016) can also allow proper resolution of shallowly self-intersecting shapes. However, union operations are insufficiently expressive for interface tracking; for example, they fail to support the inversions that arise during splitting (e.g., Figure 10, center left). Only the positive winding number rule offers the correct definition of interiority. On the other hand, triangle mesh-based CSG methods provide other useful techniques: they typically first require computing all triangle-triangle intersections and remeshing the surfaces such that all intersections are "resolved", that is, triangles only meet at shared edges or vertices. The same process, also known as *mesh co-refinement*, is critical to global explicit topology change techniques (§4), and care is required to perform it safely under floating-point arithmetic. We discuss some recent developments in CSG algorithms in §7.2.

2.8. General Strategies for Topology Change

We classify strategies for processing topology change in explicit interface tracking into three general families. First, the apparent ease with which standard implicit interface tracking methods (e.g., the level set method, volume of fluid, etc.) handle topological merging and splitting events has inspired many researchers to propose hybrid interface tracking schemes that combine aspects of explicit and implicit methods (Figure 15, top row). Among such hybrids, the methods relevant to the present review use a Lagrangian mesh for the base representation, surface advection, and potentially other operations, but temporarily convert the mesh to an implicit or auxiliary representation, such as an SDF, a smooth indicator function, or a collection of points or vertices, to resolve topological events. A reconstruction procedure is then applied to the resulting implicit data to reassemble a new valid explicit mesh, either locally or globally. We will refer to these approaches as *implicit topology change* mechanisms, and discuss this family in §3. Note that we draw a sharp distinction between these implicit *topology change methods* that are used within explicit interface tracking methods, and wholly implicit (or other hybrid) *interface tracking methods*, which do not use an explicit mesh to perform forward Lagrangian interface advection.

Implicit topology change mechanisms tend to offer efficiency and comparative simplicity, but they make sacrifices in accuracy, detail, or volume preservation through the inherent regularization induced by the representation conversion and mesh reconstruction procedures. On the other end of the spectrum are *explicit topology change methods*

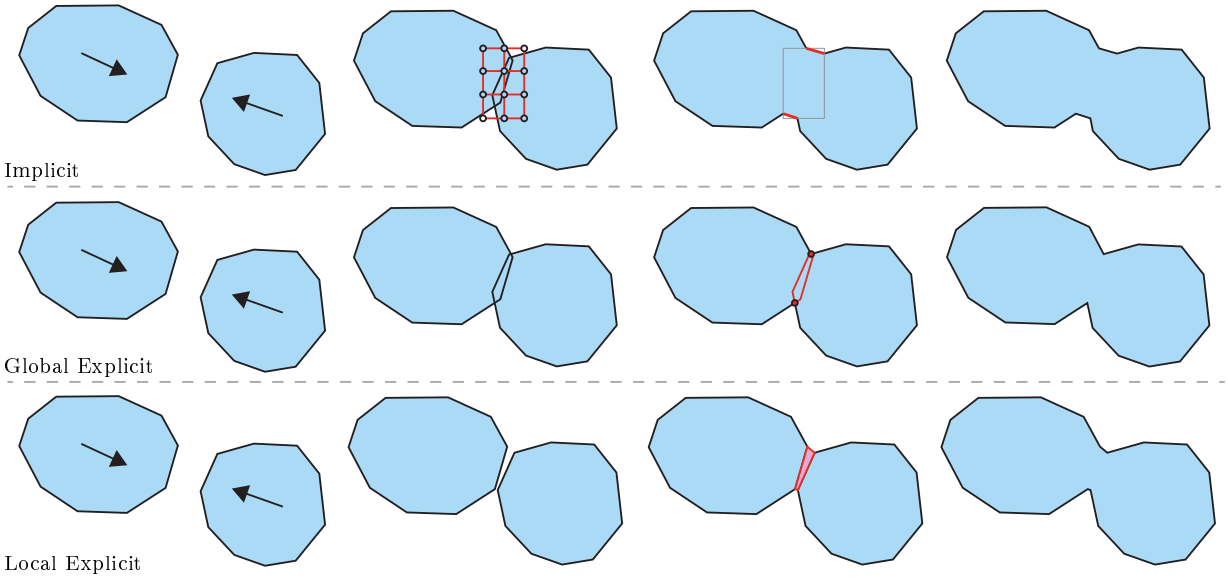

Figure 15: Prototypical instances of the three families of topology change algorithms, illustrated in 2D for the simple case of two colliding polygonal bodies. Top: Implicit methods convert regions undergoing topology change to an implicit representation, reconstruct a corresponding corrected mesh (or patch), and replace the old mesh geometry. Middle: Global explicit methods compute the precise intersections between overlapping mesh components, subdivide the intersecting elements in place, distinguish interior from exterior regions, and discard redundant interface components. Bottom: Local explicit methods detect proximate or mildly intersecting geometry and perform local modifications to connect (or disconnect) the relevant interfaces, which slightly perturb the surface position.

that do not make use of an intermediate implicit representation, instead preferring to process topological changes directly on the triangle mesh itself via mesh surgery. These approaches can introduce greater complexity (and attendant challenges in terms of robustness and efficiency), but a successful implementation can arguably provide higher geometric and topological accuracy and fidelity.

Explicit topology change approaches come in two flavors, representing the remaining two categories we consider. The first of these applies global, Boolean-like, in-place modifications to correct the mesh, so we refer to it as *global explicit topology change*. (Figure 15, middle row). These schemes require that topological changes be performed only once mesh triangles have interpenetrated into or crossed through one another to create an invalid mesh state that must be corrected. This correction procedure involves a classic computational geometry task of finding the precise intersection geometry among all triangles and mutually subdividing those triangles at their intersections, so that individual triangles meet one another only at shared vertices and edges (a.k.a., mesh co-refinement). The mesh configuration is then analyzed to remove the subset of mesh components that are redundant or unnecessary, as suggested by Figure 11. These methods are "in-place" in the sense that the actual surface being represented is not deformed or reshaped; it remains in position while topologically redundant mesh components are identified and eliminated, as can be observed in Figure 15, middle row, right three columns. We describe these methods as being global, because they seek to identify and resolve all topological errors simultaneously in a unified process that considers the global mesh configuration, through winding numbers or related concepts. We discuss this family in §4.

The last category we consider is *local explicit topology change* (Figure 15, bottom

row). Such methods incrementally perform individual local topological edits on the mesh when components of the mesh come close together or slightly intersect. The deployed operations are local in the sense that each one involves only small isolated patches of mesh elements and they do not consider global surface relationships; they are incremental in the sense that such edits are performed sequentially (at least conceptually) rather than in a single global process. Compared to the global methods above, local methods often have the disadvantage of mildly perturbing the shape of the mesh surface from its current state to carry out a merge or split; because they eschew global information, it can also be more challenging to make them robust to highly tangled configurations. On the other hand, local schemes do not necessarily require overlap to have occurred before they can process a topology change. As a result, they can support "preemptive" merging or splitting of narrow gaps or slender necks, based on a distance threshold, while the mesh is still in a topologically valid state. Furthermore, topology change can often also be more easily controlled (e.g., allowed or prevented) by incorporating additional criteria to decide whether to execute each local operation. We discuss this family in §5.

Naturally, there remain a small handful of methods that cannot be perfectly classified into exactly one of these categories, as they possess hybrid aspects. We group such methods into the (qualitatively) closest matching category, and comment on their classification.

2.9. Benefits of Explicit Interface Tracking

Traditionally, explicit interface tracking has been considered difficult to implement and prone to robustness issues, which raises an important question: if one can overcome these challenges, what value does it offer in return, as compared to implicit interface tracking methods that treat topology change more easily? Let us consider a few possibilities.

- **Interface Accuracy:** Explicit approaches directly integrate in time the positions of vertices (interface markers) in a Lagrangian manner, which allows for tracking the interface evolution with a high degree of accuracy. Whereas basic implicit methods rapidly smooth away sharp or thin features and sacrifice high frequency details, even for rigid motions, basic explicit schemes preserve fine-scale features well and are effective at tracking motions of many kinds, assuming an appropriate high-order integrator is applied to the vertex trajectories. The interface is always precisely and unambiguously represented, without the need for additional redistancing, sharpening, or mesh extraction.

- **Interface Data:** In many relevant problems, essential application-dependent quantities are associated with the interface itself. Examples include surfactants in some interfacial flows (Muradoglu and Tryggvason, 2008), colors or textures on sculpted 3D models (Stanculescu et al., 2013), vorticity or circulation in vortex-sheet methods (Stock et al., 2008; Da et al., 2015), velocity or potential in potential flow formulations (Keeler and Bridson, 2015; Da et al., 2016), boundary-layer data in coupled multi-scale simulations (Aboulhasanzadeh et al., 2012, 2013), and interface thickness in soap bubbles (Saye and Sethian, 2013; Ishida et al., 2020). As such, a wide array of numerical methods are designed to use boundary or interface data, such as boundary element methods (Sauter and Schwab, 2010) and surface finite element methods (Dziuk and Elliott, 2013). With explicit interface tracking, such data can be directly assigned to, advected along with, and evolved on the vertices, edges, or triangles of the Lagrangian mesh.

- **Control of Topology Change:** The need to intentionally process topological changes, while often cited as a drawback, can also be a benefit of explicit tracking for some applications. For example, not all materials merge instantly and seamlessly upon contact, and thus additional physical criteria or modeling may be desired. In interactive applications, a user may wish to manually dictate whether and where topology change occurs. Explicit tracking can more readily offer such control, whereas under purely implicit interface tracking topology change is almost always automatic and dictated numerically by the discretization resolution. This issue can impact a method's ability to converge to the desired physical solution, as recently highlighted and explored for a VOF method by Chirco et al. (2022).

- **Access to Geometric and Topological Properties:** The geometry of triangle meshes is well studied, with meshes offering fairly direct computation of surface areas, volumes, centroids, as well as important differential geometric quantities like surface normals and curvatures (Meyer et al., 2003; Jiao and Zha, 2008). The latter properties are essential for many geometric and physical flows, including normal flow, mean curvature flow, minimal surfaces, and surface tension effects. Likewise, many topological properties, such as Euler characteristic, genus, connected components, Betti numbers, etc. are comparatively straightforward to extract from interface meshes compared to implicit surfaces. (A recent review by Bærentzen (2025) discusses topological properties of triangle meshes.)

- **Ease of Spatial Adaptivity:** It is often valuable to be able to adapt the interface resolution so that it varies spatially, for example to focus memory and computational effort on regions deemed to have greater importance. Doing so with implicit interface tracking methods often requires more elaborate adaptive volumetric structures (octrees, unstructured tetrahedral meshes, etc.), whereas locally coarsening or refining a triangle mesh is a comparatively simple operation, e.g., using the operations mentioned in §2.4.

- **Compatibility with Existing Triangle Mesh Infrastructure:** Triangle meshes are a fundamental geometric representation that has been widely used for decades; as such, there exist many associated code libraries, file formats, computational techniques, and software tools that are tailor-made for generating, storing, manipulating, and visualizing them. This popularity means that explicit interface tracking can be readily integrated with many existing toolsets and application domains. Conversely, many methods and tools designed for triangle meshes can also be integrated into explicit interface tracking methods (e.g., for remeshing, measuring geometric quantities, computing intersections, etc.)

We stress that these are general trends. Purely implicit interface tracking methods offer many benefits of their own that should be weighed in choosing an interface tracking mechanism for a particular setting. Nevertheless, the benefits listed above should make clear why explicit interface tracking is worthy of consideration.

2.10. A Note on Robustness and Efficiency

Because it involves manipulating triangle meshes, explicit interface tracking necessarily relies on a variety of geometric constructions and predicates. A construction is the computation of new geometric elements, such as vertices, intersection points or segments,

triangulations, convex hulls, etc.; a predicate is a decision procedure or question, such as determining whether a point lies on, to the left, or to the right of a line or plane, or whether two segments intersect. A great deal of effort in computational geometry has been dedicated to developing accurate constructions and predicates, leading to libraries like the powerful Computational Geometry Algorithms Library (CGAL) (The CGAL Project, 2025). Ultimately, while some geometric problems can be solved exactly (e.g., using arbitrary precision rational arithmetic), in practice explicit interface tracking is performed under finite-precision floating-point arithmetic, which incurs error. Unfortunately, even if such errors are small in magnitude, they can nevertheless cause severe issues. For example, if raycasting is used to determine the interior/exterior status of a point via the winding number, and the ray passes exactly (or almost exactly) through a vertex of the triangle mesh, small floating-point errors can lead to a false positive or false negative (i.e., an intersection is detected when none occurred, or an intersection is missed when one did occur). Such errors can have disastrous downstream effects that corrupt the topology of the mesh interface, rendering the output entirely incorrect. In any implementation of explicit interface tracking, it is therefore critical to consider the effects of floating-point error to ensure robust geometric results. We will not attempt to review this important topic, as it represents a significant area of research in its own right. Our survey will instead describe topology change strategies in terms of their geometry and topology, but omit discussion of the finer details of a robust implementation. We refer the reader to relevant texts on robust geometric computing, drawn from computer graphics and computational geometry (Shewchuk, 1999; De Berg et al., 2008; Eberly, 2021).

Efficiency of implementation is another key concern for interface tracking methods. Beyond exploiting both parallelism and standard code optimization strategies, algorithmic and data structure aspects play an essential role. As a canonical example, detecting all proximities or intersections among a set of n triangles representing an interface is a costly $O(n^2)$ operation if implemented by naively performing all pairwise tests. This particular issue is well-studied in computer graphics, and is addressed by dividing collision-detection into two phases: a broad phase and a narrow phase. The goal of the broad phase is to quickly and cheaply cull away triangle pairs that obviously do not intersect, using simplified geometry (e.g., bounding boxes around triangles) and spatial acceleration structures (e.g., octrees or bounding volume hierarchies). This drastically lightens the computational load on the subsequent narrow phase, in which expensive triangle-triangle intersection tests are directly performed on the remaining candidate pairs. Another fundamental efficiency consideration is the choice of data structure used to store and manipulate the interface triangle mesh itself. We will not cover these types of issues, but instead refer the reader to the relevant computer graphics literature (Shirley et al., 2009; Ericson, 2004).

3. Implicit Topology Change

We begin with implicit topology change procedures. As we will see, these take many forms, but the most direct approaches advect a triangle mesh, convert it (locally or globally) to an implicit representation (often but not always stored on a grid), and perform a reconstruction procedure to build either a new mesh patch or an entirely new mesh. More fundamentally, the distinguishing feature of implicit approaches is that the topology

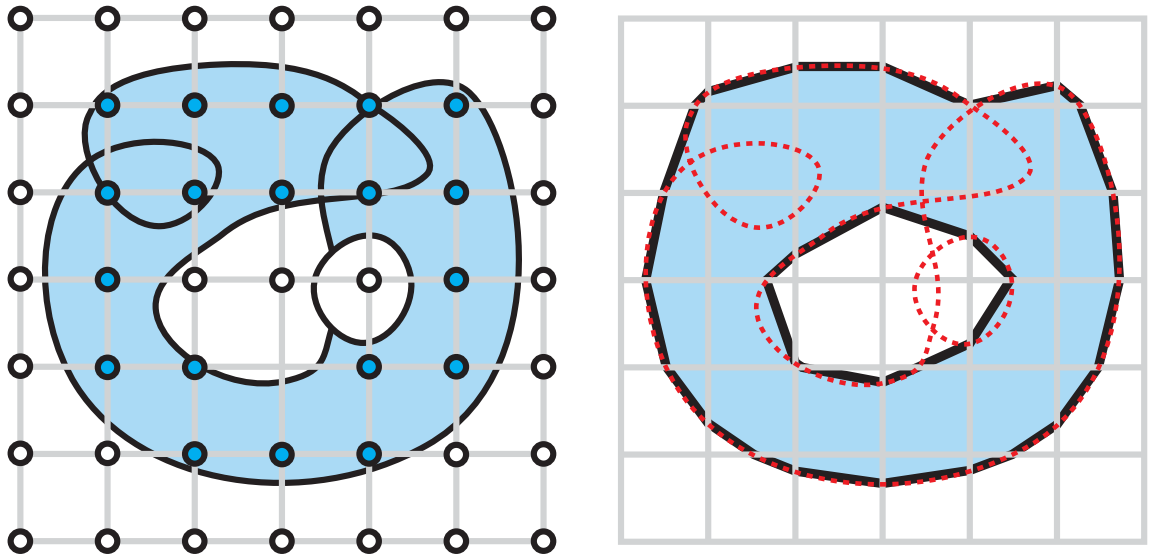

Figure 16: The globally grid-based FronTier method (Glimm et al., 1999), illustrated in 2D. Left: An interface deformed by advection is overlaid with a grid, with in/out indicator labels at node positions computed by raycasting. Right: Using the node labels together with the exact intersection points (edge-crossings) between the grid and the original curve (dashed red curve), marching squares cases can be used to reconstruct a new piecewise linear interface with corrected topology, albeit with some loss of fine details.

change itself is principally achieved within an auxiliary (typically volumetric) representation and inherited by the surface upon reconstruction, rather than by deliberate, explicit detection and surgical modification of the triangle mesh itself. Topology change thus arises *implicitly* through the properties of, or by processing within, the chosen intermediate representation. In this section and those that follow, we have sought to group the methods to be discussed by domain or approach, and then in loosely chronological order within each group.

3.1. Computational Physics: The FronTier framework

The classic example of a topology-adaptive three-dimensional explicit interface tracker in computational physics is the *FronTier* framework proposed by Glimm and colleagues. In fact, they proposed multiple different early 3D implicit *and* global explicit approaches across several papers during the late 1990s and into the 2000s. Their original grid-based ("GB") tracking method (Glimm et al., 1999) used as its implicit representation the set of all intersection points between the grid lines of an overlaid Cartesian grid and the entire surface triangle mesh (i.e., edge-crossings), together with indicator labelings of all grid nodes (Figure 16, left). By enforcing certain simplifying restrictions on the final geometry (e.g., only one intersection per edge, interior and exterior nodes must be separated by an edge intersection, etc.), a typical marching cubes procedure can be used to robustly reconstruct a completely new mesh, with topology changes and topological inconsistencies having been successfully resolved (Figure 16, right). Subsequently observing that this approach often regularizes (i.e., smooths away) sub-grid surface features and thus also suffers volume loss, Glimm et al. (2000) then suggested a combined approach in which, rather than being applied at every step, the grid-based scheme is used only intermittently or as a failsafe for a separate global explicit topology resolution scheme, dubbed grid-free

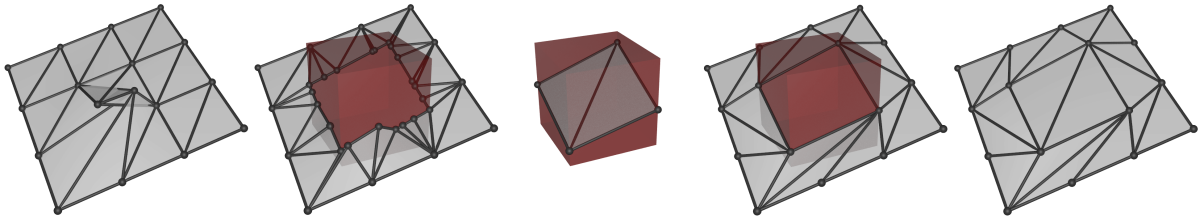

Figure 17: A visualization of the locally grid-based approach for correcting topological errors (Du et al., 2006), applied to a simple single grid cell case in 3D. The grid cell (red) containing a mesh self-intersection (far left) is identified, and the exterior interface triangles are clipped against it and subdivided (second from left). Marching cubes is applied to construct a new interior patch for the cell (center). The exterior clipped triangles are then simplified to match the new interior patch border via edge collapse operations (second from right). Finally, the new patch is merged into the surrounding mesh (far right), thereby eliminating the topology errors.

or "GF" (which we discuss in §4.1). In the same paper, Glimm et al. (2000) also described in more detail their heuristics for the elimination of inconsistent node labelings in the GB scheme.

To further reduce the excessive smoothing induced by the grid, Du et al. (2006) proposed strictly localizing the marching cubes reconstruction process to one or more minimal box-shaped regions of the grid containing any detected intersections or inconsistencies in mesh topology, leaving the rest of the mesh mostly untouched (see Figure 17). In exchange, this *locally grid-based* ("LGB") approach introduces the need to reconnect the newly generated triangles back to the unaffected triangles outside of the remeshed box(es). To do so, any original (exterior) triangles crossing a box boundary are precisely clipped against the box and retriangulated. The bordering curves of the clipped exterior triangles must then be matched to the new interior triangles by simplifying the exterior triangles. Finally, these border edges can be stitched together so that proper watertight connectivity is restored. Bo et al. (2011) subsequently replaced this clip-and-stitch strategy with an alternative in which the triangles initially intersecting the box region are discarded rather than clipped. An incremental triangulation procedure is then applied to reseal the irregular gap(s) between fully exterior input triangles and the new interior triangles, with care taken to avoid introducing non-manifold vertices or edges. They applied this version of the FronTier code to perform impressively detailed two-phase compressible flow simulations of liquid jet breakup involving millions of triangles.

3.2. Computational Physics: Local Contour Reconstruction Method

Another natural implicit representation to distinguish interior from exterior is a smoothed numerical indicator function approximating Eq. 5. In one of the earliest 3D front tracking schemes in computational fluid dynamics, Unverdi and Tryggvason (1992) constructed a grid-sampled indicator function from an interface mesh to inform their underlying grid-based Eulerian fluid simulator. Their approach first transfers the surface normal vectors of the mesh onto a grid using a smoothed delta function, as in the immersed boundary method (Peskin, 2002). Since these normals correspond to the gradients of the desired indicator function, a Poisson problem is then solved over the grid to recover a consistent indicator function whose numerical gradient best matches the normal data. This approach yields a reasonably well-behaved indicator even for meshes in close proximity, a fact which they exploited to the benefit of the underlying fluid sim-

ulator. However, these authors did not use the recovered indicator function to perform topological operations on the mesh itself.

Shin and Juric (2002) did just that in proposing their *Level Contour Reconstruction Method* (LCRM). This scheme uses the approach of Unverdi and Tryggvason to construct a grid-based indicator function from the advected mesh, but then, similar to the global grid-based FronTier scheme, applies a marching cubes-like strategy to rebuild an entirely new, topologically consistent mesh. Rather than simply use an indicator function isovalue of 0.5 for the reconstruction, they instead select a single global isovalue that optimally recovers the total mesh volume measured before remeshing, conveniently ensuring global volume conservation (although they did not provide details of how this optimal isovalue is determined). A further advantage of the LCRM scheme is that one need not even maintain explicit logical connectivity among adjacent triangles during tracking, since the reconstruction process does not require it. Each triangle can simply be advected in isolation. (In computer graphics, such a logically disconnected triangle mesh is often called a triangle soup.) Shin et al. (2005) later observed that reconstruction based on a single global isovalue leads to non-physical transport of volume between different areas of the mesh or even fully disjoint components, since volume is often lost in high curvature regions but redistributed uniformly and globally by the existing algorithm. To address this issue while still preserving volume, their revised algorithm selects an isovalue locally within each cell to encourage its agreement with the original mesh interface location, and only afterward applies a globally shared volume-correcting offset to each of the local isovalues. Notably, LCRM's reconstruction scheme and its use of a Poisson problem to form the indicator field is closely connected to *Poisson surface reconstruction* (Kazhdan et al., 2006), a popular technique later proposed in computer graphics for reconstructing closed meshes from unstructured oriented point cloud data.

Ceniceros and Roma (2005) presented a 2D front tracking algorithm which is similar to that of Unverdi and Tryggvason (1992) except that rather than constructing an indicator by solving a Poisson problem, it constructs a signed distance field from the mesh, which tends to be less costly because it does not require the solution of a linear system. The result is essentially a hybrid level-set/front-tracking scheme, in that it uses a mesh for advection but provides level set data (a signed distance field) for coupling with grid-based physics. However, they did not consider topological changes. Marić et al. (2015) adapted this approach to three-dimensional spatially adaptive simulations performed on a fully unstructured tetrahedral mesh, also without considering topology change.

Singh and Shyy (2007) adopted the LCRM approach to handle topology change, but employed marching tetrahedra for the reconstruction (to sidestep the ambiguous cases of marching cubes) and incorporated a normal-probing strategy to decide when and where to perform topology changes. That is, if the normal distance between interfaces is close to the length of one grid cell, grid-based reconstruction is triggered and performed over the entirety of each disjoint interface involved in such a topology change. In general, restricting the reconstruction to disjoint components involved in the topology change leads to less conversion- and reconstruction-induced smoothing than methods like the original LCRM or FronTier's original grid-based scheme, which perform global reconstruction; however, it still incurs more smoothing than strictly localized methods that restrict remeshing to smaller patches of the interfaces (Du et al., 2006; Wojtan et al., 2009; Bo et al., 2011). On the other hand, the latter methods require additional mesh processing to carefully clip out erroneous regions and stitch replacement patches in place. Singh and Shyy (2007) addi-

tionally applied a global volume compensation via directly offsetting the interface vertices in the normal direction, and modified the edge-collapse operation during remeshing to preserve volume by adjusting the position of the merged vertex. Subsequent enhancements of LCRM include performing higher-order interpolation of the indicator function (Shin and Juric, 2007; Yoon and Shin, 2010) and incorporating a signed distance field to achieve better estimates of curvature (Shin and Juric, 2009). Recently, Gennari et al. (2025) adopted an LCRM approach, but for efficiency used classic marching cubes table look-ups, rather than marching tetrahedra or the original LCRM's more programmatic marching-cubes-like scheme. Following Singh and Shyy (2007), Gennari et al. (2025) also perform reconstruction only for the colliding liquid bodies to avoid loss of detail in uninvolved disjoint interfaces.

3.3. Computer Animation

Researchers in computer graphics observed that explicit Lagrangian meshes can be highly effective at retaining fine-scale visual detail in the context of physics-based computer animation of liquids and viscoelastoplastic materials (Wojtan and Turk, 2008; Wicke et al., 2010), but that their inability to handle topological changes limited broader usefulness. This led to several computer graphics researchers pursuing explicit interface tracking.

Bargteil et al. (2007) proposed a Lagrangian finite element technique for strongly viscoplastic materials using a conforming *volumetric* tetrahedral mesh of the material, in which the outer boundary triangles of the tetrahedral mesh represent the interface (the ambient air is not meshed nor simulated). They evolve the tetrahedral mesh according to the physics, applying collision-response techniques (Bridson et al., 2002) to prevent any overlaps of the mesh. When the mesh becomes excessively deformed, they construct an entirely new boundary-conforming tetrahedral mesh of the material using a slightly modified implementation of an existing optimization-based tetrahedral mesh generation method (Alliez et al., 2005). This global remeshing serves to replace permanently deformed and ill-conditioned tetrahedra with new, high-quality, near-isotropic ones; as an important side effect, remeshing also causes close material regions to merge and thin regions to split apart. Unlike many other implicit methods, this approach does not use a Cartesian grid. It is nevertheless an implicit approach, since it first computes from scratch a new volumetric 3D Delaunay triangulation of the surface vertices as an intermediate representation and topology change occurs implicitly through the subsequent global remeshing (i.e., reconstruction) process; the old mesh is not modified, but entirely discarded and replaced. Unfortunately, as with many implicit schemes, the full reconstruction also tends to discard a fair amount of interface detail at the same time.

Wojtan et al. (2009) proposed an implicit, grid-based topology change approach for triangle meshes that is broadly similar to that of Du et al. (2006), i.e., advect the interface mesh, find its grid intersections (edge-crossings) in regions with topology errors, and use marching cubes cases together with a clip-and-stitch technique to knit a reconstructed patch into place (recall Figure 17). However, they introduced several key modifications and improvements. First, in addition to using grid edge-crossings, they also construct a grid-sampled signed distance field from the mesh geometry to inform the decision of where to remesh, by examining where the mesh and distance field disagree. Second, rather than employ an overly conservative (and potentially large) cuboidal bounding box containing one or more topologically inconsistent regions, they restrict the mesh reconstruction to

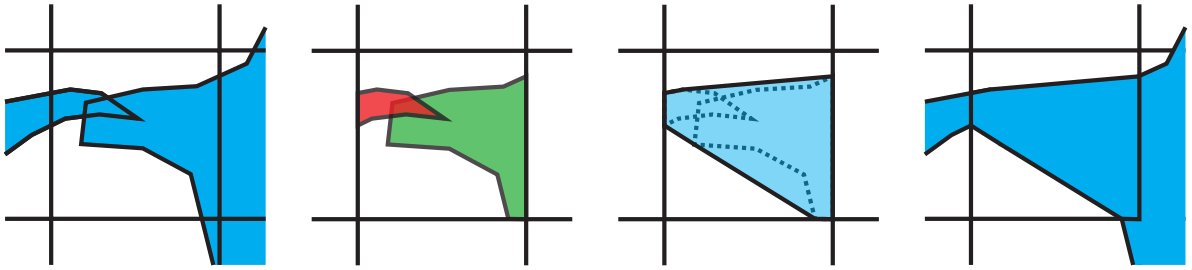

Figure 18: A 2D illustration of the convex-hull based reconstruction strategy proposed by Wojtan et al. (2010), applied to a single cell. From left to right: A cell with an intersection is identified. The mesh is clipped against the cell. The convex hull of all the mesh components (red, green) inside the cell is computed. Lastly, the cell's new convex geometry is stitched back to the surrounding mesh. As a result, the topology has been regularized while retaining greater sub-grid detail than marching squares.

individual topologically inconsistent *grid cells*, further localizing the unnecessary over-smoothing caused by reconstruction. Third, motivated by graphics applications, they proffered a visually motivated strategy for identifying regions to be remeshed, seeking to replace only those regions that are least apparent to a viewer or where topological errors are severe.

Observing that basic marching cubes loses volume and destroys vital sub-grid features, Wojtan et al. (2010) improved their method a year later by replacing marching cubes with a per-cell convex hull-based wrapping strategy that better preserves narrow threads, thin sheets, and other sub-grid geometry, while still regularizing the topology (Figure 18). (Use of the convex hull slightly biases the algorithm towards preserving or gaining volume of the interior material, at the expense of the exterior material.) Furthermore, when stitching in a new surface patch, they observed that simplifying the triangles outside of the replacement cells to match the boundary curves on cell borders unnecessarily sacrifices interface detail (see Figure 17). They therefore instead subdivide only the new triangles *inside* the new patch until they match the outer clipped triangles. The overall result of these two changes is much better preservation of interface detail across mesh reconstruction operations. To separately handle thin threads breaking, they additionally adopted the local explicit splitting approach of Lachaud and Montanvert (1996) (discussed later in §5): identify narrow triangular cross-sections, cut them by duplicating the relevant vertices, close the ends with new triangles, and perturb the surfaces slightly apart. Thus, the overall combined method is technically a hybrid between a new implicit scheme (their core contribution) and a local explicit scheme (adopted from Lachaud and Montanvert).

Müller (2009) proposed an implicit scheme that more closely followed the original GB FronTier scheme of Glimm et al. (1999), in that it applies marching cubes-like cases *globally* rather than locally to reconstruct the mesh from edge-crossings (Figure 16). However, a notable contribution was his principled method to determine the correct interior/exterior node labelings. Specifically, Müller proposed to use the positive winding number rule together with raycasting-based calculation of the winding number at grid nodes, as discussed in §2.7, which is necessary for consistent handling of both overlapping and inverted geometry (merging and splitting). Additionally, observing (like Wojtan et al. (2010)) that basic marching cubes leads to severe volume loss, Müller proposed an *extended* marching cubes approach: he allowed up to two edge-crossings per grid edge and introduced additional interior vertices within each cell to better preserve the geometry,

especially for thin sheets. A potential drawback of this scheme is that the number of cases is significantly higher, so the meshing must be performed algorithmically rather than via a classic marching cubes table lookup. The introduction of extra vertices interior to cells during reconstruction exhibits similarities to the Local Front Reconstruction Method (LFRM) introduced soon afterward (Shin et al., 2011) (described in §3.6.2), although Müller (2009) used the extra vertices to encourage detail/shape preservation rather than volume preservation.

3.4. Computer Vision: Active Contour Models

In classical computer vision, explicit interface tracking largely began with active contour models or *snakes*, introduced by Kass et al. (1988), with important refinements by Cohen and Cohen (Cohen, 1991; Cohen and Cohen, 1993) among many others. Snakes are a two-dimensional dynamic polygon-based or spline-based representation widely used for image-related tasks such as edge detection, segmentation, stereo matching, and more. Typically, an energy-minimization approach is used to apply gradual position modifications to the vertices or control points defining the curve, such that the curve seeks alignment with relevant image features, such as color discontinuities. The original method did not support topology changes, and hence struggled to compete with alternative level set formulations for the same kinds of problems (Caselles et al., 1993). McInerney and Terzopoulos therefore proposed the *T-snakes* method (for topology adaptive snakes) (McInerney and Terzopoulos, 1995, 2000), which finds all intersection points between the polygonal curve and a regular 2D (planar) triangle mesh of a uniform grid and uses marching triangles cases to construct a new interface mesh of the whole domain, assuming only one edge-crossing is retained per edge. The grid nodes' inside/outside labelings are incrementally updated as the curve crosses over them, by locally checking which side of a nearby curve edge the node lies on. This process is simplified by the algorithm's assumption that on a given timestep, the curve may only shrink or inflate, but not a mixture of the two. This implicit, reconstruction-based strategy can be seen as a 2D precursor to the 3D GB FronTier method (Glimm et al., 1999), albeit using marching triangles rather than marching cubes (or squares) and placing stronger limitations on the allowed motion. Soon after, McInerney and Terzopoulos generalized their approach to three dimensions (McInerney and Terzopoulos, 1997, 1999) using marching tetrahedra and dubbing it *T-surfaces*. Thus, this is a striking instance of closely analogous mesh topology-change mechanisms being proposed independently in computer vision (T-surfaces) and computational physics (GB FronTier) at around the same time (1997-1999).

Bischoff and Kobbelt (2004a,b) proposed *R-snakes* (for restricted snakes) which is a 2D active contour scheme where, rather than moving the interface vertices along arbitrary directions, vertices ("snaxels") may move only *along* edges of a background grid, with up to two oppositely oriented interface vertices allowed per edge. This restriction simplifies the types of topology changes that can arise and allows the mesh to be reconstructed with a marching squares-type method. When a vertex crosses a grid node, it subdivides itself, sending new vertices out along the unexplored grid edges incident on the grid node. When two vertices approach each other and collide on an edge, they are deleted, allowing the method to naturally handle merging and splitting. Zheng (2010) proposed a similar but more restricted approach in which vertices do not have sub-grid positions along edges, but instead move discretely from grid node to grid node. However, this simplification greatly reduces the flexibility of the interface shape.

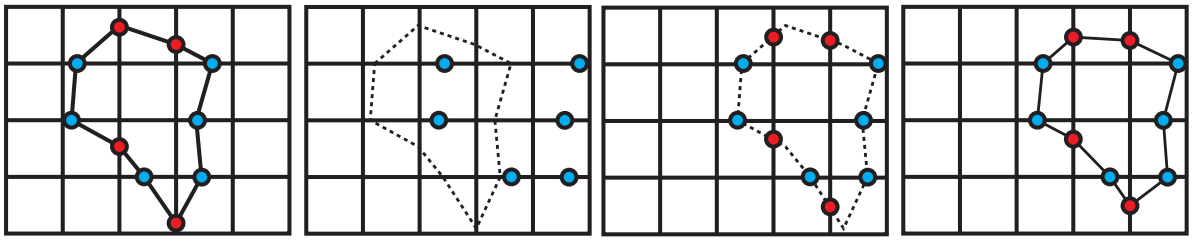

Figure 19: An illustration of the basic 2D split EBIT method (Pan et al., 2024). Far left: An initial interface mesh before horizontal advection. Cyan nodes are "aligned" and red nodes are "unaligned" with respect to the horizontal axis. Center left: Aligned nodes are advected along the horizontal axis. Center right: New unaligned nodes are determined by intersecting advected edges (dashed lines) with grid lines. Far right: The new mesh is reassembled based on the nodes on each cell via marching squares.

Rather than use an interface mesh alone, Pons and Boissonnat (2007a) make use of *volumetric* Lagrangian triangle (in 2D) or tetrahedral (in 3D) meshes of the whole domain (i.e., both exterior and interior regions), of which the interface is a subset (e.g., for tetrahedra in 3D, the interface is a subset of its triangular faces). However, while the whole domain volume is meshed, vertices are still allocated only on the interface. Each volume element is assigned a label to indicate its material (often simply inside or outside), allowing the method to also support more than two materials. They advect the mesh forward in time via its vertices, which can lead to some inverted simplices, potentially indicating topology change. The existing mesh is wholly replaced by computing the Delaunay mesh of all the new vertex positions, which by construction is free of inverted elements. The new simplex labels are determined by checking which material the centroids of the new simplices are contained by, based on the previous deformed simplices (and ignoring the inverted ones). Finally, they remove any vertices that have become interior to the volume and hence rendered unnecessary. They demonstrate results for medical image segmentation tasks in two and three dimensions. This scheme is classified as implicit, since it implicitly handles topology change by constructing an entirely new Delaunay mesh (similar in this respect to the method of Bargteil et al. (2007), published in the same year).

3.5. Computational Physics: Edge-Based Interface Tracking

Zaleski and colleagues recently proposed several variants of a new *Edge-Based Interface Tracking* (EBIT) method. This family shares loose similarities with R-snakes in the sense that interface vertices are restricted to lie on grid edges. Their first version in two dimensions (Chirco and Zaleski, 2023) drew inspiration from an early preprint of Semushin (1988); further details were described by Pan et al. (2024) in an application to multiphase flow. The method uses a dimension-by-dimension splitting approach to transport the mesh, so the advection in each axis is performed independently. Horizontal advection, for example, proceeds as follows (as diagrammed in Figure 19): first, the vertices lying on horizontal grid lines ("aligned vertices") are moved horizontally, hence remaining precisely on their original grid lines. Vertices lying on vertical grid lines ("non-aligned vertices") must then conceptually be moved horizontally, but since they will no longer lie on their original (vertical) grid edges, positions of new unaligned vertices are computed by intersecting the advected mesh edges with the vertical grid lines. The final mesh is then implied by the relationships between the aligned and unaligned vertices in each cell, essentially via marching squares cases (they add an extra cell-center node to

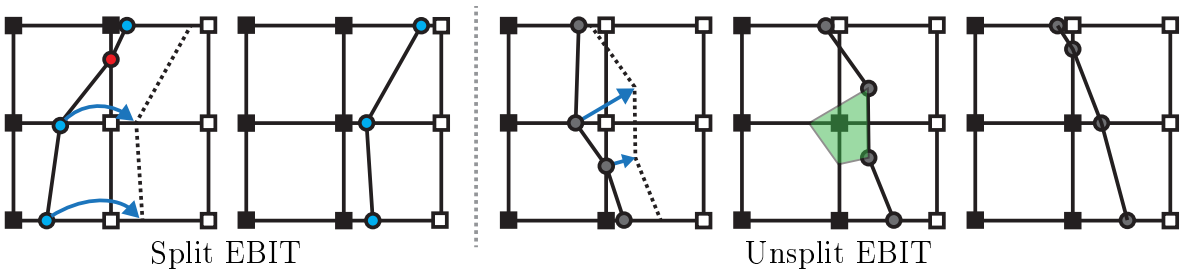

Figure 20: Approaches to node color (inside/outside) labeling under 2D EBIT schemes. Black and white squares indicate node labels. Left pair: In *split* EBIT (Pan et al., 2024), a grid node changes color when an aligned vertex (cyan circle) slides past it, which can be checked trivially based on their coordinates in that dimension. Right trio: In *unsplit* EBIT (Pan et al., 2026), a node changes color when an interface segment sweeps over it, which is checked by performing a point-in-polygon test on the implied polygonal region (green) constructed from the start and end positions of the edge.

resolve the ambiguous case, which we omit from our figures for simplicity). The process is then repeated in the remaining axis (or axes in 3D). Grid nodes are assigned color labels to distinguish interior and exterior, allowing natural treatment of topology change with the typical assumptions of marching squares: no vertices may lie on an edge whose endpoints have the same color, and only a single vertex may lie on an edge whose endpoints have different colors. In effect, this scheme is equivalent to advecting the Lagrangian mesh along one axis, finding its grid edge-crossings, and determining the new interface via marching squares (or cubes) cases, and finally repeating in the remaining dimension(s). Thus, it shares strong similarities with the GB FronTier approach of Glimm et al. (1999) (and, e.g., Müller (2009)), except it is performed dimension-by-dimension rather than in a single unsplit advection step, and, at the implementation level, the actual explicit mesh connectivity is implied by the grid rather than stored explicitly. In contrast to FronTier but similar to T-snakes, EBIT schemes update the inside/outside (color) labeling of grid nodes incrementally when the moving interface *sweeps over them* (see Figure 20), rather than determining labelings from the static mesh configuration after advection using winding numbers or raycasting. However, their approach is designed to be more general and robust than that of T-snakes, allowing arbitrary motion rather than only expansion or contraction. Notably, this collision-based node relabeling scheme is also consistent with the *collision parity* concept proposed by Bernstein and Wojtan (2013); see §4.3. Additionally, observing the typical volume loss issues from global marching-type grid reconstructions, the authors of EBIT reduced this effect by proposing a circle-fit approach to find new grid edge-crossings for their unaligned vertices, rather than assuming the transported interface edges are straight segments. In terms of benefits, the implicit nature of EBIT's mesh connectivity allows implementations to leverage parallelism effectively. The split nature of the method also makes it comparatively simple to extend to three dimensions (Pan et al., 2025). Lastly, the dimensional splitting makes tracking the correct grid node colorings almost trivial (by adjusting colors when a translating aligned vertex traverses a node's coordinate in the active dimension; see Figure 20, left), although tracking the extra cell-center vertices adds some minor complexity. These algorithms have been implemented in the freely available *Basilisk* code (Popinet and coauthors, 2026).

An unsplit version of the EBIT method has also been recently proposed (Pan et al., 2026) to address two limitations the authors identified for the split scheme. First, splitting effectively limits the advection accuracy to first order. Second, for quantities stored

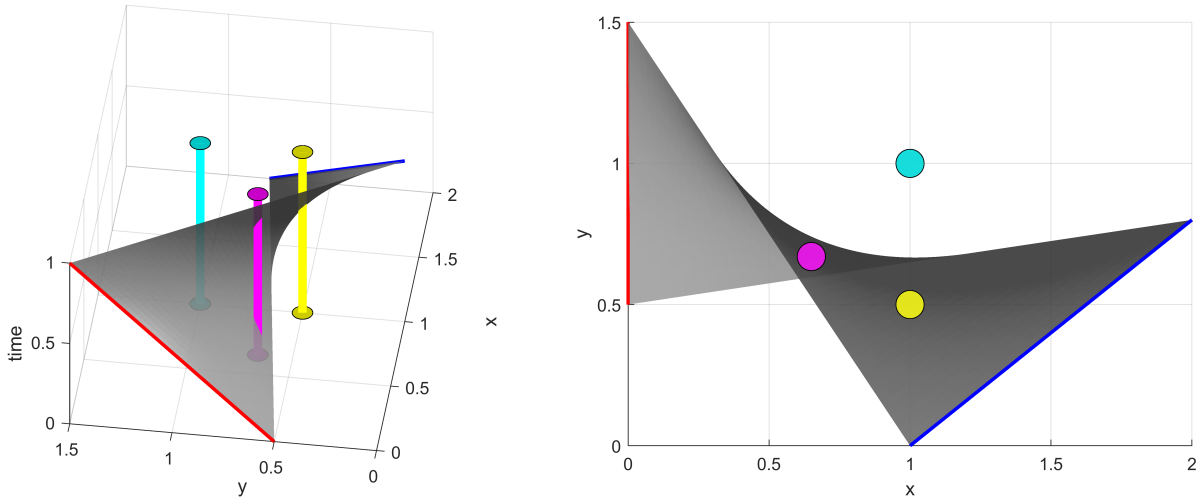

Figure 21: In the 2D unsplit EBIT method (Pan et al., 2026), a grid node should change its inside/outside label when the interface crosses over it, leading to a problem of continuous collision detection (CCD). Left: An interface segment with linear trajectories sweeps out a bilinear patch in 3D space-time. The red segment is the path of one vertex of the segment; blue is that of the other. A static point changes sides if its straight space-time segment intersects the patch once. Right: The same objects projected down to 2D space. The cyan point is not crossed by the patch, the yellow point is crossed once (and so has switched sides of the interface), and the magenta point is crossed twice (in and then out again). Pan et al. (2026) therefore check for crossing using a point-in-polygon test on the 2D quadrilateral.

on and advected along with the interface, splitting makes it more challenging to establish correct correspondences for this data across time-steps. In the proposed 2D unsplit scheme, the main difference is that mesh vertices are instead advected along the (arbitrary) flow direction rather than along one axis at a time. This modified method thus bears an even closer resemblance to the original GB FronTier scheme (Glimm et al., 1999). Some additional geometric complexity arises for unsplit EBIT compared to its split counterpart because of the need to adjust node colors by detecting when grid nodes have been swept over by the arbitrarily deforming interface polygon(s) during advection (Figure 20, right). Robust detection of such node-crossings requires considering the time-dependent geometry rather than a static configuration at a discrete time instant, making it a *continuous* collision detection (CCD) problem. Each 2D interface segment sweeps out a curved bilinear patch in 3D space-time as its vertices move with (assumed) linear trajectories over a timestep (see Figure 21). A static 2D point (the grid node) is swept over by the moving 2D straight segment when the 3D space-time segment corresponding to the point intersects the 3D bilinear patch. The authors approximate this test by checking if the node is contained by the corresponding 2D quadrilateral in space only (i.e., the 2D projection of the 3D space-time quadrilateral bordering the patch), which may be twisted; this choice naturally ignores double crossings, which would have no net effect anyway. However, the authors note that using their circle-fit approach to determine new edge-crossings requires further care to achieve consistent node-color results, since the implied polygonal shape is no longer a quadrilateral (the correct space-time geometry is also unclear, which they do not discuss). Although a 3D unsplit EBIT has not been presented yet, the necessary computation of collision-parities is naturally generalizable to three dimensional triangle meshes using 3D continuous collision detection techniques (i.e., considering 4D space-time), as demonstrated by Bernstein and Wojtan (2013).

Wang et al. (2025) recently proposed another related 2D scheme that uses vertices restricted to edges (or "edge-cuts") of a regular triangular mesh rather than a uniform grid; however, this method additionally attempts to precisely conserve volume, so we defer its discussion to the next section.

3.6. Computational Physics: Explicit Volume Tracking Methods

Since volume is a conserved quantity in many relevant applications (e.g., incompressible flow), there are a large number of interface tracking methods in computational fluid dynamics that fall under the umbrella of volume tracking or Volume of Fluid (VOF) methods (although VOF is sometimes used to describe only the subset of volume tracking methods in which fractions are evolved using conservative Eulerian advection schemes). Most volume tracking schemes do not meet our definition of explicit interface tracking, and thus lie beyond the scope of this review. However, there are a few hybrid volume tracking schemes that employ explicit interface tracking-like Lagrangian ideas, which do fit our definition. We discuss these below. For broader background on volume tracking methods, Rider and Kothe (1998) offer a valuable review of the earlier history and Marić et al. (2020) provide a more recent review of geometric VOF schemes with an emphasis on unstructured grid approaches.

The defining feature of all volume tracking-type schemes is that they seek to precisely retain volume by construction (aside from round-off error introduced by finite precision arithmetic). Classical volume tracking approaches represent the geometry implicitly using a volumetric mesh or grid in which each cell contains a specified scalar volume of material; the actual interface geometry is then reconstructed from that volume fraction data when needed (recall §2.5). Conservation is assured if the numerical volume advection and mesh reconstruction processes together yield the same volume before and after a given timestep. Importantly, emphasis is placed on ensuring *local* conservation, i.e., at the grid cell level. This emphasis contrasts with global conservation, which is sometimes achieved by ad hoc corrections under other types of interface tracking schemes. Like the other implicit topology change methods, merging and splitting are handled implicitly, since the reconstruction procedures typically only have access to the cell volumes (and in some methods, material centroids (Dyadechko and Shashkov, 2005)) and are oblivious to the topology of the original surface.

We focus our discussion below on a subset of hybrid explicit interface / volume tracking methods whose characteristics fit (or nearly fit) our chosen definition of explicit interface tracking. Specifically, we consider those methods that perform some form of forward Lagrangian advection of a volumetric mesh/grid together with the explicit interface itself. We distinguish two subclasses: first, VOF reconstruction-based schemes, which use the advected interface only to determine volume fractions and then perform a VOF-style interface reconstruction that matches the target volumes (as discussed in §2.5); second, "advect-and-adjust" schemes, which explicitly retain at least some components or points of the advected interface geometry (possibly for use in marching cubes-type reconstructions), and then locally adjust the interface geometry for volume conservation.

3.6.1. Volume Tracking with VOF Reconstruction

Mosso et al. (1997) proposed a 2D geometric volume tracking scheme inspired by Arbitrary Lagrangian Eulerian (ALE) methods. It performs a forward Lagrangian advection (the "Lagrange step") of an unstructured volumetric quadrilateral mesh containing

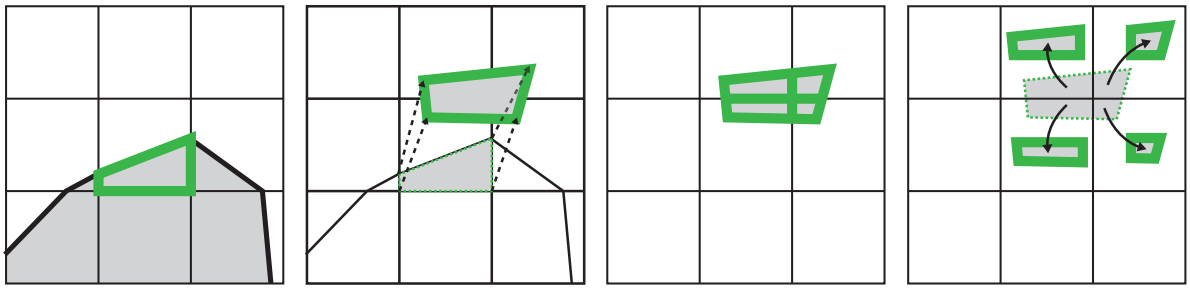

Figure 22: The volume-tracking method of Ashgriz et al. (2004) begins with per-cell (near-)interface polygons (far left), advects them along the flow (center left), subdivides the deformed polygon along the grid lines (center right), and redistributes the subdivided partial volumes to the surrounding cells (far right). VOF reconstruction can then be applied to each cell based on the total volumes it has received.

per-cell material fractions, and then on each advected cell containing the interface, the interface is reconstructed from the volume data to form polygons for each material. Finally, these material polygons are intersected with a new *undeformed* regular background mesh to redistribute material fractions from the advected (and hence warped) cells to the new uniform cells (the "Remap step"), in preparation for the next time step. Thus, the method exhibits a Lagrange-Reconstruct-Remap sequence. Shahbazi et al. (2003) proposed a closely related algorithm on 2D triangle meshes that is shown to achieve second order accuracy. The triangle mesh is advected forward (carrying volume fractions), a VOF reconstruction procedure is performed to recover a polygonal interface, and the intersections of the deformed polygon with a new undeformed triangle mesh are computed to determine new per-triangle volume fractions. The paper discusses both forward and backward advection strategies, but adopts and evaluates only the forward direction. Le Chenadec and Pitsch (2013) proposed a similar scheme that fully develops both forward and backward variants, and even combines them to achieve a trapezoidal-like method. (They additionally incorporate a concurrent level set method calculation, which is used to improve the geometric properties of the method, analogous to the Coupled Level Set/Volume of Fluid or CLSVOF method (Sussman and Puckett, 2000).)

For the above methods, since the geometric reconstruction of the explicit interface mesh happens in the intermediate step (i.e., between Lagrangian advection and remapping), it is not an interface mesh that is advected but a non-interface-conforming *volumetric* mesh carrying per-cell volume fraction data; hence these schemes do not quite fit our definition of an explicit interface tracking method. Were the interface reconstruction step to be done outside of the advection-remapping pair, and the interface polygons, rather than the volume fractions, carried along with the volumetric cells, such a Lagrange-Remap-Reconstruct sequence *would* fit our definition. Ashgriz et al. (2004) take just such an approach in two dimensions (Figure 22). They forward advect the vertices of the polygons representing only the reconstructed *liquid portion* of each grid cell in Lagrangian fashion, compute the deformed polygons' intersections with the grid to determine new volume fraction allocations per cell, and finally perform the next mesh reconstruction from the resulting volumes. They additionally suggest a global volume adjustment for all interface cells to compensate for small net volume changes. (An alternative, more localized approach would be to proportionally rescale the partial volumes of each subdivided polygon before distributing them, so that their total matches the pre-advection volume of the individual source polygon.)

Importantly, the volume correction treatment above is needed because forward advection of the liquid polygons can lead to volume change (or even polygon inversions), for several reasons. If the driving velocity field $\mathbf{V}$ is itself not strictly incompressible, there is clearly no reason to expect volume preservation, so we can ignore this case. Restricting to the case of numerical incompressible flow, $\mathbf{V}$ is often (but not always) divergence-free only in a discrete or weak sense, rather than in the continuous pointwise sense, depending on the PDE discretization technique. This fact was recently highlighted in the front tracking context by Gorges et al. (2022) and Aulisa et al. (2024). Numerical integration of vertex trajectories (e.g., by a Runge-Kutta scheme) introduces additional error sources. However, even if one could advect the polygon vertices under an analytically divergence-free vector field with exact trajectory integration, doing so would *still* be insufficient in general: polygon edges would also need to deform along with the motion of the vertices and become curved in order to truly preserve volume under an arbitrary flow, which is impractical to do exactly. Thus, the advected interface mesh will rarely if ever have precisely the same volume as its pre-advection counterpart, unless more direct interventions are applied.

3.6.2. Volume Tracking with Advect-And-Adjust Approaches

Rather than use VOF-style reconstructions, the next set of volume tracking methods generally use the following template: advect the polygonal/polyhedral interface cells forward, compute target cell volumes using mesh-cell intersections (either at the previous or current time step), either fully retain the advected interface mesh or reconstruct it from edge-crossings in a marching cubes-like fashion, and then perform additional per-cell interface mesh adjustments to recover the target volumes (and possibly improve shape preservation).

In the earliest example in two dimensions, Aulisa et al. (2003) proposed to advect the cell polygons (including interface edges) forward, intersect them with a new uniform Cartesian grid, and compute the net area received by each of the new cells, much like Ashgriz et al. (2004). In contrast to that method, however, the points where the advected interface intersects the grid edges (i.e., edge-crossings) are computed and preserved to enable a marching squares-like reconstruction (they allow two crossings per edge in some cases, enabling slightly more flexibility than standard marching squares). Then a small number of new interface vertices, interior to each cell, are introduced and adjusted in the normal direction to recover the expected per-cell areas. Thus, it can be interpreted as a modification to the original grid-based FrontTier method (Glimm et al., 1999), with extensions to preserve volume (and restricted to 2D). A simple marker reduction process is also used to eliminate cases where too many intersections occur on a single grid edge. Subsequently, Aulisa et al. (2004) modified this scheme by confining the area correction to the interface mesh itself rather than the cells, and by modifying how they redistribute surface vertices, reducing the influence of the grid. They also demonstrated an extension to 3D using quadrilateral meshes of a few simple shapes. However, neither paper explicitly considered topological changes.

Shin et al. (2011) proposed a strategy in three dimensions, related to the earlier scheme of Aulisa et al. (2003), called the *Local Front Reconstruction Method* (LFRM). First, advected mesh triangles are intersected with the individual cells of the uniform grid. The initial (target) volume within each cell is evaluated from the associated intersection polyhedron (only for cells where no topology change is occurring). To reconstruct a

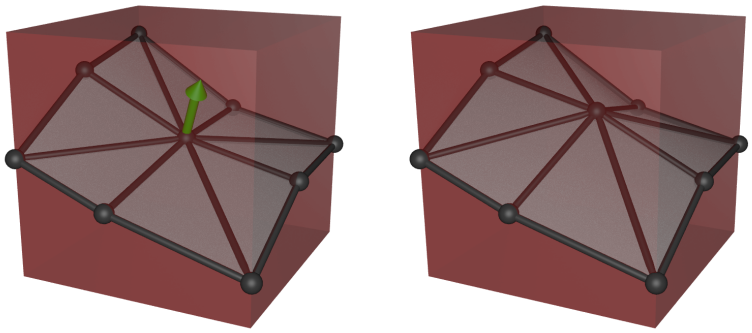

Figure 23: In the LFRM (Shin et al., 2011), an interface mesh patch per cell is computed by using marching triangles on each cell face to construct a border loop and then triangulating the loop using its centroid (left). The central vertex is then moved in the normal direction (green arrow) to correct the interior volume, yielding the adjusted mesh patch (right).

new interface, a marching triangles approach is first used on cell faces to determine a polygonal loop on each cell's boundary. This 3D polygonal loop is then triangulated using an extra interface vertex added at the loop's centroid within the cell's interior. (The Cubical Marching Squares isosurfacing method (Ho et al., 2005) uses a similar idea.) The position of the interior vertex is then adjusted to ensure that the replacement interface has precisely the same volume within the cell as the computed target volume (Figure 23). For cells where topological change is occurring, they do not attempt to conserve volume, and instead rely on a standard marching tetrahedra approach (i.e., analogous to GB FronTier but using tetrahedra instead of cubes). LFRM's volume-conserving vertex position adjustment shares conceptual similarities with the volume-conserving edge-collapse operation proposed by Singh and Shyy (2007) for LCRM. Ho et al. (2016) later presented a related scheme they dubbed the intersection marker method. It is generally similar to LFRM, with some differences in the interface reconstruction process, such as using marching squares rather than marching triangles to assemble cell face loops, and projecting extra cell-face vertices back onto the advected interface mesh. However, Ho et al. did not discuss topology changes.

The approaches above generally assume (often implicitly) that the volume of the mesh is preserved during forward Lagrangian mesh advection; this is not true in general as we discussed at the end of the previous section (§3.6.1). The two-dimensional Polygonal Area Mapping (PAM) method (Zhang and Liu, 2008) therefore offers a similar Lagrangian volume tracking approach, but introduces a backtracking stage to determine the correct *pre-advection* material volumes. Specifically, the method backtraces trajectories of the vertices of each uniform grid cell to determine the cell's pre-image, intersects this warped pre-image cell with the pre-advection interface geometry to find the resulting material polygon (and the exact area contained within), then traces the interface vertices in the cell *forward* to determine their final locations, and finally modifies the resulting new material polygon's interface vertices to achieve the target volume and perform mesh simplification and improvement. In this way, local volume is *exactly* preserved across time steps, while only minimally disturbing the advected interface. This method also allowed for basic merging via a combination of localized union and convex hull operations applied to the interface polygons that arrive in a given cell. Thus, PAM's topology change mechanism is really a hybrid: its pervasive use of a grid to localize and simplify regions undergoing topology change matches implicit schemes, particularly the per-cell convex

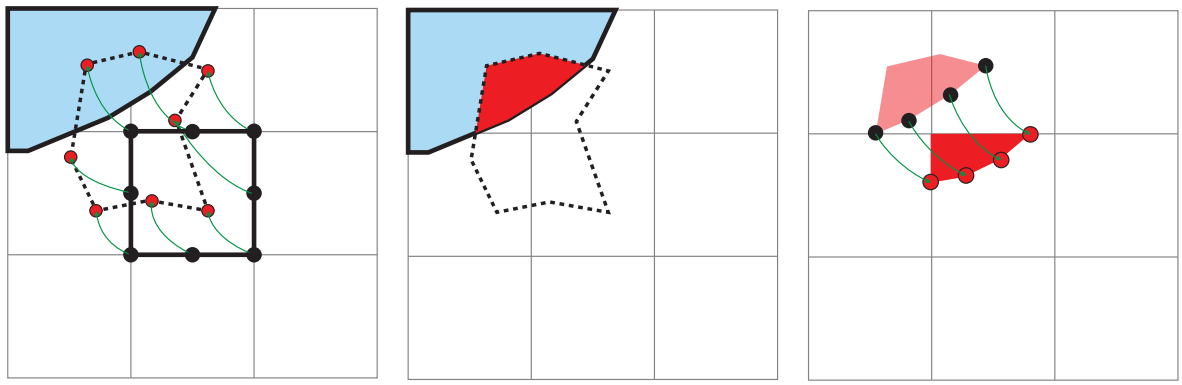

Figure 24: An illustration of Polygonal Area Mapping (Zhang and Liu, 2008). Left: The vertices of a grid cell are advected backwards in time to determine the cell's (approximate) pre-image (dashed line polygon). Center: The cell's pre-image polygon is intersected with the existing polygonal interface geometry (cyan shape), which determines the precise area that the cell should receive (red shape). Right: The interface vertices (black disks) of the intersection polygon (light red) are advected forward to determine their new positions (red disks). Vertex positions inside the cell can then be adjusted (not visualized here) to preserve the original area contained in the pre-image.

hull reconstruction approach of Wojtan et al. (2010); its use of direct local editing of overlapping polygons relates naturally to local explicit schemes.

PAM was then extended to fourth order (with respect to grid cell size) with several improvements and modified to support multiple disjoint polygons per sub-cell (Zhang and Fogelson, 2014), under the name improved PAM (iPAM). In this version, the authors intentionally do not provide a topology change methodology, observing that such processes should be application-dependent. Regardless, this unique approach preserves most of the benefits of both volume tracking and explicit interface tracking. The authors also highlight fundamental connections to Boolean algebras (later explored further (Zhang and Li, 2020)). Subsequent work in this direction developed a rigorous unified theory with which to design and analyze the convergence behavior of a wide range of interface tracking schemes, called MARS for mapping and adjusting regular semi-algebraic sets (Zhang and Fogelson, 2016). Leveraging the MARS framework, the authors proposed a cubic iPAM scheme, using cubic splinegons rather than linear polygons. They also argued that it should be possible retain fourth order accuracy (with respect to cell size) even across topological changes, although to our knowledge they appear not to have demonstrated it. Overall, it appears that PAM-type schemes are the only family of explicit interface tracking approaches that can ensure precise local and global volume conservation, but, since its original version, the authors have advocated against any universally applicable procedure for topological change.

Wang et al. (2025) recently proposed an implicit approach in two dimensions that has connections to both EBIT and PAM. Compared to EBIT (Chirco and Zaleski, 2023), they similarly use as the primary representation a set of vertices restricted to the edges of a background uniform triangle mesh, rather than a Cartesian grid. However, to allow for thin filaments, their method supports up to two "edge cuts" (edge-crossings) along a single edge of a triangle, and adopts a corresponding generalization of marching triangles to recover the implied interface. Topology changes are handled implicitly through reconstruction. Similar to PAM (Zhang and Liu, 2008), they first trace the new undeformed background triangulation backwards in time to determine the triangle pre-images. The

inside/outside status of all triangle vertices can then be computed by testing if they land within the current polygonal interface. Then the edges of the backtraced triangles are intersected with the existing interface to compute areas, and the computed intersection points are advected forwards to the end of step time. Finally, the forward-advected intersection points are projected onto their associated end-of-step triangle edges to compensate for minor advection errors. (Unlike in PAM, they reconstruct rather than retain the per-cell interface patch.) Lastly, to improve area/shape conservation, they introduce an extra interior vertex in one of the marching triangles cases, and slightly shift the edge-crossings to perform explicit local area correction, using the areas computed in the triangle pre-images. They sketch out (but do not implement) a possible 3D extension.

3.7. Computational Physics: Approaches Leveraging Level Set Ideas

A quite different interface tracking approach that also mixes explicit and implicit ideas is semi-Lagrangian contouring (SLC), advanced by Strain and collaborators (Strain, 2001; Bargteil et al., 2006). It constructs a mesh surface at a new timestep by performing marching cubes on a (deformed) signed distance field; cleverly, the value of each SDF query is computed on the fly by *backtracing* in semi-Lagrangian fashion through the velocity field and evaluating the distance from the backtraced point to the prior timestep's surface mesh. In this way, advection is "baked into" the mesh construction process itself. However, although SLC builds an explicit mesh on each step, the mesh never undergoes forward Lagrangian advection; thus, by our definitions this scheme is not an explicit interface tracker with an implicit topology change mechanism, but rather an implicit (or hybrid) interface tracker, closer to the semi-Lagrangian level set methods from which it originally derived.

Nochetto and Walker (2010) presented an interesting hybrid topology change approach in two dimensions. They combine a persistent Lagrangian volumetric triangle mesh with a localized level set evolution strategy to achieve topology change. Each triangle is labeled as liquid or air, and the vertices are advected according to the governing physics. When interface edges (i.e., those shared by both a liquid triangle and an air triangle) come close together, the triangulated volume mesh is locally refined and a temporary discrete signed distance field of the interface is constructed and stored on mesh vertices. A diffusion equation is then applied to the distance field to cause this implicit interface representation to naturally merge or split. Finally, a mesh optimization procedure is applied to realign the nearest mesh edges with the new zero level set, and the material labels of the triangles are reassigned such that the corrected volumetric mesh agrees with the topology of the signed distance field. The optimization process is expressed as a semi-implicit discretization of a gradient flow, based on the (squared signed) distance integrated along the interface. By our definitions, this method is a true hybrid topology change scheme: it is both local explicit (it directly advects and then locally modifies a persistent explicit mesh) *and* implicit (it employs an auxiliary, implicit level set representation). We discuss it here (i.e., consider it an implicit scheme) because the topology change fundamentally occurs within the level set representation; the mesh is simply adapted to agree afterwards. A unique aspect of this method is that it causes topology change by applying diffusion to intentionally regularize the distance field; other implicit topology change methods instead rely on the low(er) effective resolution of the discrete implicit data and/or the conversion processes to handle it indirectly.

The "eXtreme Mesh" (or X-MESH) approach (Moes et al., 2023; Quiriny et al., 2024)

also combines level sets and volumetric meshes, in a slightly different way. In two dimensions, it uses a triangle mesh with permanently fixed connectivity, on which both the physics and a level set equation (or temperature evolution for the Stefan problem) are solved. At each step, the nearby mesh vertices and edges are locally adjusted to conform to the zero level set, thereby ensuring a mesh-conforming sharp interface as the interface propagates along. Whereas Nochetto and Walker rely on level set evolution only for local topology change treatment, X-MESH relies pervasively on level set evolution for *all* interface motion and topology handling steps; the mesh is dynamically adjusted at each step to agree with the level set, rather than being advected in a Lagrangian manner. As such, like SLC, we do not classify X-MESH as an explicit interface tracking scheme at all, but an implicit (or hybrid) one.

4. Global Explicit Topology Change

Global explicit topology change methods allow forward advection of the mesh to temporarily introduce intersections of the interfaces, before seeking to resolve the resulting geometric and topological errors using a global in-place correction that does not move the actual surface; they do so by directly modifying the existing mesh to discard redundant components, without recourse to any auxiliary representation. This process is illustrated in Figure 25. In effect, these methods instantaneously project the mesh from an invalid state to the "nearest" valid state, in accordance with our earlier discussion of winding numbers and inside/outside determination (see §2.7 and especially Figure 11). We consider here only methods that precisely calculate and use the intersection geometry of the piecewise linear interfaces in the topology correction process, without otherwise perturbing the position of the surface; methods that detect intersections of some geometry but use this information only as criteria to trigger other kinds of local edits are instead discussed in §5.

The goal of global explicit topology correction is most naturally achieved by first subdividing all mutually intersecting elements (edges and triangles) of the surface mesh so that all intersections are "resolved"; that is, any intersections among mesh triangles coincide with (i.e., occur exclusively at) vertices and edges (see Figure 26). This procedure is also known as mesh co-refinement (Lévy, 2024). The volumetric domain is thereby divided into a set of closed watertight regions, since the pre-advection mesh is also assumed watertight; it then remains to identify which of these volumetric regions are interior and which are exterior, and use that knowledge to remove any triangle geometry identified as redundant or unnecessary for the final result. In the context of Boolean (CSG) operations on meshes, this type of pipeline is referred to as computing mesh arrangements (Zhou et al., 2016) or the Weiler model (Weiler, 1986; Lévy, 2024). The key differences in the interface tracking setting are that the input is a single (possibly multi-component or self-intersecting) mesh rather than two or more, and the mechanism by which the closed regions are labeled as interior or exterior follows the positive winding number rule, rather than Boolean logic.

4.1. FronTier's Grid-Free Scheme

Motivated by applications in computational physics, especially fluid dynamics, the earliest two-dimensional explicit interface tracking methods in computational physics, proposed in the late 1980s, proceeded in a global explicit fashion: Glimm et al. (1988)

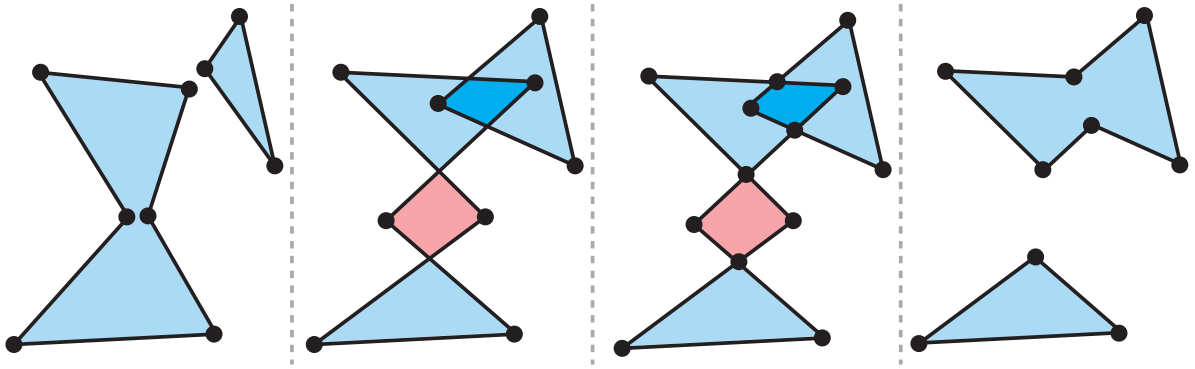

Figure 25: The basic pipeline of a typical global explicit topology change mechanism, illustrated in 2D. Far Left: The mesh before advection is valid and non-intersecting. Center Left: After advection, the mesh is in a topologically invalid, intersecting state. Note that so far there are no vertices at the intersections between interface segments. Shading is used to illustrate volumetric regions, with pale cyan for valid interior, pink for inverted, and brighter cyan for doubly-nested interior (the labeling is not known to the algorithm at this step.) Center Right: Mesh co-refinement subdivides the intersecting geometry, creating new vertices and intersection-conforming edges. Global topological analysis now allows the algorithm to label the resulting watertight enclosed regions as interior or exterior, and thus determine which interface segments and associated vertices are superfluous. Far Right: The resulting geometry after correcting the topology by discarding redundant geometry. A split and a merge have thus occurred simultaneously. Actual deformation of the surface occurs only during the advection step.

identify polygon surface edges that intersect, insert new vertices at the points of intersection, and identify and remove unphysical curve components. They refer to this overall process as untangling. To ensure the complexity of the necessary modifications is manageable and overlaps are small in size, they adopted a time-step restriction. Their algorithm's assumptions rule out support for a curve crossing more than one other curve or inverting within one time step. To determine which curve sections have been rendered redundant, they rely on various observations and heuristics regarding the connectivity and local neighborhood structure around an intersection vertex. A dozen years later, Glimm et al. (2000) proposed a three-dimensional variant of this global explicit approach within their *FronTier* framework, which they called "grid-free" (GF) to contrast with their earlier implicit GB approach (§3.1). After identifying intersecting triangles, they perform mesh co-refinement using Delaunay triangulation. They then similarly rely on

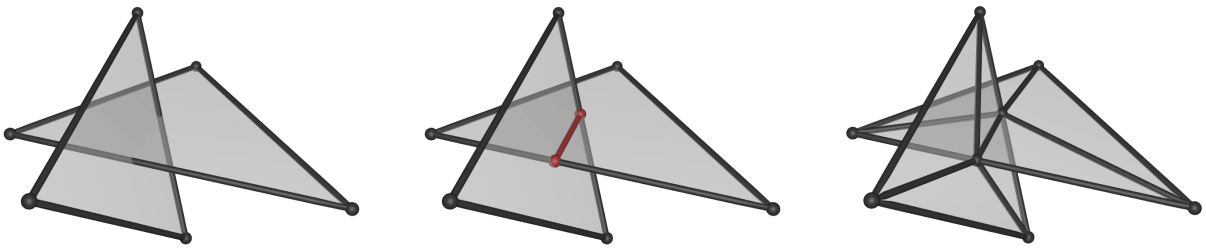

Figure 26: Global explicit topology change methods rely on mesh co-refinement, illustrated here for two intersecting triangles in 3D. Left: An input pair of triangles with unresolved intersections: one edge (black) of each triangle penetrates through the interior of the other triangle's face (grey). Center: The exact intersection geometry (red segment with two red vertices) has been computed. Right: By subdividing the triangles based on the intersection geometry, all intersections are resolved and the mesh is successfully co-refined: all triangles now meet only at valid vertices and edges. Additionally, the surface described by the mesh has not moved or deformed.

a series of heuristics to choose which connected surface patches to retain so that a valid surface is again recovered. Afterwards they check the mesh for topological consistency, and potentially attempt a smaller timestep if the process failed to produce a consistent output. However, in addition to being costly, they observed that the complexity of the algorithm remained an obstacle to ensuring robustness, so they proposed the combined strategy discussed in §3.1, in which their alternative implicit (GB) reconstruction is applied (globally) as a failsafe procedure when this global explicit GF approach fails. Subsequent work on FronTier appears to have largely dispensed with the global explicit approach, instead preferring the compromise offered by their locally grid-based (LGB) schemes (Du et al., 2006; Bo et al., 2011), which can be less accurate but more robust in practice. The three-dimensional GF FronTier algorithm was also predated by the photolithography methods we discuss next.

4.2. Photolithography

Interface tracking problems arise in photolithography simulation of etching and deposition processes for integrated circuit fabrication. In that context, explicit interface tracking has been referred to as marker or (ray-)string methods. Authors in this domain quickly encountered difficulties due to the formation of loops (geometrically inverted regions) in the surface, which led them to propose what they called *delooping* algorithms. The dissertation of Toh (1990) proposed perhaps the earliest such schemes, in both two and three dimensions, computing precise intersections among faces and using knowledge of element orientation to remove inverted regions and correct overlaps, although only limited algorithmic details are provided. Toh commented that the algorithm is fragile, as it struggles to handle more complex scenarios such as multiple intersecting loops, and its computational cost is stated to be high.

Helmsen and colleagues pursued several advances in this same direction (Helmsen et al., 1992; Helmsen, 1994). They sought to improve robustness in computing triangle intersections by perturbing vertices in difficult configurations, and they adopted an octree data structure to accelerate intersection detection. More fundamentally, to determine which triangles to discard, they use raycasting to determine a valid initial interface triangle, and then traverse the whole mesh from triangle to triangle, labeling each triangle as either "surface" or "loop", i.e., to be kept or discarded. Their algorithm uses a parity-like approach for this decision where if the traversal path over the surface crosses a non-manifold (intersection) edge, the corresponding triangles on opposite sides of that edge are assumed to be of different types (i.e., when an outer "surface" crosses into another outer surface, the former transitions to being a redundant interior "loop" surface). They pointed out that this simple binary switching approach fails for nested loops and other nontrivial cases, and suggested that a more correct result can be achieved by accounting for triangle orientation at intersection edges; we present this idea in two dimensions in Figure 27, and it carries over naturally to three dimensions. However, for simplicity they did not implement this variant.

To the best of our knowledge, this early work in photolithography is the first *three-dimensional* precedent for global explicit topology change schemes, preceding FronTier's 3D GF scheme (Glimm et al., 2000) and the subsequent work discussed below. It also appears to be the first to describe a correct traversal-based topology-resolution strategy, consistent with the positive winding number rule and associated concepts discussed in §2.7. Indeed, it can achieve not only "delooping" (i.e., removal of a single erroneous

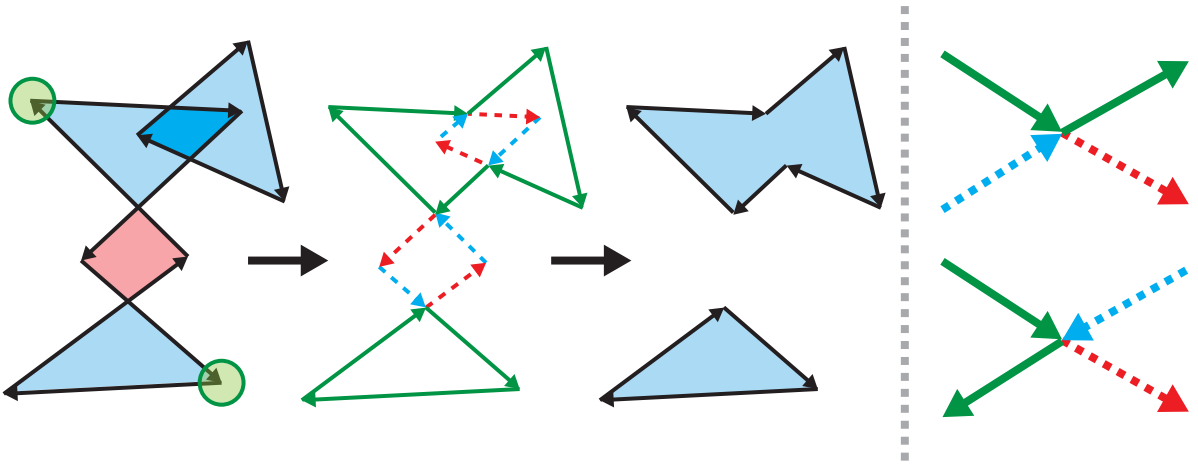

Figure 27: Far left: The traversal-based topology-resolution approach suggested by Helmsen (1994) and implemented by Zaharescu et al. (2007), visualized in two dimensions. First, valid exterior seed points (green circles) are found using raycasting. Center left: The mesh is then traversed from the seed points along the oriented edges, marking them as valid. When the traversal encounters intersections (i.e., non-manifold vertices), the traversal path turns to remain on the new exterior interface (green edges). Center right: After traversal, all the visited valid edges are retained and the others are discarded. Far right: There are two cases to consider to decide the new path at an intersection. The traversal from the incoming edge (top left green arrow) should not stay on the same path (dashed red arrow), nor can it continue on another incoming edge (dashed cyan arrow). Thus the traversal must continue on the remaining, correctly oriented outgoing edge (other green arrow). (If there are multiple valid outgoing edges, the furthest counterclockwise edge is taken.)

loop) but general topology change, and Helmsen's dissertation additionally describes its adaptation to mesh Boolean operations. Later authors in the photolithography domain went on to adopt the FronTier method itself (VanDerWoude, 2000).

4.3. Visual Computing

Researchers in computer graphics and computer vision have explored global explicit approaches too, motivated instead by tasks in 3D geometric modeling, physics-based animation, shape segmentation, and multiview reconstruction. Jung et al. (2004) explored the special case of 3D surface offsetting (shape inflation), which moves vertices in their outward normal directions, quickly leading to self-intersections. Like the methods above, mesh co-refinement is first performed to resolve all triangle-triangle intersections. The result is that the mesh consists of manifold patches, connected and separated by non-manifold edges (and associated vertices). A mesh traversal procedure (broadly consistent with that suggested by Helmsen (1994)) is used to find the desired outer surface of the revised shape, discarding invalid interior mesh patches. This process begins by choosing a seed triangle on the convex hull known to be a valid component of the output surface. Starting from the seed, they grow the valid surface region by proceeding from triangle to adjacent triangle until a non-manifold edge is encountered. The orientations of the several triangles incident on the non-manifold edge are used to determine which manifold mesh patch to traverse to next, and which to discard (see Figure 27).

Zaharescu et al. (2007) sought to move beyond the restricted case of mesh offsetting to address 3D multiview surface reconstruction. They proposed the TransforMesh algorithm, which essentially generalized the preceding approach to robustly resolve *arbitrary* mesh deformations with topological changes. To do so, they allowed for multiple seed

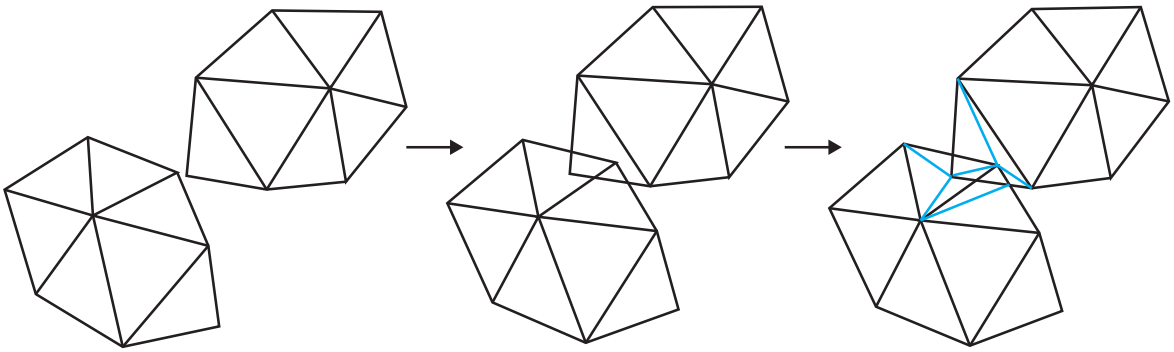

Figure 28: The volumetric merging of Clausen et al. (2013) (shown in 2D for illustration) proceeds by allowing two shapes (left) to collide and intersect (center). They are then co-refined to complete the merge (right) by adding new edges (cyan) and subdividing intersected edges with new vertices.

triangles (and hence mesh components) and used raycasting rather than the convex hull to find them. Thus, their approach fully realized the algorithm first suggested (but not implemented) by Helmsen (1994). To ensure numerical robustness, they made use of the CGAL library (The CGAL Project, 2025) to process the mesh co-refinement. In an extended journal version of their work, Zaharescu et al. (2010) provided additional algorithmic details, made the connections of their method to winding numbers and raycasting more explicit (per §2.7), and more extensively evaluated applications to shape morphing and multiview reconstruction.

In the context of physics-based computer animation, conforming tetrahedral mesh Lagrangian finite element methods had been shown to be highly effective for simulating 3D viscoplastic solids (Wicke et al., 2010), if combined with dynamic tetrahedral remeshing to maintain element quality under large deformations (Klingner and Shewchuk, 2007). However, those methods did not support topology change. Clausen et al. (2013) therefore extended them with a 3D interface tracking method based on tetrahedral meshes rather than interface triangle meshes, as follows. They do not simulate the complementary air (vacuum) region, so the outer surface of the tetrahedral mesh is considered to be the interface. When tetrahedra collide and interpenetrate due to deformation, the intersected tetrahedra are co-refined to reestablish a single connected, compatible volumetric mesh (Figure 28), i.e., they apply a global explicit strategy for merging, justifying its categorization in this section. The local over-refinement (i.e., small tetrahedra) yielded by this step is subsequently eliminated through the dynamic mesh improvement procedures mentioned above. By tracking per-tetrahedron material volume, this method avoids volume being permanently lost; missing local volume identified from collided (overlapping) tetrahedra is gradually reinjected through the physical model rather than geometrically, by adjusting the rest volume(s) of the affected tetrahedral element(s). This aspect means that their work could be considered a variant of volume tracking methods (§3.6). To handle splitting, they made use of a fracture-like physical model, but the geometric task was treated by subdividing an affected tetrahedron along a split plane into separate tetrahedra in (near-)contact; as discussed (§2.6), we do not consider fracture modeling in this review. However, non-manifold vertices and non-manifold edges can also arise naturally under their scheme; by treating these cases through (essentially) the vertex popping/separation and edge popping mechanisms proposed by prior authors (Brakke, 1992) (which we cover in §5), these processes also enable splitting. Thus, this aspect of

their approach can be classified as a local explicit scheme.

Bernstein and Wojtan (2013) presented a technique for interactive geometric shape modeling that considers the colliding trajectories of surface mesh elements to determine which pieces of an intersected mesh to preserve or delete. Like other methods in this section, they compute exact intersections between collided geometry to support topology change. However, their technique is rather unique in that it is designed to handle open (i.e., 2-manifold with boundary) and non-manifold meshes as well as closed ones, since many applications involving artist-driven geometric modeling do not assume or require that meshes represent watertight solids. To achieve this behavior, the method eschews winding numbers, surface marching approaches, and inside/outside labeling. Instead the authors propose a concept of *collision parity*, which counts the number of times a mesh component has collided with other components during the deformation, thereby enabling an array of useful 3D modeling capabilities, including Boolean-like behaviors with open meshes. They determine the collision-parity for complex triangulated surfaces undergoing large timesteps by leveraging continuous collision detection techniques originally developed for cloth and elastic body simulation (Moore and Wilhelms, 1988; Provot, 1997; Bridson et al., 2002). In an appendix, they prove a few slightly restricted consistency relationships between the winding number (which they call the containment number), their collision-parity, and Boolean union operations for closed meshes.

4.4. Active Contours

Researchers in two-dimensional computer vision and image processing have likewise explored using exact mesh intersections for global explicit topology change of active contour models. Ngoi and Jia (1999) described detecting self-intersections, and illustrated how both merging and splitting can be handled by reconnecting the intersecting contours, with care to distinguish positively and negatively oriented contours (recall that in two dimensions, merging and splitting can be processed by a single operation due to the interior/exterior duality, as discussed in §2.6). Notably, they also mentioned the case of region deletion, which is often overlooked. However, details of their precise topological procedure are sparse.

Ji and Yan (2002) described an intricate procedure for handling undesired looping as well as merging operations, by transferring the curve onto a discrete grid, identifying closed loops and intersections, determining the correct orientation of each loop, and adjusting the topology as needed. Their approach makes use of contour traversal, similar in spirit to that of Jung et al. (2004) and Zaharescu et al. (2007). The algorithm description offered by Ji and Yan (2002) is relatively complex, but they demonstrated successful segmentation of multi-component (2D) geometries. Although the algorithm digitizes the polygon edges onto a grid, this operation is best thought of as a separate regularization/remeshing step that simplifies the mesh for subsequent intersection-handling: topology change is still achieved by explicit global topology analysis and deliberate manipulation of the mesh, rather than implicitly through a reconstruction (e.g., as in T-snakes), hence its classification here as a global explicit scheme.

Stoeter and Papanikolopoulos (2004) offered a more straightforward approach without involving a grid. They find exact intersections of the polygonal curves, split the curves at the intersections, and then decide which interface sections to keep or discard based on orientation. This scheme correctly allows for splitting via discarding inside-out regions. However, merging is handled with a separate operation, by combining distinct contours

that collide. They explicitly chose not to correctly handle the creation of interior holes during merging, instead retaining only the simple outer contour of the merged shapes (see their Figure 9 and the discussion in their Sec. IV.B.).

5. Local Explicit Topology Change

Rather than rely on a global in-place correction via co-refinement and topological analysis, the second major approach to fully explicit topology change applies local mesh restructuring to perform topological changes in a manner that mildly perturbs the interface shape , triggered by elements in close proximity to one another or by small intersections (large, complex intersections introduce robustness challenges in the local setting, especially in 3D, because global topological knowledge is needed to properly resolve them). We refer to these methods as local explicit topology change schemes to reflect the fact that they generally perform mesh editing operations with localized stencils and do not rely on global information. . For the most part, these approaches are intended as direct discrete instantiations of the two main continuous processes outlined in §2.6: merging and splitting. An impressive diversity of such schemes have been developed, owing to the flexibility of triangle meshes.

5.1. Computer Vision

5.1.1. Two Dimensions

Early work on this approach came out of the computer vision and image processing community, especially for segmentation and shape reconstruction tasks in two dimensions using active contours (snakes). Samadani (1989) proposed a method in which polygonal loops representing snakes are able to disconnect (becoming open curves) and reconnect depending on the energies driving the snake as well as proximity. This approach relies heavily on the design of energies that (in part) drive the snake towards closing the open contours; since other interface tracking techniques do not necessarily use an energy model, this strategy is generally inapplicable outside of active contours. Ďurikovič et al. (1995) proposed to split a snake when separate vertices on two nearby contours are found to be moving towards each other. In their scheme, the two close vertices are duplicated and a pair of new dividing segments is inserted to separate the interface (Figure 29, left). Later authors also used this approach (Mikula and Urbán, 2012; Balazovjech et al., 2012).

Araki et al. (1997) detect an intersection of the contour and reconfigure the nearby edges to perform a split (or by duality, a merge), without duplicating any vertices or subdividing any edges (Figure 29, center), and Delingette and Montagnat (2000) adopted essentially the same strategy. Notably, this is an early instance of a local explicit approach where actual (but modest) intersections, rather than proximity, trigger the topology change. However, while the detection of intersections *initiates* the process, the actual edits it applies locally reconnect existing vertices (rather than co-refining the mesh) and modify the surface shape (rather than correct it in place), meaning this approach is still classified as being local explicit. Araki et al. (1999) later added a separate merging mechanism for colliding regions, which replaces two intersecting polygons with their convex hull. However, doing so has the drawback of eliminating concave regions and interior voids, and increasing volume. (Adding this step also seems potentially unnecessary, since as discussed earlier, duality in 2D allows a single operation to resolve both splitting and merging.)

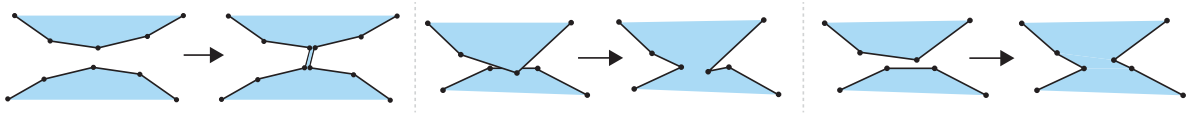

Figure 29: Possible treatments for a merge operation of the cyan region (and corresponding dual splitting of the white region) in 2D, as proposed for snakes. In each pair, the left shows the starting configuration, and the right shows the result. Left pair: Close vertices are duplicated and new (near-coincident) connecting edges are added (Ďurikovič et al., 1995). Center pair: An intersecting edge-pair is identified, and existing vertices are reconnected using new edges (Araki et al., 1997). Right pair: A close edge-pair is identified, and existing vertices are reconnected with new edges (Choi et al., 2001).

Choi et al. (2001), Pauš and Beneš (2009), and Benninghoff and Garcke (2014) perform a geometrically similar topological merge/split operation to Araki et al. (1997), but use proximity as a key criterion rather than actual intersection (Figure 29, right). Doğan et al. (2008) detect intersections, subdivide the coarse curve near (but not precisely at) the intersections, and then locally reconnect the curve to process the merge or split. However, they do not provide specific geometric details. Nakhmani and Tannenbaum (2012) adopted the same topology change method proposed by Araki et al. (1997), but to detect self-crossings of the curve that trigger such changes they proposed a winding (or turning) number-based scheme for improved efficiency.

Several of the active contour methods mentioned above incorporate additional domain-specific criteria to initiate merging/splitting, beyond proximity or intersection alone. These details are outside the scope of our survey, but they exemplify one of the central benefits of explicit interface tracking: fine-grained, application-specific control over the execution of topology change.

Under the implicit volumetric mesh scheme of Pons and Boissonnat (2007a) (discussed earlier in §3.4), topology change is handled by computing a new Delaunay mesh of all the advected interface vertices at each step and determining the new elements' material labels by point queries on the old (deformed) mesh. However, this full mesh reconstruction is a drawback, as it may be costly and can also cause undesired flickering between two nearby configurations over time. (For example, consider the 2D Delaunay triangulation of four vertices that are cocircular: very small changes in the vertex positions can flip the diagonal edge, even if the triangles are well-shaped.) Pons and Boissonnat (2007b) therefore proposed a closely related non-Delaunay *kinetic triangulation* approach (restricted to two dimensions), in which the evolving volumetric triangulation is instead updated dynamically and lazily, via edge flips and collapses. It is thus a local explicit scheme. The mesh connectivity is changed only locally when and where the vertex motion makes it necessary to replace (nearly) flattened triangles that would otherwise become inverted (see Figure 30), rather than recomputing an entirely new Delaunay mesh on each step. The labels for the new triangles produced by edge flipping are assigned based on the labels of the non-degenerate triangles that they replace. Topology changes are therefore handled essentially automatically through these mesh improvement/correction operations, i.e., without needing to design explicit merging and splitting procedures. The authors observed that this new method tends to produce more coherent deformations over time, with fewer undesired perturbations of the interface. They applied this method to medical image segmentation and three-phase combustion in two dimensions; the latter example highlights that the method naturally handles multiphase (i.e., greater than two phase) settings, which we discuss further in §6.

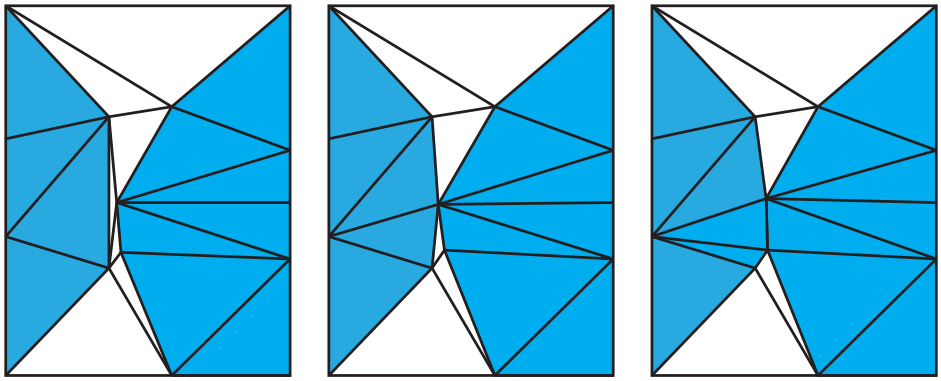

Figure 30: The topology change method of Pons and Boissonnat (2007b) uses a volumetric triangulation of the whole 2D domain. Left: Two interfaces are close together, which is reflected in the presence of slender triangles. Center: Flipping an edge involved in the most slender triangle connects the two sides at a point. Right: A second edge flip occurs, causing the interfaces to become fully merged.

5.1.2. Three Dimensions

Moving to three dimensions, the earliest instance of a local explicit merge operation appears to have been proposed by Leitner and Cinquin (1992) for deforming tensor-product spline surfaces rather than triangle meshes. For two nearby surface patches, with each patch consisting of a 2×2 grid of quadrilaterals, the central '+' shape of each mesh patch is logically disconnected to create an opening, the incident quadrilateral faces are twisted, and the faces on one patch are then connected to the corresponding faces on the other surface patch in an interleaved fashion. The result is a connected interface with the same number of faces. This approach is limited in its generality and applicability, although one could conceivably construct an analogous merge operation for triangle meshes.

As we noted in §2.6, Lachaud and Montanvert (1996) provided a useful classification of the types of topological changes in three dimensions. They also presented some of the earliest local explicit triangle mesh-based approaches for both merging and splitting (later elaborated further (Lachaud and Montanvert, 1999; Lachaud and Taton, 2003)). For splitting (Figure 31), they identify when a tubular region of a mesh becomes a narrow tunnel bordered by three edges forming a "virtual triangle", i.e., the implied interior triangle is not actually a face of the mesh. If the mesh evolution leads to one of those edges shrinking further and hence needing to be collapsed, doing so would ordinarily create a non-manifold edge; instead, a split is executed as follows. The three vertices and edges of the virtual triangle are duplicated (one set for each side, very slightly offset), and two oppositely oriented triangles are inserted to separately "cap" the two sides. This step closes the mesh and restores manifoldness. Finally, the originally desired edge-collapse operation can be executed on each of the two new triangles.

For merging, Lachaud and Montanvert (1996) identify two nearby vertices lying on separate interfaces that are to be connected (Figure 32, left). They prepare for the merge operation by first subdividing all edges incident on those vertices to yield a finer local triangulation. (This initial subdivision step allows the authors to sidestep special cases for scenarios such as folding, where the two target vertices are also close in geodesic distance. We omit this step in Figure 32.) The triangles immediately incident on each of the two central vertices are deleted to create corresponding openings on both surfaces (Figure 32, center). Finally, a roughly cylindrical strip of new triangles is inserted to

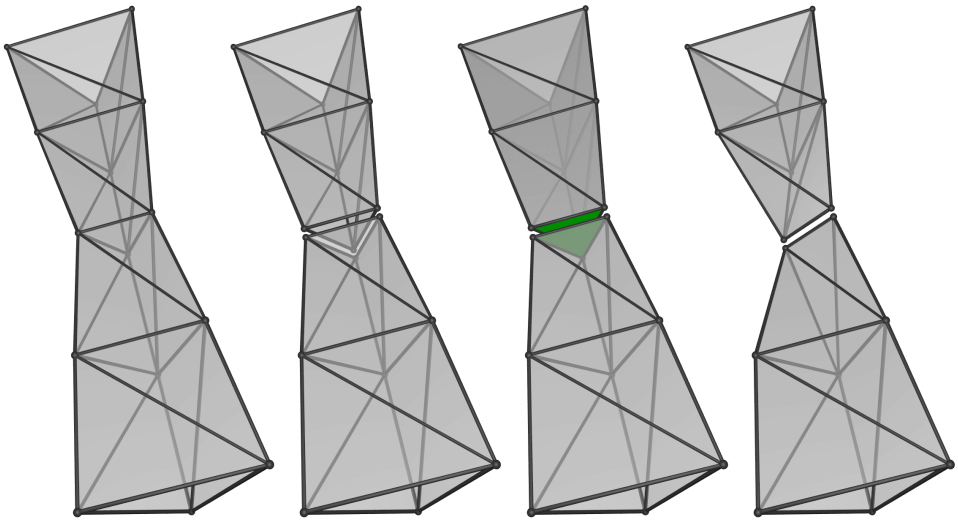

Figure 31: The split operation proposed by Lachaud and Montanvert (1996). Far left: An edge collapse is requested for one edge of a narrow triangular region (which would create a non-manifold edge if not otherwise handled). Center left: The three vertices and edges are duplicated and shifted apart to allow the mesh to bifurcate, leaving two open triangular holes. Center right: The holes are separately capped with new triangles (green) to restore manifoldness. Far right: The originally requested edge collapse is performed on each corresponding new edge.

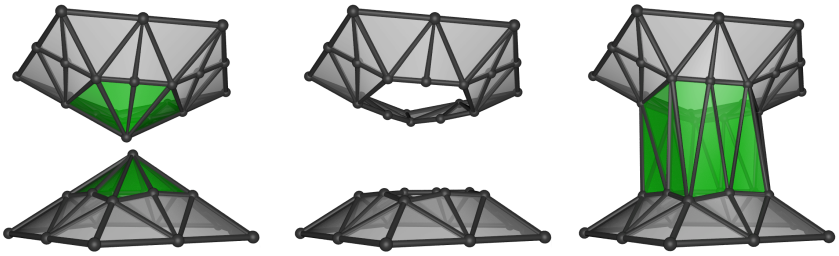

Figure 32: The vertex-centric merge operation proposed by Lachaud and Montanvert (1996) and also adopted by Stanculescu et al. (2011). Left: Two close vertices on distinct interfaces are identified along with all of their incident triangles (green). Center: Those triangles are then deleted creating a pair of holes. Right: The two bordering loops are connected together with a triangulated tube (green).

connect the two holes with a tube and complete the merge operation (Figure 32, right).

Notably, Lachaud and Montanvert (1996) also mention the infrequently discussed case where an interface shrinks to become a single tetrahedron, as can occur naturally if the driving velocity field $\mathbf{V}$ is not divergence-free. In such a scenario, a shortening edge leading to an edge collapse would then create a degenerate non-manifold configuration of two coincident triangles having oppositely oriented faces. Their algorithm instead deletes the tetrahedron (Figure 33).

Bredno et al. (2003) proposed a more general approach for merging and splitting which they state is applicable in dimensions 2, 3, and 4. They detect all mesh intersections, outright delete all the offending triangles, and then use a more general triangulation procedure that fills the resulting open contours with new triangles, either by closing up a single hole or inserting a tube joining two holes on previously disjoint interface patches (similar to Lachaud and Montanvert (1996), but more general). Depending on whether a given pair of holes is handled by closing each independently or by connecting them together, either splitting or merging is achieved, respectively. They propose heuristics

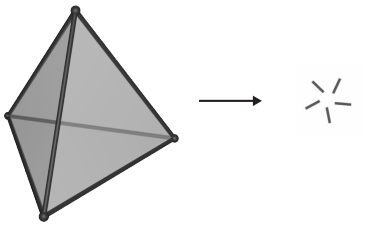

Figure 33: Region deletion as proposed by Lachaud and Montanvert (1996). When a closed region shrinks to a small size such that it consists of only a single tetrahedron, further shrinking initiates the deletion of the tetrahedron.

for making this decision, based on hole proximity, orientation, and size. The stitching process to construct the necessary hole-filling triangulations has similarities to the method proposed by Barequet and Sharir (1995) in the context of repairing meshes with defects.

Duan et al. (2004) adopted essentially the same splitting method as Lachaud and Montanvert (1996), except they do not perform an edge collapse immediately after capping off the tube, instead leaving the triangular tube caps untouched (as in Figure 31, center right). For merging, in their application domain, surfaces can simply be left in close contact until the end of the entire computation (i.e., when some final equilibrium is reached). They therefore perform merging as a post-process as follows. On each interface that is in near-contact, they identify and cluster entire patches of triangles in proximity, and discard those triangles, leaving polygonal holes in the nearby surfaces. Nearby holes are grouped together and a stitching procedure is applied, similar to that of Bredno et al. (2003). Abhau and Scherzer (2010) also took a similar approach for merging, but used topological considerations to construct a database of possible matching shapes to connect two or more holes. They first identify proximal or intersecting patches and delete those triangles. Nearby holes in the mesh are grouped together and the database is then queried to find an appropriate topological match that is then stitched into place.

5.2. Computer Animation

5.2.1. Morphing

In the context of animating a smooth, continuous metamorphosis ("morphing") between two 3D shapes, DeCarlo and Gallier (1996) discussed the issue of topology change, describing what we have termed merging and splitting as pinching and strangling, respectively. Their approach was intended for animation design, so it involves laborious manual construction of mappings (correspondences) between the mesh structures before and after the topology change, and requires performing careful cutting/gluing at vertices or along polygonal curves. In effect, merging occurs by two vertices on opposite interfaces precisely colliding, by design. The mesh connectivity is then reconfigured (replaced) so that the contact point opens into a circular polygon shared by the two regions. Naturally, splitting is the inverse: a circle around a slender region collapses to a single point shared by both sides of the thread. The mesh connectivity is replaced so that the single vertex becomes two coincident vertices, and then each side pulls apart. Due to the specialized and interactive nature of morphing design, however, this approach is inapplicable to the arbitrary motions arising in interface tracking. On the other hand, the idea of duplicating and separating a non-manifold vertex is similar to certain splitting approaches we consider later (Brakke (1992); Brochu and Bridson (2009)).

5.2.2. El Topo

A less commonly discussed issue for local explicit approaches is that performing local mesh edits that add, move, or replace geometry can inadvertently introduce troublesome new mesh intersections, depending on the complexity of the interface(s) in the neighborhood of a given mesh edit. Depending on the application domain, even a single uncorrected material overlap can yield an invalid physical state, potentially ruining an entire simulation. Unfortunately, in turbulent large-scale simulations, tightly spaced and near-degenerate interface configurations can occur frequently. One option could be to augment a local explicit scheme with a subsequent global explicit scheme from §4 to correct the resulting topological errors as a postprocess, although we are not aware of any such combined approach. Another demonstrably effective approach is to leverage robust collision-detection and response techniques, adapted from the cloth simulation literature (Bridson et al., 2002; Harmon et al., 2008), to ensure the mesh always remains intersection-free across all advection, remeshing, and topology change steps.

Brochu and Bridson (2009) pursued this approach with their *El Topo* framework. The essential premise is to consistently enforce an intersection-free invariant: the interface mesh should remain in a non-intersecting state at (nearly) all times. This outcome is achieved in two parts. First, immediately after performing advection of the triangle mesh it may (temporarily) become entangled; collision-handling techniques from the cloth simulation literature (Bridson et al., 2002; Harmon et al., 2008) are therefore applied to safely return the interface to a nearby intersection-free state (notably the work of Bargteil et al. (2007) had earlier used a similar approach). Second, when performing each individual mesh improvement and topological edit operation, collision detection is performed on the implied trajectories, and any individual operation yielding a collision is simply rolled back (canceled) for safety. For example, an edge collapse can be viewed as two connected vertices moving to become precisely coincident, causing the incident triangles and edges to deform; such pseudo-motions can be checked for collisions. Since the vast majority of mesh operations proceed safely without causing a collision, rollback acts as an occasional fail-safe, albeit possibly delaying a desired mesh edit to a later iteration or time step. Individual mesh edits can be robustly tested using continuous collision detection (CCD) techniques (previously mentioned in §3.5) on the affected vertices, edges, and triangles. In doing so, it is critical to minimize the impact of floating-point error (i.e., avoiding false negatives, corresponding to overlooked collisions). Fortunately, robust and general CCD between moving and deforming triangles is becoming an increasingly mature topic (Wang et al., 2021), having arisen early on in the study of computer animation of elastic objects and cloth (Moore and Wilhelms, 1988; Provot, 1997; Bridson et al., 2002).

Since rolling back failed mesh edits can stall progress, Brochu and Bridson (2009) endeavored to design topology change operations for which the stencil of affected elements is as local as possible, i.e., each discrete change to the mesh should be a *small, atomic operation*. In support of this goal, they allow some non-manifold configurations to occur, provided they still partition the space into distinct watertight interior and exterior regions; this design choice allows a mesh undergoing a topology change to proceed incrementally from a manifold state to a non-manifold one and back to a manifold one through a series of atomic operations (possibly across multiple iterations or time steps). This approach avoids large, discontinuous, and instantaneous manifold-to-manifold changes to the mesh, for which new intersections are more likely to arise and intersection-safety is more difficult to guarantee.

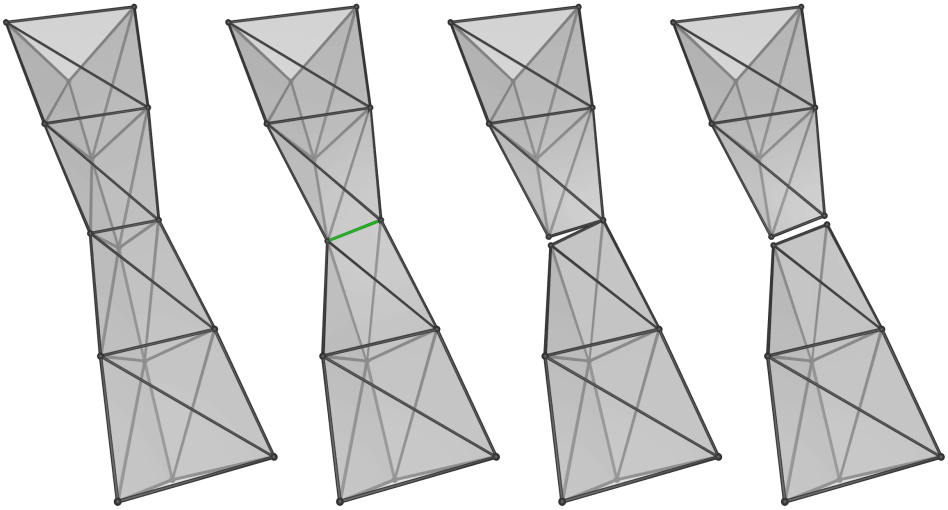

Figure 34: The split process proposed by Brochu and Bridson (2009). Far left: An edge collapse is requested for one edge of a narrow triangular region. Center left: The collapse proceeds, creating a non-manifold edge (green) shared by the two sides. Center right: One of the non-manifold vertices of the non-manifold edge undergoes a vertex separation, duplicating and separating it and its incident geometry. Far right: The other non-manifold vertex undergoes a vertex separation, completing the whole split process. This operation proceeds through non-manifold configurations rather than avoiding them like Lachaud and Montanvert (1996) (cf. Figure 31.)

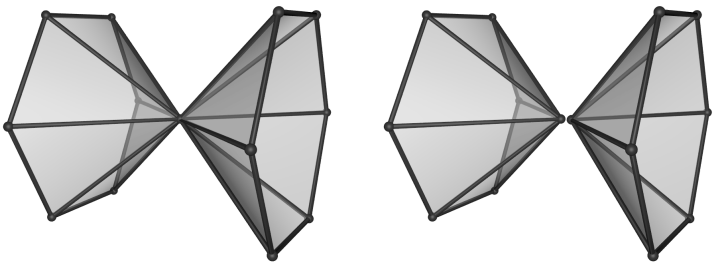

Figure 35: The key atomic operation supporting the incremental splitting process of Brochu and Bridson (2009) is *vertex separation*, which duplicates a non-manifold vertex, moves the new duplicate vertex slightly away from the source vertex, and reassigns the incident triangles to recover a locally manifold state. (Brakke (1992) calls this operation a disjoining pop.)

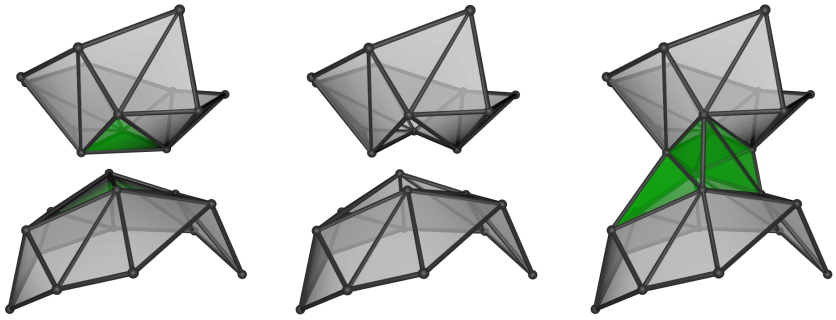

Figure 36: The edge-centric merge operation proposed by Brochu and Bridson (2009). Left: Two close edges on distinct interfaces are identified, with the four incident triangles shown in green. Center: The triangles are deleted to create quadrilateral holes. Right: The two border loops of the holes are then connected together with a tube consisting of eight triangles. This operation involves fewer triangles than that of Lachaud and Montanvert (1996) (cf. Figure 32.)

For splitting, recall that Lachaud and Montanvert (1996) identify a narrowing triangular tube (where one of the edges is a candidate for an edge collapse), cut the mesh along a triangular seam, insert triangular caps, and then perform an edge collapse on each side. This is an instance of a manifold-to-manifold transition carried out in one large operation. Brochu and Bridson (2009) instead allow the initially requested edge collapse to proceed naturally, yielding a non-manifold edge and two non-manifold vertices which may persist if necessary (see Figure 34). To complete the split, each non-manifold vertex is subsequently processed (individually) by duplicating it, perturbing the two resulting vertices apart, and reassigning the relevant incident edges and triangles. They call this duplicate-and-split operation *vertex separation* or pinching (Figure 35). The three atomic edits (one edge collapse, two duplicate-and-separate operations) leading to the final split are each processed independently and checked for intersection-safety. Brochu and Bridson (2009) note that their vertex separation operation shares some similarities with a mesh repair method proposed by Guéziec et al. (2001), which was designed to eliminate non-manifold mesh components. It was also long predated by one of the "disjoining (vertex) pop" operations proposed by Brakke (1992) in his Surface Evolver code; we defer discussion of Surface Evolver to §6.2.2, in the section on multimaterial techniques.

For merging, rather than detecting approaching vertex pairs and replacing the many surrounding triangles (Figure 32), Brochu and Bridson (2009) detect pairs of approaching edges (see Figure 36). For a given edge pair, they delete (only) those two edges and their four total incident triangles, which opens two quadrilateral holes on each surface. Then an eight-triangle tube is stitched into place to create the connection. This method is similar to those of Lachaud and Montanvert (1996) and Stanculescu et al. (2011), but involves significantly fewer triangles, making it more efficient to robustly test for intersection-safety. (Interestingly, the existence of the edge-centric merge (Brochu and Bridson, 2009) and vertex-centric merge (Lachaud and Montanvert, 1996) suggests a similar, but seemingly unexplored, option: a triangle-centric merge, in which just two close triangles are replaced by a six-triangle tube.)

Lastly, Brochu and Bridson (2009) also handle region deletion: if a tetrahedron-shaped region consisting of a single element shrinks, an edge collapse will occur, briefly creating a non-manifold pair of coincident but oppositely oriented triangles. Such pairs are then detected and deleted in a separate clean-up pass. Thus, their full suite of atomic operations consists of both remeshing and topological operations: edge collapse, edge split, edge

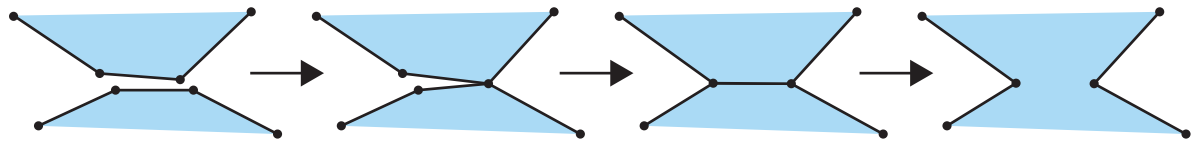

Figure 37: Da et al. (2014) proposed a merging strategy based on snapping pairs of vertices, illustrated here in 2D (compare with other 2D merging schemes in Figure 29). A close pair of vertices is identified and snapped to the midpoint position between them, creating a non-manifold vertex. Another close pair of vertices is likewise snapped, yielding a pair of coincident (but oppositely oriented) edges. This redundant interior double-edge is then deleted to complete the merge.

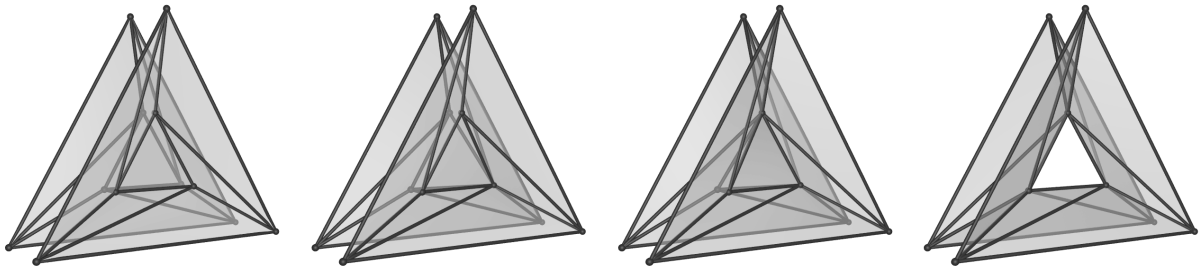

Figure 38: The snap-based merging strategy of Da et al. (2014) in 3D. When two interfaces are close together (far left), pairs of nearby vertices are allowed to snap together. First, a non-manifold vertex is created (center left), then a second nearby snapped vertex-pair creates a non-manifold edge (center right), and finally a third snap creates a pair of coincident triangles, which are deleted to create an opening between the two regions (far right).

flip, vertex relocation (smoothing), vertex separation, edge-based merge, and deletion of coincident triangles. Collectively, these suffice to capture all cases of merging, splitting, and deletion. (Note that while the work was published in a scientific computing journal, Brochu and Bridson were largely motivated by applications in computer animation of liquid and smoke (Brochu et al., 2010, 2012).)

The largest of El Topo's atomic mesh operations is its merging method, as it involves the deletion of four triangles and the creation of eight in a single step, which mildly conflicts with the preference for small atomic operations. Furthermore, while continuous merging and splitting are inverses (see §2.6), their discrete realizations differ sharply in Brochu and Bridson's scheme. Da et al. (2014) therefore extend this framework by presenting a more incremental merging strategy, which uses what they call vertex snapping. This process is diagrammed in 2D in Figure 37 and in 3D in Figure 38. In the simplest case, when a pair of vertices on different interfaces approach, they are "snapped" together into a single non-manifold vertex. This operation is exactly the inverse of Brochu and Bridson's vertex separation (pinching) operation (Figure 35). A second vertex snap adjacent to the first will then create a non-manifold edge. Once all three vertex-pairs of a particular triangle pair have snapped together in this fashion, the result is a redundant pair of coincident and oppositely oriented triangles, which are then deleted to complete the merge and recover a manifold state. (Each individual snap occurs independently, hence non-manifold vertices and edges may persist across timesteps.) To allow more flexible snap-based merging when approaching meshes do not conveniently happen to possess well-aligned vertex pairs, Da et al. (2014) additionally subdivide approaching edges or triangles as needed to create new vertex-pairs that snap more easily with less interface

distortion. For example, when a vertex on one interface approaches a triangular face of another, the face is divided into three sub-triangles by introducing a new vertex, which is then snapped with the approaching vertex; approaching edge-pairs are subdivided and snapped analogously. Da et al. (2014) demonstrate that this approach produces smoother merging sequences compared to the edge-based merging of Brochu and Bridson (2009).

5.2.3. The Deformable Simplicial Complex

In a rather different approach called the *Deformable Simplicial Complex* (DSC) (Misztal, 2010; Misztal and Bærentzen, 2012), Misztal and Baerentzen developed what is in many respects a three-dimensional generalization of the two-dimensional kinetic triangulation method of Pons and Boissonnat (2007b), using a tetrahedral mesh. Given target endpoint positions for all vertices produced by advection, the vertices are moved one by one in linear trajectories from their current positions towards their targets, and if necessary, a vertex is halted (temporarily) at the point at which its movement would cause a tetrahedron to become degenerate (flat). After moving as many vertices as far as possible in this way, they perform tetrahedral mesh improvement (i.e., addressing element quality and interface topology), and then return to moving vertices that had been halted, repeating until all vertices reach their targets. As in the method of Pons and Boissonnat, the mesh improvement phase implicitly resolves topological changes, since both merging and splitting correspond to slender regions (elements) becoming degenerate and being removed, replaced, and/or flipped away. However, a key difference is that Pons and Boissonat employ *only* vertices lying on the interface itself, which places sharp limits on the generality of shapes that can be represented; DSC additionally makes use of extra interior and exterior vertices, which enables essentially arbitrary interface shapes and can provide much better-shaped tetrahedra. Furthermore, the extension to 3D requires significant changes: more complex tetrahedral mesh improvement operations (Klingner and Shewchuk, 2007; Freitag and Ollivier-Gooch, 1997) and extra mesh operations to specifically maintain the quality of the interface, similar to some used by Brochu and Bridson (2009). DSC's authors demonstrated successful applications to geometric flows (e.g., mean curvature, offsetting), cut locus construction, point cloud reconstruction, and fluid animation; later they pursued multiphase flow (Misztal et al., 2013), topology optimization (Christiansen et al., 2014), and image segmentation (Nguyen et al., 2019). As compared to the interface-only approach of Brochu and Bridson (2009) which relies on collision-handling strategies to avoid degeneracies and intersections, this scheme achieves the same objective by requiring that the volumetric mesh always be valid and contain no inverted elements. The use of tetrahedral elements also makes this approach convenient for integration with applications requiring volumetric finite element-based PDE solvers, whereas surface-based approaches would require adding an auxiliary 3D grid or mesh.

5.2.4. Hole-Pairing

Much like the method of Bredno et al. (2003) discussed earlier, Chentanez et al. (2015) developed an approach which detects all intersecting triangle mesh geometry, discards those triangles, and then performs a hole-pairing and hole-filling procedure to either close each hole or find pairs of holes to connect together, thereby resolving topology changes. They additionally take steps to ensure the mesh is manifold and to remove interior mesh components after merging. This scheme sidesteps the need to perform collision-*response* as required by Brochu and Bridson (2009); intersection-*detection* is sufficient. However, the result is not strictly guaranteed to be intersection-free. Chentanez et al. (2016) later

heavily optimized their method's efficiency by extensively exploiting GPU parallelism, and thereby achieved impressive runtimes.

5.3. Shape Modeling

In the context of 3D shape modeling, explicit interface tracking can offer a powerful tool for interactive sculpting of smooth shapes through a virtual clay metaphor. Since earlier methods of this kind did not directly support topology change (e.g., (Angelidis and Singh, 2006)), Stanculescu et al. (2011) were inspired to develop a topology-adaptive strategy. Similar to the merging operation of Lachaud and Montanvert (1996), they identify close vertex pairs, delete the surrounding ring of triangles, and generate a connecting tube. (The primary difference is they do not subdivide the surrounding triangles first.) Their splitting operation is quite similar to that of Brochu and Bridson (2009): non-manifold edges and vertices are created naturally during remeshing operations (i.e., edge collapses), and a separate clean-up pass duplicates and separates them. They sidestep the need for triangle-based collision detection and response by bounding the maximum vertex displacement per step, frequently adapting the mesh to maintain a uniform surface sampling, and using simplified sphere-based proximity checks. A follow-up paper significantly generalized this method to allow one to augment the interface itself with curve networks that evolve, deform, and change topology along with the supporting interface (Stanculescu et al., 2013). This feature can be used to represent and track sharp geometric features or for stylistic/decorative purposes.

Constant mean curvature surfaces are a generalization of minimal (zero mean curvature) surfaces which have seen significant mathematical study, and have proven useful for modeling the shapes of soap foams/films and various architectural structures, such as inflatable domes or tensile membranes. Pan et al. (2012) presented a discrete method for computing such surfaces, using a deformable triangle mesh whose vertices move to minimize an energy functional. While their focus is on a novel energy discretization, they do handle splitting, as a side-effect of edge flipping performed during the optimization process to improve element quality. When narrow tubes arise, they encounter edges where the would-be flipped edge already exists, and performing the flip would introduce a non-manifold edge. They execute the flip nonetheless, collapse the resulting non-manifold edge, and separate the resulting non-manifold vertex à la Brochu and Bridson (2009), which yields a split. (Thus, tube splitting is triggered by an edge flip, whereas in similar schemes, it is triggered by an edge collapse.) They did not consider merging. Their work was inspired partly by the much earlier Surface Evolver method for energy-minimizing surfaces (Brakke, 1992), which we discuss later in the context of multicellular/multimaterial geometries (§6).

5.4. Computational Fluid Dynamics

Nobari and Tryggvason (1996) provided an early instance of three-dimensional explicit interface tracking applied to colliding and coalescing droplets. They performed merging through a specialized local explicit operation "to automatically remove the front bounding the thin film between the drops at a prescribed time", albeit without providing geometric details of the procedure. In subsequent work they commented that this three-dimensional mechanism had not been generalized enough to handle more complex configurations, such as the Rayleigh-Taylor instability they studied in two dimensions (Tryggvason et al., 2001).

Cristini et al. (2001) proposed to perform splitting as follows: first identify narrow tubular regions and place two cutting planes orthogonal to the tube. Then, locally adapt the mesh to ensure rings of nearby mesh edges all lie in the associated planes, and delete the triangles between the two planes. The split is completed by inserting triangulated hemispherical caps onto each resulting opening. An advantage of this approach is that the resulting surfaces are comparatively smooth (assuming the cut happens while the surface is still fairly well-resolved), and thus it avoids suddenly introducing large curvatures at the cut that can, for example, be problematic for curvature estimation to evaluate surface tension forces.

Razizadeh et al. (2018) proposed a combined merging/splitting strategy quite similar to that of Da et al. (2014); see Figure 39. They first identify a nearby pair of oppositely oriented triangles on distinct interfaces. They snap the two closest vertices of the pair together to the midpoint between them, and then crawl over the neighboring edges and triangles snapping corresponding vertex pairs (and by association, their incident edges/triangles) if they are also close enough. In a typical head-on merging case, this pairing-and-collapsing process creates a contiguous patch of adjacent non-manifold coincident triangle pairs (which they call *dual elements*), bordered by non-manifold edges. These coincident triangle pairs are redundant and can simply be deleted (along with any edges or vertices likewise rendered redundant), which connects the two volumetric regions together. Thus, this process is analogous to the snap-based merging proposed by Da et al. (2014), with a few differences. First, Razizadeh et al. (2018) perform the process as one collective operation, first assembling the entire patch of dual elements and then deleting it all together, where Da et al. interleaved individual vertex snapping and triangle deletion operations. Second, Da et al. incorporated collision-checking at each step. Third, Da et al. (2014) perform additional targeted local remeshing to encourage the creation of more readily "pairable" vertices and hence reduce the amount of deformation that occurs when vertices are merged. Notably, in addition to merging, Razizadeh et al. (2018) also rely on their pair-collapse-delete approach to facilitate splitting: the triangle-matching process will cause sufficiently thin threads to collapse flat; deleting the resulting degenerate triangles separates the thread into two disjoint parts. While they allow non-manifold edges and vertices to occur during this process and possibly persist, they do *not* perform vertex separation (recall Figure 35). They instead rely on subsequent vertex-pair collapses applied to the triangles incident on the non-manifold edges and vertices to complete the split (see Figure 40).

Regnault (2023) studied bubble collisions in multiphase flow for which they proposed another merging approach. First they identify nearby patches on approaching surfaces and carefully construct a one-to-one mapping between the two surfaces. Each side is remeshed to introduce compatible triangles with edges that align with a smooth contour making up the border of the region to be connected. The connection is performed either by constructing a triangulated connecting tube or by snapping together the two rings of vertices on the open contours at the midplane between. Lastly, the redundant interior surface patches are deleted, and volume-preserving remeshing (Kuprat et al., 2001) is performed to smooth the junction regions. This approach has a broadly similar flavor to the method of Bredno et al. (2003) or Duan et al. (2004) in the sense that it connects together larger patches that originally consisted of many triangles; however, the merging contour is carefully constructed to be much smoother. (We conjecture it may also share similarities with the approach of Nobari and Tryggvason (1996), for which details are

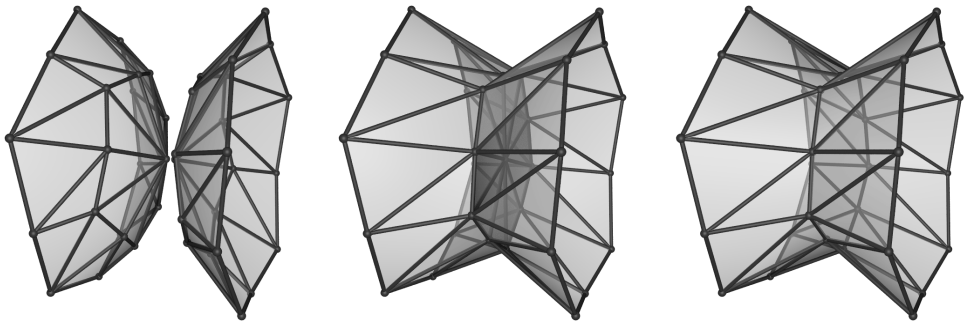

Figure 39: The merging strategy of Razizadeh et al. (2018). First, nearby vertices and their associated edges/triangles (left) over a pair of interface patches are snapped together to create a contiguous patch of coincident duplicate triangles (center). The resulting redundant interior patch is then deleted to complete the connection (right).

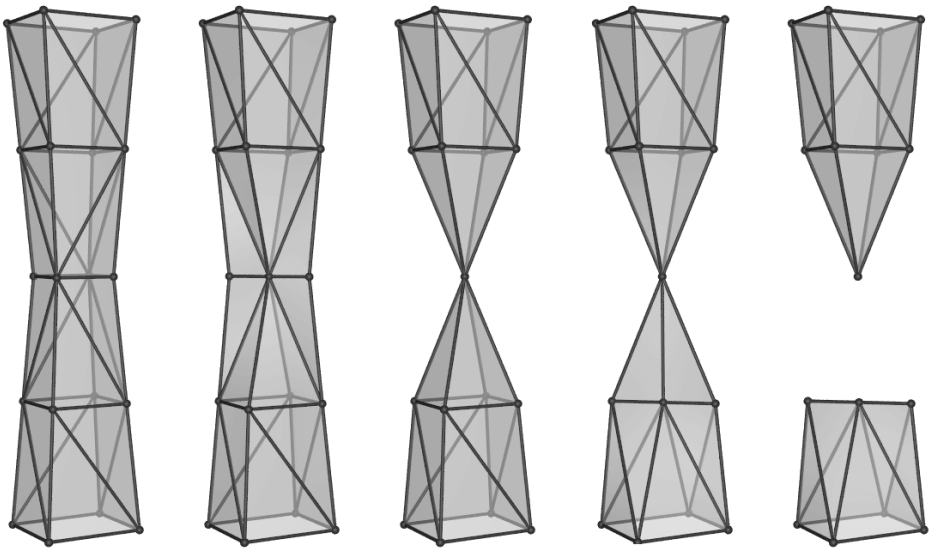

Figure 40: Similar to their merging approach, Razizadeh et al. (2018) achieve splitting by collapsing vertex pairs, which leads to duplicate triangles (second and fourth columns) that are then deleted (third and fifth columns). Rather than perform vertex separation to resolve non-manifold vertices (Figure 35), they wait for further collapses that eliminate incident geometry and thereby disconnect the thread.

not available.) Regnault (2023) performs splitting by checking for narrow threads (as detected by small distances between interfaces in the normal direction) and then using a clustering approach to segment the overall mesh into distinct "bubbles" to be separated. Next, the triangles in the border regions between bubbles are deleted and a hole-filling procedure is applied to close up the meshes. A similar volume-preservation procedure is once again applied to avoid losing material. This method is shown to work well for a set of simple topology change scenarios, but it is unclear to what degree it can handle more general cases (the authors state explicitly that it doesn't handle the dual merging behavior that leads to breakup of a sheet).

As in computer vision and computer animation, there have been a few methods proposed in computational fluid dynamics that rely on volumetric tetrahedral meshes, for both the driving physics and the interface tracking. Quan and Schmidt (2007) developed a moving tetrahedral mesh method for incompressible two-phase flow, which incorporated a treatment for splitting. They first identify a thin tubular liquid region to undergo splitting, place two cutting planes on either side of the desired split location (a few tetrahedra widths apart), and simply relabel all tetrahedra between the cutting planes as gas instead of liquid. Since such a relabeling results in jagged tetrahedral geometry for the caps of the separated regions, they additionally project the cap vertices into a hemispherical shape and perform local mesh improvement to improve the tetrahedra. The end result is thus similar to the hemispherical capping approach of Cristini et al. (2001), except that Quan and Schmidt use a tetrahedral mesh. Their initial method could only be applied to cutting along a major axis, so they extended it to arbitrary planes in a subsequent paper (Quan et al., 2009). The later paper also introduced a corresponding merging operation, which similarly overlays a (conceptual) hyperboloid-like connecting tube between two approaching interfaces, relabels the tetrahedra within it from gas to liquid to form a bridge, and again performs vertex projection and smoothing to improve the interface. The approaches of Quan et al. also have connections to that of Nochetto and Walker (2010) (see §3.7): both use deforming volumetric meshes, but whereas Quan et al. use local explicit edits to deliberately merge or split, Nochetto and Walker layer a level set method onto the mesh to treat these changes implicitly.

6. Multimaterial Interface Tracking

6.1. Interface Tracking Concepts for Multiple Materials

With a few important exceptions, the methods we have discussed so far target the two-phase setting in which there is only a single type of interface, which separates an interior domain Ω^- from an exterior Ω^+. However, there are many application scenarios involving three or more distinct materials or phases $\{\Omega^1, \Omega^2, \Omega^3, ..., \Omega^n\}$, where all of the pairwise interfaces $\{\Gamma^{1,2}, \Gamma^{1,3}, \Gamma^{2,3}, ..., \Gamma^{n-1,n}\}$ between the phases must be tracked (we will use the terms material and phase interchangeably). Here, as in the two-phase setting, an individual phase Ω^i can consist of multiple disjoint components. Some example applications include three-phase flows such as air/water/oil (Jabbour et al., 1996), biological cellular structures such as embryos (Maître et al., 2016), multi-component 3D shape modeling (Brochu and Schmidt, 2017), soap bubble cluster dynamics (Da et al., 2015), and crystal grain growth (Lazar et al., 2011).

In the presence of more than two phases, new types of topological changes arise. First, a generalized multimaterial merging behavior occurs in the case of two distinct phases

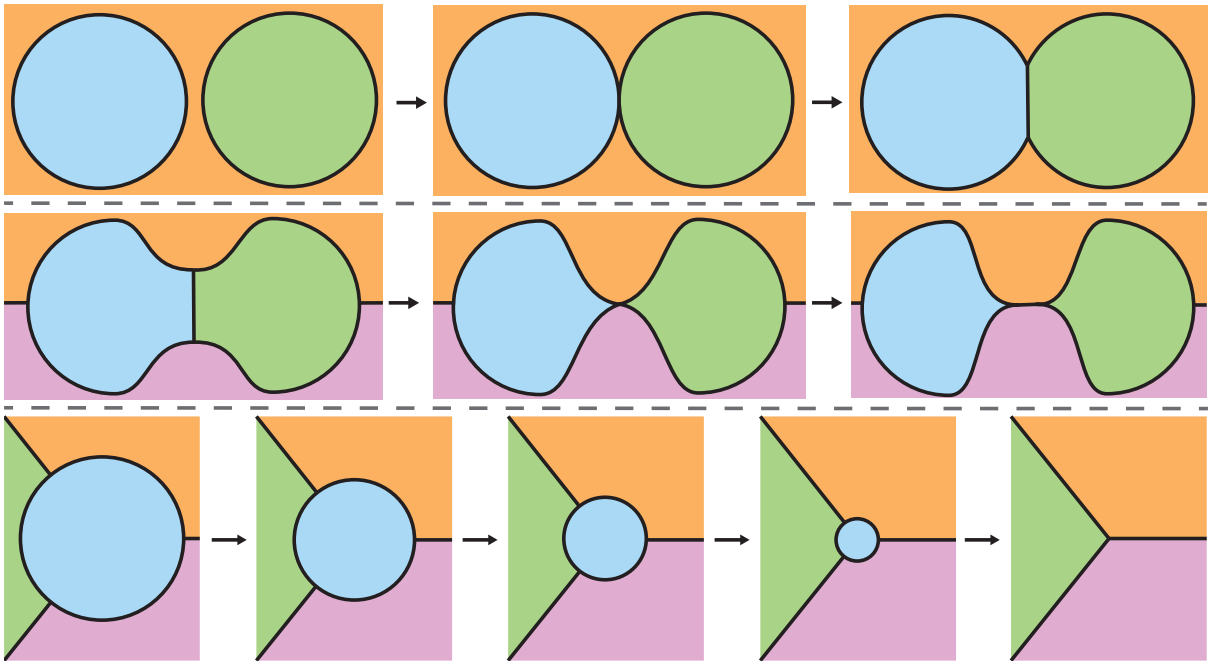

Figure 41: Multimaterial topology change types in 2D, with time ordered from left to right. These can be compared with their two-phase counterparts in Figure 8. Top: Multimaterial merging connects previously disconnected, colliding phases (green and cyan). Middle: A T1 process rearranges the local connectivity of connected phases, creating and removing interfaces corresponding to different phase pairs. The connection between cyan and green is eliminated and a new interface between pink and orange is formed. Bottom: A T2 process deletes a collapsing phase (cyan) that is also in contact with multiple other phases.

coming into contact while immersed within a third phase, creating a new interface between the two; a familiar physical example is two soap bubbles colliding to form a "double bubble" (Figure 41, top). Second, local rearrangement of the connectivity of multiple contacting phases can occur, which is often called a T1 process in foam dynamics (Weaire and Hutzler, 1999) or grain switching (or neighbor switching) in grain boundary dynamics (Kawasaki et al., 1989; Wakai et al., 2000). This operation, diagrammed in Figure 41 (middle), can also be interpreted as a combined generalization of splitting and merging operations, where several nearby phases (already partially interconnected through the network of interfaces) change their local connectivity relationships. A tangible example of this process occurs when a soap bubble squeezes between and past other soap bubbles in a cluster. Thirdly, whereas a fully isolated interface component could collapse to zero volume and disappear in the two-phase case, in the multiphase case the same process can occur while the disappearing region is still in contact with two or more adjacent regions of differing phases (Figure 41, bottom) — consider a single bubble within a cluster leaking all of its air into its neighbors and disappearing. This operation is often called a T2 process (or grain/cell disappearance). Thus we can see that each of the two-phase operations (merging, splitting, deletion) has a loosely corresponding multimaterial generalization (multimaterial merging, T1 process, T2 process); we encourage the reader to compare Figure 8 with Figure 41.

The multimaterial setting can also require modifications to the triangle mesh interface representation itself. With two phases (one interface), non-manifold mesh geometries are infrequent and often transient, with many algorithms intentionally avoiding them

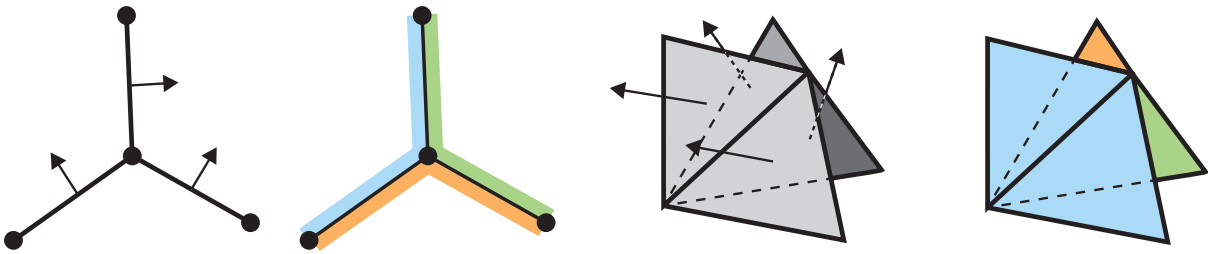

Figure 42: In multimaterial scenarios, there is no way to consistently orient the face normals (arrows) of the mesh at non-manifold junctions. Instead, one can require that the phase labels (colors) on the same side of an interface are consistent between neighboring elements. Left pair: 2D setting, with three materials at vertex. Right pair: 3D setting, with four materials at an edge.

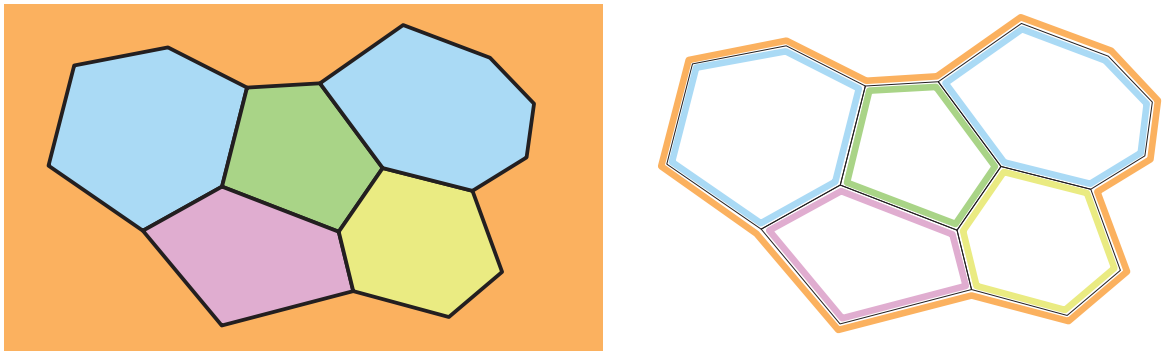

Figure 43: Multimaterial scenarios require the use of non-manifold meshes to represent multiple phases in persistent contact. Left: A 2D multimaterial domain with colors indicating the phase of each volumetric region bordered by a polygon. Right: The corresponding 2D mesh with colored phase labels attached to the front and back side of each edge, including for the orange "ambient" phase. A valid mesh must have the same phase label for all interior faces within a closed region of the mesh.

altogether. With three or more phases, however, interfaces regularly meet at persistent triple- and quadruple-junctions, such as the Plateau borders arising in soap films. The underlying triangle mesh representation therefore *must* allow non-manifold vertices and edges. In addition, there is no way to consistently orient all the triangle normals in a single non-manifold mesh; for example, at a shared non-manifold edge representing a triple-junction curve, two of the triangles can have mutually consistent orientations, but the third must disagree with one of the other two (Figure 42). One reasonable way to generalize the important notion of triangle orientation and its relation to interior/exterior labeling is to assign two *phase labels* (or material labels) to each oriented triangle, one for its front phase and one for its back phase (Figure 43). Then, instead of requiring consistent triangle orientations for triangle-pairs sharing a manifold edge, one can instead require that all triangles bordering a single closed region have the same phase label on the inside of that region (Da et al., 2014). Alternatively (and equivalently) one can explicitly introduce a new object type representing a volumetric region or phase (closed polyhedron, or set of polyhedra), and store logical links from each triangle to the two volumes that it borders (Brakke, 1992; Lazar, 2011).

For explicit interface tracking strategies that hinge on computing winding numbers, the associated topology resolution strategies must also be generalized to multiple materials (Yang et al., 2018; Heiss-Synak et al., 2024). Rather than a single winding number at a point, one must consider a vector consisting of a distinct winding number per mate-

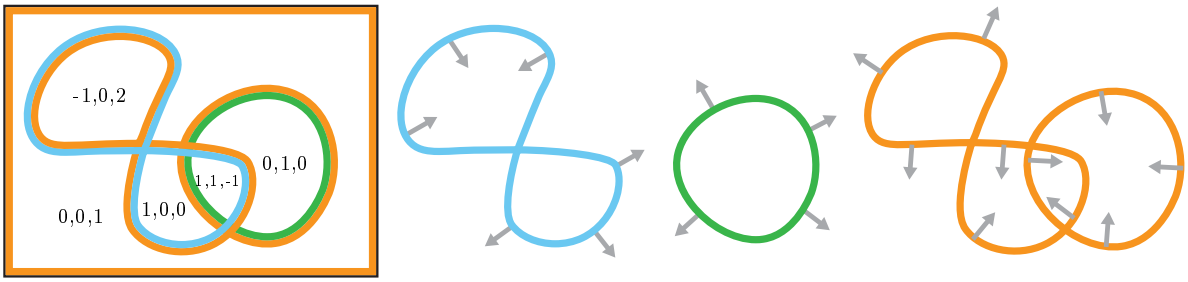

Figure 44: Left: A set of multimaterial interface curves representing cyan, green, and orange materials. The exterior ambient orange material is also visualized as a surrounding box. We have labeled each closed region with a material vector indicating the winding number for each material (ordered cyan, green, orange). There are topological inconsistencies, because not all closed regions are bordered on their interior by a single material. Right trio: Corresponding implied oriented meshes for each material.

rial, also referred to as a material vector (Figure 44). The values of this vector can, for example, be determined by (conceptually) forming a single *oriented* mesh per material from the double-sided representation described above, and computing the winding number separately on each such mesh, as before. The ambient exterior material is assumed to have a distinct winding number as well, which can be accounted for by implicitly assuming a surrounding border of that material (located at infinity). When the interfaces are in a physically valid state, the material vector should, for any given point not on an interface, have a value of 1 for the material containing that point and a value of 0 for all other materials, consistent with the expectation that a point should only belong to one material at a time. After advection of such a valid mesh, each individual material can exhibit the same types of topological errors discussed in §2.7, i.e., negative winding numbers ("inside out") and winding numbers greater than one ("multiply nested"). Instances of both can be seen in the top-left of Figure 44 (left) where the material vector is $[-1, 0, 2]$. However, a new type of topological inconsistency can also arise with multimaterial domains: more than one material can exhibit a positive winding number ("overlapping") at the same point. This effect can be seen towards the lower right of Figure 44 (left), where the material vector is $[1, 1, -1]$ and the cyan and green phases are overlapping. Notice that this overlap region violates the validity requirement that the interior of each closed region should be bordered by a single phase label. (A similar overlapping effect occurs in some multimaterial level set schemes (Losasso et al., 2006). However, the vacuum/gapping effects observed for multiple level sets that have drifted apart cannot occur for explicit interface tracking, since a triangle inherently carries both of its material interfaces in tandem.) Overlap cases occur, for example, when two materials collide to initiate multimaterial merging. In the case of Figure 44, to resolve the topology change, a new separating interface must be inserted into the lens-shaped overlap region and the existing redundant interfaces must be removed (the new cyan-green interface in Figure 45). The shape of this new interface patch may be application-dependent, but a natural choice is to place it midway between the previous interfaces; this corresponds to the boundary surfaces between cells in a generalized Voronoi diagram of the material interfaces bordering the overlap region. This equidistant, Voronoi-type approach is also consistent with prior work on multimaterial level set schemes (Losasso et al., 2006; Saye and Sethian, 2012).

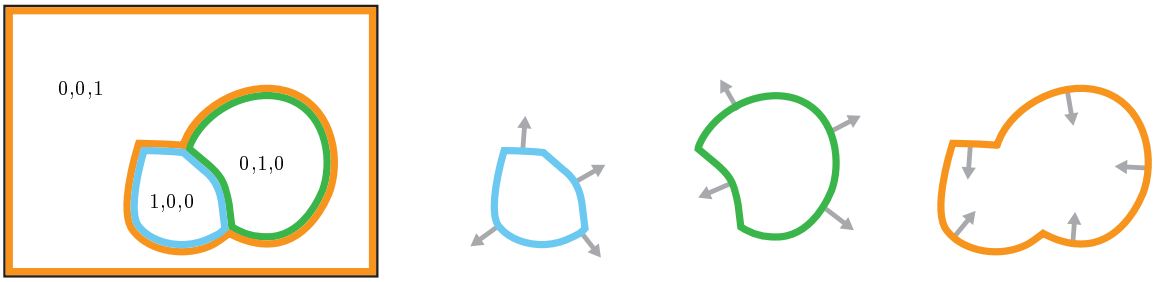

Figure 45: Left: The multimaterial interface curves from Figure 44 after resolving the topology. The winding number vectors now all have only a single active material per region and each region is bordered on its interior by a single material. Right trio: The corresponding implied oriented meshes for each material. A new interface has been formed where the cyan and green regions previously overlapped.

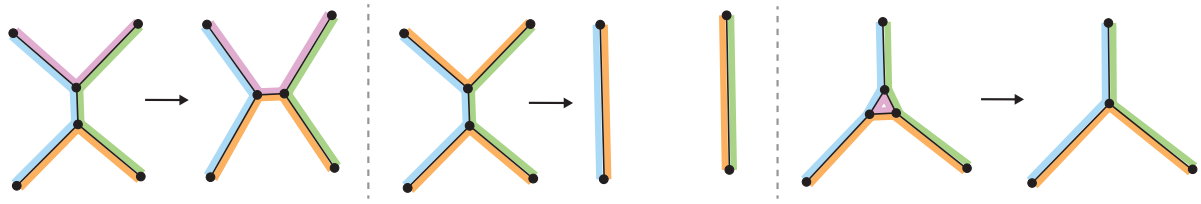

Figure 46: Three discrete 2D topology change operations, with edge color labels indicating material types. Left: A T1 process (grain switching) rearranges the local connectivity, when all four materials differ. Center: A T1 process with the same material (orange) on opposing sides causes those regions to become joined. Right: A T2 process deletes a small cell (pink).

6.2. Materials Science

Multicellular interfaces arise frequently in material sciences, often for the evolution of grain boundaries in polycrystalline structures (Gottstein and Shvindlerman, 2009) or the dynamics of interconnected bubbles in foams (Weaire and Hutzler, 1999). Recall that, in this context, some Lagrangian schemes in the style of explicit interface tracking are referred to as vertex models, although front tracking is also a common term. All of the methods in this domain use local explicit topology change strategies.

6.2.1. Two Dimensions

The earliest 2D vertex-based numerical model for multicellular interfaces was proposed by Weaire and Kermode (1983) to simulate soap foams (or froths). The foam is described by a collection of vertices representing (only) the triple junctions, which are connected together by circular arcs representing the cell faces. For the T1 process, they identify a short edge that separates two phases that are also incident on two other phases; the operation is performed by adjusting the local connectivity to disconnect the two materials sharing the edge, and connect the other two opposing materials with a new short interface segment (as in Figure 46, left). T2 processes are detected by checking for triangular cells with near-zero area; they are performed by replacing the small cell with a single vertex, and adjusting the neighborhood connectivity accordingly (as in Figure 46, right). Note that small cells with four or more sides can also collapse by first undergoing one or more T1 processes; this ultimately transforms them into triangular cells, which can then be deleted with the above T2 process. The original code (2D-FROTH) implementing this method was documented in detail in a later publication (Kermode and Weaire, 1990).

Soares et al. (1985) proposed a similar approach for grain growth using straight-line cell edges instead of circular arcs, but with the same T1 and T2 operations. Frost et al.

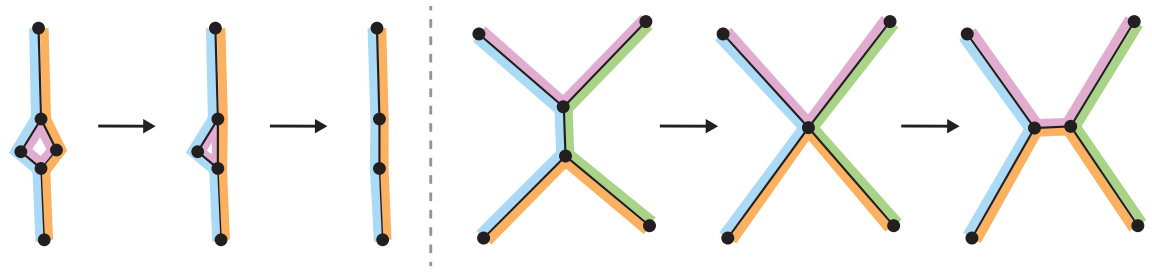

Figure 47: Left: In 2D, the use of curved or subdivided cell faces allows for two-sided grains (pink region). The deletion of such a cell is sometimes called a T3 process. Right: Some models of T1 processes (Brakke, 1992; Kinderlehrer et al., 2006) explicitly allow an unstable quadruple junction to form (through collapsing short edges), before then splitting it back into two stable triple junctions in a separate operation. Compare with Figure 46 (left) where a short edge immediately triggers the rearrangement, without entering the intermediate quadruple junction state.

(1988) likewise used the same operations, but with parabolic arcs between vertices.

Similar to Soares et al. (1985), Nagai et al. (1988) also used straight-line cell edges, but treated T1 processes in a slightly more general way. They identify a short edge, but reconnect the surrounding interfaces in a manner dependent on the material labels of the surrounding phases. The result is that the two phases initially sharing the central short edge become disconnected, and either: (1) the two previously disconnected regions become connected by a new shared interface, if they have distinct labels, per Weaire and Kermode (1983) (Figure 46, left) or (2) the two previously unconnected regions join into a single region if they have matching phase labels (Figure 46, center). Their T2 process is triggered when all three edges of the small triangular region fall below a length threshold, deleting the three edges, and collapsing the relevant three vertices into a single new vertex. Enomoto et al. (1989) did likewise, but in a subsequent paper Kawasaki et al. (1989) instead triggered a T2 process even if only one of the three edges becomes too short, consistent with the earlier approach of Weaire and Kermode (1983).

All of the above methods use vertices only to represent the triple junctions, and represent each entire cell face by a single arc or line segment. Fradkov et al. (1993) independently developed a similar approach for topology changes, but rather than use a single edge or arc per cell face, their cell faces consist of a sequence of multiple edges (and vertices). Doing so allows for general curved cell faces and makes the method more closely resemble a typical explicit interface tracking approach. Bronsard and Wetton (1995) described a similar method and additionally accounted for interactions with potentially curved domain boundaries. Independently, Weygand et al. (1998) also added extra vertices along cell interfaces. Using more finely sampled interfaces allowed these methods to distinguish between a T2 process, which collapses a cell bordered by three other phases, from a *T3* process, which collapses a cell bordered by just two phases (Figure 47, left). The latter can arise only with curved interfaces. All three groups above demonstrated simulations of such a case. For simplicity, we will typically refer to any multimaterial instance of cell deletion as a T2 process, since in practice they are often handled similarly.

Kinderlehrer et al. (2006) adopted a T1 process in 2D similar to the methods above, but their approach explicitly allows the mesh to enter a temporary state exhibiting a quadruple junction, rather than instantly jumping from two existing triple junctions to a new pair of triple junctions (Figure 47, right). Brakke (1992) also used the same approach

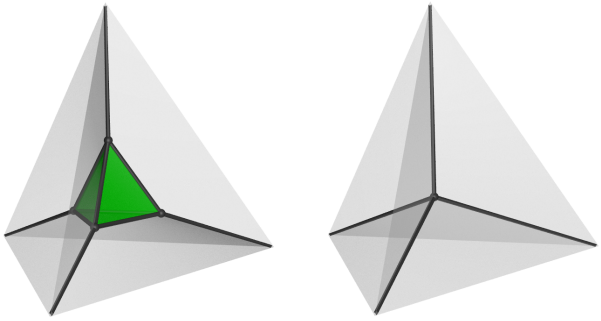

Figure 48: The 3D T2 process proposed by Nagai et al. (1990). Left: A small region, bordered by four adjacent phases and consisting of a single tetrahedron, is identified for collapse (green). Right: The tetrahedron is removed and replaced with a single vertex.

in 2D much earlier, but since his work is best known for its contributions to the 3D case we discuss it in the next section instead.

6.2.2. Three Dimensions

Nagai et al. (1990) were the first to describe a three-dimensional generalization of these elementary operations for vertex models, similarly tracking only quadruple junction vertices explicitly and assuming unsubdivided polygonal cell faces. For the T2 process, they identify a region consisting of one tetrahedron with at least one short edge, and collapse that tetrahedron into a single vertex (Figure 48), directly analogous to the 2D operation. Handling the 3D generalization of T1 processes is more intricate. Their proposed recombination operation (which we will call a "T1 flip" for brevity) is reversible, and either creates or deletes a triangular face between two phases (Figure 49). In doing so, it connects two phases that were previously not in contact (or disconnects two connected phases in the reverse case). In structure, the forward operation is similar to a merge operation in the two-phase case, except that here the two regions coming into contact are surrounded by three other regions, the two colliding vertices were already indirectly connected by an existing edge, and the regions do not fully merge but remain separated by the creation of a new interface. A similar analogy can be made for the reverse operation and two-phase splitting. (This analogy suggests possible alternative treatments of merging and splitting in the two-phase case, which are closely related, but not identical, to those of Lachaud and Montanvert (1996).)

Fuchizaki et al. (1995) subsequently observed that the quality of results from the preceding simulation method was limited by the assumption that cell faces consist of a single polygon loop with no explicit triangulation. Similar to two-dimensional methods that subdivide cell faces (i.e., edges), these authors introduced a single additional vertex at the center of each 3D face and showed that this choice achieves improved results (triple line cell edges are still assumed to each be a single unsubdivided straight line segment). The method of Fuchizaki et al. (1995) was used by Weygand et al. (1999) for further study of grain growth.

The T1 flip operation described above transitions between a state with two quadruple junction vertices and one with three quadruple junction vertices, and all the incoming curves are triple junction edges. This approach thus altogether avoids higher valence vertices and edges, as these can introduce greater complexity and are uncommon in physical foam and grain settings. Another family of approaches, exemplified by the pioneering *Surface Evolver* method of Brakke (1992), instead allows high valence configurations to

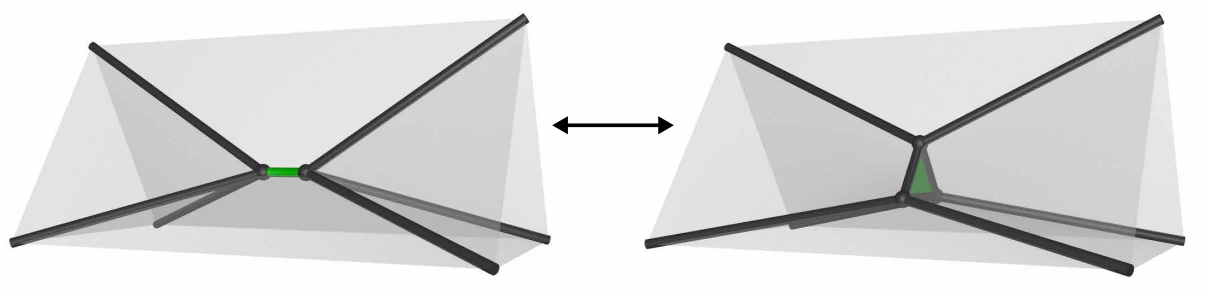

Figure 49: The pair of operations proposed by Nagai et al. (1990) to handle 3D T1 processes, which are invertible; we refer to these as forward and backward T1 flips. From to left to right, a short (green) edge connecting two quadruple junction vertices is replaced with a small (green) triangle that connects the left and right regions. The inverse operation (right to left) eliminates the connecting triangle and inserts an edge. (We show some faces as quadrilaterals, but models based on triangle meshes rather than polygon meshes will have the quadrilateral faces divided into triangles.)

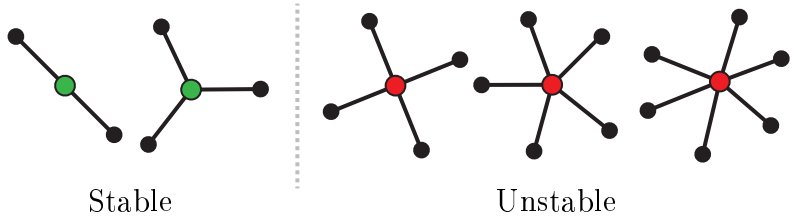

Figure 50: Example configurations demonstrating the two possible stable (green) vertex configurations and three of the infinitely many possible unstable (red) vertex configurations. In 2D, stable vertices are simply those with either two or three incident edges.

arise naturally through the collapse of short edges during mesh improvement. Distinct topology resolution operations are then perfomed to effectively complete the T1 process and allow the mesh vertices and edges to escape back to a lower valence state. Brakke refers to such operations as vertex or edge "popping". To understand this approach, it is useful to characterize *stable* and *unstable* vertex configurations of the non-manifold mesh interface (others have called these configurations regular and irregular (Da et al., 2014), or generic and nongeneric (Meyer et al., 2008)). Intuitively, a geometry is considered stable if its local topology (but not necessarily its angles) corresponds to a stable configuration under Plateau's laws governing soap foams. Thus, in 2D, a stable vertex has only two or three incident edges, and an unstable one has four or more, as shown in Figure 50. A stable vertex in 2D is thus in contact with either two or three cells, but no more. However, in 3D, vertices are stable if they either lie on a cell face where two cells meet, lie on a triple curve where three cells meet (analogous to Plateau borders in foams), or lie at a quadruple point, where four cells meet. Any vertex with a higher valence configuration is unstable and must be "popped" to transform it into a nearby stable state (Figure 47, right illustrates such a popping in 2D). Analogously, in attempting to reduce surface area, a physical foam will instantly flow to separate such topologies into stable configurations. Examples of 3D stable and unstable configurations are shown in Figure 51.

Having laid out this context, we can discuss Brakke's influential Surface Evolver, also presented in the early 1990s (Brakke, 1992, 2004). Surface Evolver is a software package aimed at computing minimal surfaces and other energy-minimizing shapes subject to constraints, in 2D and 3D. Surface Evolver has been widely used, including in simulations of foam dynamics and grain growth (Marthinsen et al., 1996; Monnereau et al., 1999; Wakai et al., 2000). It assumes both edges and faces are subdivided into a finer, more general

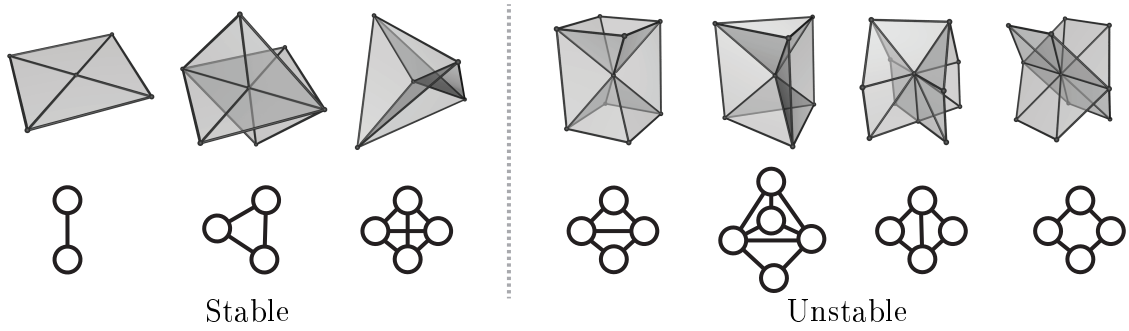

Figure 51: Example configurations demonstrating stable and unstable configurations in 3D. The region graph for the central vertex of each mesh is shown beneath the mesh (Da et al., 2014). Stable topologies exhibit complete region graphs; unstable topologies do not. (We define the region graph concept in §6.3.2.)

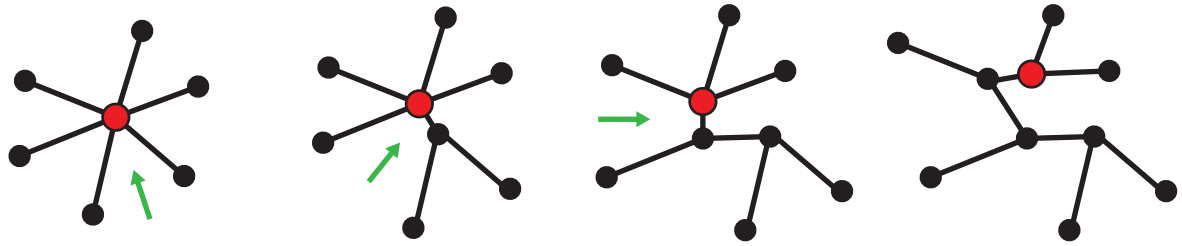

Figure 52: In 2D Surface Evolver (Brakke, 1992), a vertex with high valence (red disk) can be iteratively popped until only triple-point vertices remain. The green arrow indicates the incident phase that is next to be "pulled off" of the central vertex at each step of the process. (The ordering of which region to pull off next is chosen arbitrarily here.)

triangulation than in the vertex models discussed above, akin to typical explicit interface tracking schemes. As mentioned, Surface Evolver handles T1 processes by allowing unstable vertices to form through edge collapses and resolving them back to stable states via vertex and edge popping routines. This multiphase stable-to-unstable-to-stable topological sequence has conceptual similarities to some of the two-phase topological operations of Brochu and Bridson (2009) and Da et al. (2014), which proceed through a manifold-to-non-manifold-to-manifold sequence; the stable-to-stable T1 flip of Nagai et al. (1990) is likewise analogous to the direct manifold-to-manifold two-phase approaches of Lachaud and Montanvert (1996) and others.

In Surface Evolver's 2D version, vertex pops are straightforwardly achieved by duplicating and pulling apart the unstable vertex, while inserting a separating interface edge and adjusting the local connectivity (as in Figure 47, right, and the later method of Kinderlehrer et al. (2006)). Higher valence unstable vertices are processed by iterating this operation until no unstable vertices remain (Figure 52). In 3D, however, popping must account for both unstable quintuple (and higher) vertices and unstable quadruple (and higher) edges, where the numbering indicates the count of incident volumetric regions (not edges or faces); many complex configurations are possible. Brakke's edge popping in 3D (Figure 53) proceeds such that in cross-section it mimics 2D vertex popping (recall Figure 47, right): an unstable vertex lying along a quadruple edge curve is duplicated and pulled apart, creating a new triangulated interface joining two previously disconnected phases, with appropriate adjustments to the surrounding connectivity. This process can be applied along the entirety of a given quadruple-line edge sequence,

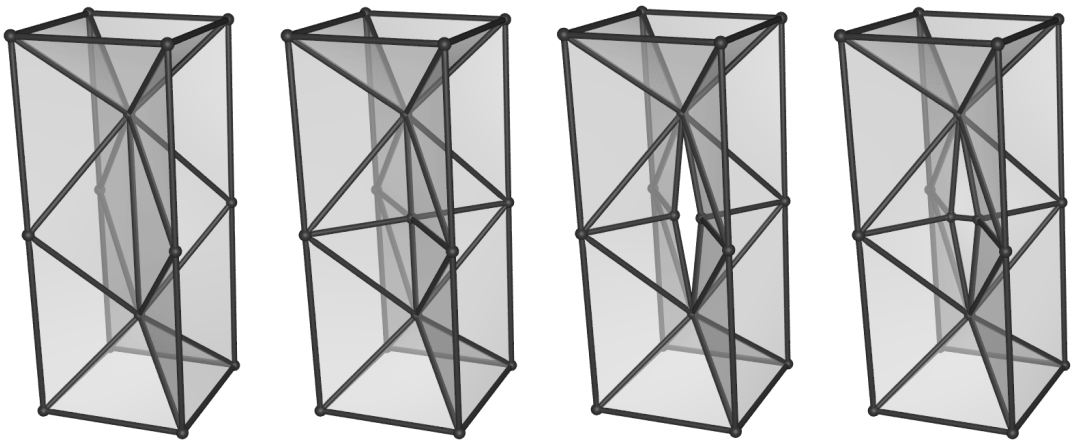

Figure 53: The edge pop operation in Surface Evolver (Brakke, 1992). A quadruple-curve (which borders four phases) is an unstable configuration (far left). If there are no vertices along the quadruple-curve (i.e., it consists of a single edge), the edge is first split (center left). The vertex on the edge is duplicated and separated (center right). Two new triangles are inserted (far right) to maintain separation between the front and back phases. If the original quadruple-curve consists of several consecutive edges, this process can be generalized to treat the whole sequence of vertices at once.

"opening it" like a zipper into two nearby triple lines. To handle 3D vertex popping, Brakke provided a number of special case treatments depending on the local configuration. In particular, this set includes the T1 flips of Nagai et al. (1990), but processed incrementally in that the unstable intermediate configuration is explicitly entered and then resolved. However, Brakke also described a general case treatment for arbitrarily high-valence unstable vertex popping, carried out as a single operation, that proceeds as follows: after first popping all edges, a small spherical cavity is inserted to surround and replace the target popping vertex, and all incoming edges and triangles are clipped against it, thereby creating a new polyhedral "bubble" having only stable vertices and edges. Lastly, one of the triangulated faces of the spherical cavity is removed to merge it with one of the incident phases, eliminating the distinct bubble. Brakke observes that despite being effective, this process creates an excess of local geometry, so where possible the special case treatments are preferred. Surface Evolver also supports a so-called "disjoining pop" to separate a single non-manifold vertex shared by two otherwise disjoint interfaces; a physical analogy is to a catenoid soap film (with or without a central interface film) pinching off. This disjoining pop operation is a much earlier precursor of the vertex separation later used by Brochu and Bridson (2009) in the two-phase setting (Figure 35). Alternatively, such a non-manifold vertex can also be instead "opened", either creating a hole or a new film depending on whether the materials of the joined regions are the same or different (Figure 54). Finally, while most of Surface Evolver's popping operations are invoked once an unstable state is entered, it also optionally supports T1 flips *without* traversing the unstable configurations by checking for small edges or faces (Brakke, 2004), in the manner of Nagai et al. (1990).

In an appendix of their work on simulating grain growth Mora et al. (2008) illustrated their topological transformations, which are shown to be similar to Surface Evolver in that they also traverse through the unstable states, rather than jump past them. The operations they describe are referred to as a triple line collapse (i.e., incremental forward T1 flip), grain *boundary* collapse (i.e., incremental reverse T1 flip), and grain collapse (i.e., T2 process). They do not discuss higher valence cases.

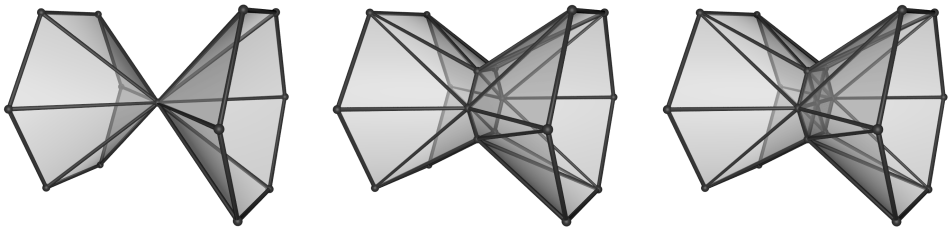

Figure 54: Surface Evolver supports an operation that can (in a single step) either transform a singular vertex (left) into a merged tube (center) or install a new separating interface (right), depending on whether the incident materials are the same or different. (It also supports separating the singular vertex as in Figure 35.)

Syha and Weygand (2010) proposed another approach similar to Surface Evolver, but they described in greater detail the necessary geometric procedures and handled arbitrary valence unstable vertices in a different, more general way. Their approach involves essentially three core operations: (1) joining of vertices (i.e., edge collapse), (2) an operation to detach one incident cell from an arbitrary valence unstable vertex, which allows for general T1 rearrangements, and (3), like Surface Evolver, a special case operation for T1 flips (forward and backward). T2 processes are not handled separately, but instead occur naturally when an edge collapse yields a redundant coincident pair of faces and one of them is removed. The proposed grain detachment process can be viewed as a generalization of the non-manifold vertex separation of Brochu and Bridson (2009): the vertex in question is duplicated, the resulting two vertices are separated by a small distance, and the incident triangulation is adjusted. The additional complexity in the multimaterial case is that as the duplicate vertex is separated out, a new edge must still connect the two vertices together, and new triangles must be generated for each incident interface to close the holes that would otherwise be created (see Figure 59). Following Da et al. (2014), we refer to this operation as (multimaterial) vertex separation. To decide which grain should detach, Syha and Weygand (2010) evaluate the physical forces in the grain boundary problem and separate the one that is undergoing the greatest force. This process can be iterated until a stable state is achieved, similar to how vertex popping is iterated in the 2D version of Surface Evolver (recall Figure 52). Syha et al. used their method to carry out simulations of the coarsening of up to 2500 grains.

In contrast to most multiphase interface methods based on triangulated surfaces, Kuprat and colleagues adopted a volumetric tetrahedral finite element approach for grain boundary evolution (Kuprat, 2000a; Kuprat et al., 2003; Kuprat, 2000b). Each tetrahedron possesses a material label, and the code tracks the bordering interfaces and curves between different materials as the tetrahedra evolve. Tetrahedral remeshing is performed in the material interiors to ensure well-shaped tetrahedra over time, and this can lead naturally to degenerate configurations arising (i.e., unstable interface vertices or edges, and small material regions). However, their strategy for performing topological operations, rather than modify the mesh structure, is to perform a relabeling of some incident tetrahedra whenever problematic configurations are detected. For example, since a vertex incident on five materials is unstable, the incident tetrahedra of one of the incident materials are simply relabeled to agree with one of the other incident materials (Figure 55, middle). The "losing" and "winning" materials in their method are chosen based on total solid angles subtended by the tetrahedra of each material at the unstable vertex,

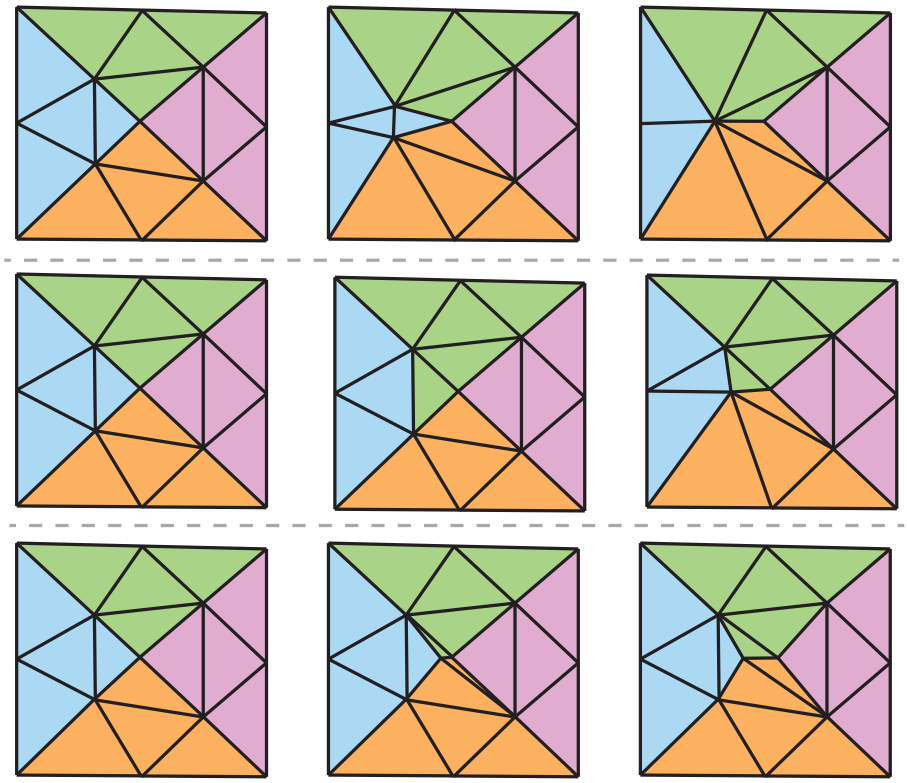

Figure 55: T1 processes with multimaterial volumetric meshes, illustrated in 2D with mesh evolution from left to right. Top: Under the DSC method (Misztal, 2010), an unstable vertex is eliminated only once all the incident elements of a particular material collapse. Middle: Under a scheme like that of Kuprat (2000a), the unstable vertex is resolved instantly by explicitly relabeling one of the incident materials (cyan to green in this case). Bottom: A (hypothetical) volumetric approach mimicking the vertex separation/popping used in surface-based schemes like Surface Evolver (Brakke, 1992).

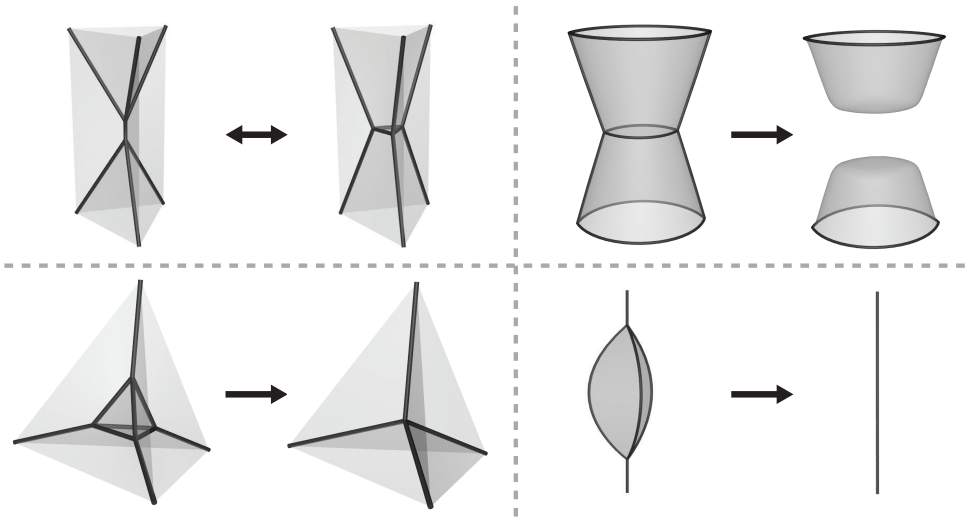

Figure 56: The five core operations that Lazar describes for topology change in grain growth. Top-left: edge replacing a triangle and triangle replacing an edge for T1 processes (the same as the T1 flip in Figure 49). Top-right: Removal of two-sided face (i.e., a lens-shaped face having two curved edges). Bottom-left: Collapse of a tetrahedral cell into a quadruple junction (the same as the T2 process in Figure 48). Bottom-right: Collapse of a three-sided cell into a triple curve.

from energy considerations; that is, under surface-tension-like forces, they presume the narrowest region to be the one to be "pinched off". An analogous process is applied to unstable edges based on the angles each incident material makes at the edge. Together these operations can reproduce arbitrary T1 processes. To minimize the region of influence of these relabeling operations, they first perform local mesh refinement around the affected vertices or edges. For T2 processes, they traverse the mesh to detect small shrinking material regions, and relabel the associated tetrahedra to match an incident material instead (again selected based on incident solid angles). Notably, this approach of using a volumetric mesh and processing topology change by relabeling simplices has some similarities to (and predates) the kinetic triangulation/tetrahedralization schemes we discussed earlier (Pons and Boissonnat, 2007b; Misztal and Bærentzen, 2012; Misztal et al., 2013) and which also support multiple materials.

In recent work on grain growth and curvature flow using a more typical interface triangulation approach, Lazar et al. highlighted the importance of additional cases of topology change (albeit also mentioned previously by others) that can arise in three-dimensional geometries when interfaces are allowed to be curved: the separation of a two-sided face (a lens-shaped face with only two distinct border curves) and the collapse of a cell with only three (curved) interfaces into a triple curve (Lazar, 2011; Lazar et al., 2011). This yields their menu of five unique topological operations that occur in the grain growth setting: (1) replacing an edge with a triangle, (2) replacing a triangle with an edge (i.e., the inverse of the first), (3) removal of a two-sided face, (4) the collapse of a tetrahedron into a single vertex, and (5) the collapse of a volumetric region with only three interfaces into an edge. These are diagrammed in Figure 56. Moreover, more complex configurations can be reduced to combinations of these five. They observed that these five could be reduced geometrically to three operations, but did not do so. The two-dimensional variant of their algorithm similarly supports a two-sided cell collapsing into an edge (also called a T3 process).

Lazar commented that Surface Evolver cannot handle all configurations and can re-

quire manual intervention to resolve them. Lazar's vertex model / front tracking approach successfully performed simulations of up to 100,000 individual grains, far surpassing earlier methods in scale and robustness.

6.3. Computer Vision and Computer Graphics

The vertex models from materials science were designed primarily to address networks driven by curvature flow and related energies; these rarely, if ever, exhibit multimaterial merging (Figure 41). However, these behaviors do arise in multiphase fluid animation and other visual computing tasks, leading researchers in graphics and vision to investigate the full slate of multimaterial operations shown in Figure 41.

6.3.1. The Deformable Simplicial Complex, Revisited

The local explicit 2D kinetic Lagrangian triangle mesh scheme of Pons and Boissonnat (2007b), discussed earlier, was shown to straightforwardly support more than two materials. Rather than labeling each triangle as either inside or outside, an arbitrary set of labels can be used, and the core algorithm otherwise remains the same. Similarly, in three dimensions, the Deformable Simplicial Complex method of Misztal (2010), did not initially demonstrate more than two phases; however, in a later application to multiphase flow they showed it can likewise be done by simply using a larger label set (Misztal et al., 2013). More recently, they applied it to multiphase volume segmentation of CT scan data (Nguyen et al., 2018). They clearly demonstrated multimaterial merging, but did not specifically discuss T1 processes. Since DSC handles all topology change automatically by remeshing when elements collapse, a T1 process will complete when some of the incident elements become flat enough (see Figure 55, top); however, this may cause unstable geometric configurations to linger longer than desired, compared to more direct T1 treatments. Consider an analogy to the two-phase case: Brochu and Bridson (2009) perform direct vertex separation to immediately split apart non-manifold vertices, whereas for Razizadeh et al. (2018), splitting at a non-manifold vertex only occurs when an incident tetrahedra-shaped region eventually collapses flat and is deleted. One way to incorporate a direct T1 treatment into DSC would be the approach of Kuprat (2000a), since both use labeled tetrahedral meshes; another would be to develop volumetric analogs of the surface operations of Surface Evolver (Brakke, 1992) or Los Topos (Da et al., 2014) (see Figure 55, middle and bottom).

Kuprat (2000a)

6.3.2. Los Topos

Inspired by both Surface Evolver (Brakke, 1992) and El Topo (Brochu and Bridson, 2009), the local explicit *Los Topos* algorithm of Da et al. (2014) extends the two-phase collision-safe incremental surgery methodology of Brochu and Bridson (2009) to support the full suite of multimaterial operations (Figure 41) in three dimensions: multimaterial merging, T1 processes, and T2 processes.

They describe two possible approaches for multimaterial merging. First, they showed that one can generalize the merging approach of Brochu and Bridson (2009), i.e., identify two edges in close proximity, delete the two-triangle patch surrounding each edge, and install a new 8-triangle connecting tube (Figure 36). In the multimaterial case where the colliding materials differ, however, one must retain one of the existing two-triangle patches and update their material labels, to produce the necessary material-separating film. Second, they introduce the new snapping-based merging strategy we described

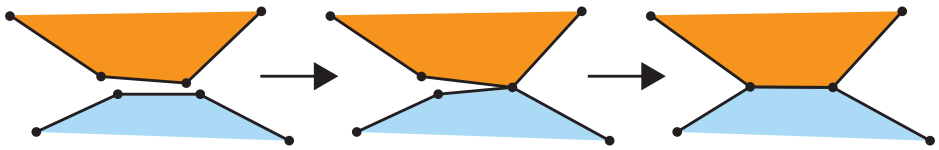

Figure 57: The merging approach of Da et al. (2014), based on vertex snapping, straightforwardly supports the multimaterial case (illustrated here in 2D). If the two materials differ as shown here, the separating interface is retained; otherwise it is discarded as in Figure 37.

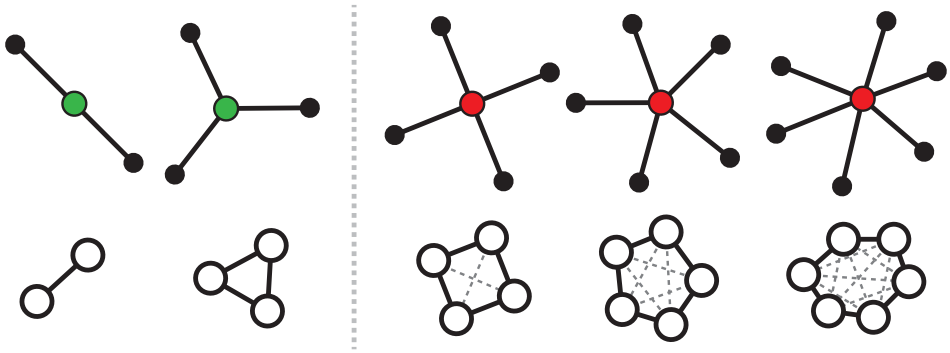

Figure 58: Da et al. (2014) define the notion of a region graph for a vertex (bottom row), which exhibits a dual structure with respect to the mesh neighborhood of a vertex (top row). Each region graph for a vertex has one node per incident material (empty black circles) and one arc (black segments) per shared interface between two materials. If the region graph has "missing" arcs (dashed gray lines), the vertex is unstable, and each missing arc corresponds to a pair of phases that meet at the vertex but do not share an interface. Thus each arc-pair indicates where possible multimaterial vertex separation can be applied (i.e., to "pull off" one or the other of its two corresponding materials.

earlier in the two-phase context (§5.2). This approach can be easily modified for multimaterial settings: when a vertex-snap induces a degenerate non-manifold (coincident) triangle configuration, the two colliding materials are checked. If they are the same, the coincident triangles are deleted to yield an ordinary merge (Figure 37); if they differ, one triangle is deleted and the other has its labels adjusted to yield a multimaterial merge (illustrated in 2D in Figure 57). (This approach is also consistent with the later method of Razizadeh et al. (2018)) for the two-phase case.)

To treat T1 processes, Da et al. (2014) introduce a variant of the multimaterial vertex separation operation of Syha and Weygand (2010), to similarly avoid handling many distinct special cases and to support arbitrary phases in mutual contact. They characterize stable and unstable configurations at a vertex by constructing what they call a local region graph for a vertex. The region graph of a vertex consists of one graph node per phase and one graph arc between a node pair if the corresponding phases share a triangle in three dimensions (an edge in two dimensions). *Incomplete* region graphs (where not all nodes are mutually connected) correspond to unstable configurations, i.e., there are phases that touch the vertex but do not touch one another. Conversely, vertices in stable configurations have *complete* region graphs. Thus, a vertex on a cell face is stable, a vertex on a triple curve is stable, and a quadruple junction vertex (where all four materials are in mutual contact) is stable, but all other high-order configurations are unstable. The region graph concept is illustrated in Figure 58 for the 2D case, and Figure 51 for the 3D case. Da et al. (2014) further observe that the set of possible vertex

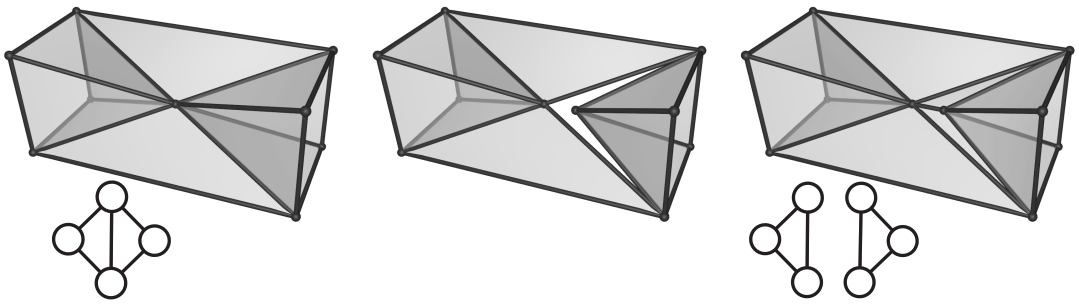

Figure 59: Multimaterial vertex separation on an unstable vertex (left), as described by Syha and Weygand (2010) and Da et al. (2014). The unstable vertex is duplicated, the new duplicate vertex is pulled apart by a small distance taking the attached triangles with it (center). If necessary (depending on the incident structure and material labels, as in this case) a new edge and new triangles are added to close any undesired gaps this creates. Notice that although the original vertex is incident on only four phases, it is still unstable according to the region graph classification of Da et al. (2014), since the left and right regions share only the vertex. The region graphs of the two resulting vertices are complete, indicating stable configurations.

separations is determined by the set of pairs of phases that are not connected by an arc in the region graph (i.e., pairs of regions that share the unstable vertex but no triangles). While trivial in two dimensions, they argued that this perspective provides insight into the three-dimensional case, which can otherwise be more difficult to reason about. For example, Figure 59 illustrates a 3D case where a vertex bordered by four regions is correctly identified as unstable and hence eligible for vertex separation, according to the region graph classification (the left and right regions do not share any triangles), but might erroneously have been assumed stable if one simply counted the number of incident phases.

They use the region graph to identify unstable vertices and candidate region pairs for separation, and process them via multimaterial vertex separation, iterating until only stable vertices remain, much like Syha and Weygand (2010). Refer to Figure 59. Since multiple possible candidate vertex separations can be applied at an unstable vertex, rather than relying on the domain-specific physics of grain boundaries as in prior work, Da et al. delegate the decision of which separation to perform (or to perform first) to a user-provided callback function for greater flexibility. The decision could be made, for example, by examining the local flow velocities for divergence/convergence in some axis, or by evaluating metrics involving the local mesh geometry or physics (e.g., energy, angles, etc.).

Since their overall algorithm is designed to employ only localized atomic operations, and multimaterial vertex separation requires moving only a single vertex (and its incident edges and triangles), Los Topos naturally retains the collision-safety guarantees of El Topo (Brochu and Bridson, 2009) by testing for intersections and rolling back in case of failure. Unlike Syha and Weygand (2010), Da et al. do not include a distinct special case T1 flip operation, thus avoiding the need to check it for collision-safety. Finally, we observe that many of the materials science methods for foams and grain boundaries rely on specific physics to rule out unstable vertices and edges, allowing them to be algorithmically sidestepped altogether (e.g., by T1 flips and the other operations described by Lazar (2011)). However, it is conceivable that other applications of interface tracking could fundamentally require so-called "unstable" (high-valence) configurations to occur, either

transiently or persistently. Given the flexibility with which it treats T1 processes, Los Topos could be extended to handle these if needed.

An interesting and seemingly unexplored alternative to their treatment of high valence unstable vertices, rather than fully disconnecting region pairs that do not share an arc in the region graph, would be to instead fully *connect* the relevant pair with a new interface, by generalizing one of Brakke's disjoining pop operations (specifically Figure 54, right) to allow an arbitrary number of incident regions. (This could also be considered as a kind of partial high-valence generalization of the forward T1 flip, i.e., the right half of Figure 49.) However, this treatment would involve a potentially larger change to the local geometry, increasing the complexity of the associated collision-handling.

6.3.3. Image Processing: A Generalization of Snakes

In a multimaterial generalization of active contour (snakes) methods for image processing, Benninghoff and Garcke (2014) concurrently proposed topological manipulations in two dimensions that are similar to the 3D ones of Da et al. (2014). They describe alongside2D multimaterial merging operation and a standard 2D T1 process, albeit without naming them as such. For multimaterial merging, they detect when two curves are close, and snap one pair of their vertices together to create a temporary quadruple junction. Then the quadruple junction is separated by a short edge into two triple junctions, as in the vertex popping of 2D Surface Evolver (Brakke, 1992). Depending on whether the colliding materials are the same or different the new interface they share is either deleted or maintained, respectively. T1 processes are handled following 2D Surface Evolver and Kinderlehrer et al. (2006). Benninghoff and Garcke (2014) do not discuss handling of valences beyond four, i.e., quintuple junctions and higher.

6.3.4. Implicit Multimaterial Methods

According to our earlier classification scheme (§2.8), the volumetric mesh methods of Kuprat (2000a), Pons and Boissonnat (2007b), and Misztal (2010), the interface mesh method of Da et al. (2014), and the various vertex / front-tracking models from materials science all make use of local explicit mesh surgery approaches for topology change. However, as in the two-phase setting, implicit topology change methods can also be deployed in the multimaterial case, with the use of a background grid again being a popular strategy.

Li et al. (2014, 2016) adapted the semi-Lagrangian contouring (Bargteil et al., 2006) approach to the multiphase setting. This primarily requires modifying the backtraced distance field query to be aware of the material that the query point lands in, so that multimaterial contouring (i.e., isosurfacing) can be performed. They perform contouring using a multimaterial variant of marching tetrahedra, which is similar to the *Lattice Cleaving* multimaterial tetrahedral mesh generation method (Bronson et al., 2013). However, like the earlier two-phase SLC schemes, the method of Li et al. does not advect the mesh forward in time, so we do not formally classify it as an explicit interface tracking scheme.

Yang et al. (2018) proposed a hybrid scheme that employs the regional level set (RLS) method (Zheng et al., 2009) (a multimaterial level set scheme) and explicit meshes in separate spatial regions. They use a distinct manifold mesh per material rather than a connected non-manifold mesh, and rely on the RLS to handle regions with nontrivial topology. They generalize the raycasting approach of Müller (2009) to multiple materials to identify the material labels and locate invalid overlapping regions (as discussed in

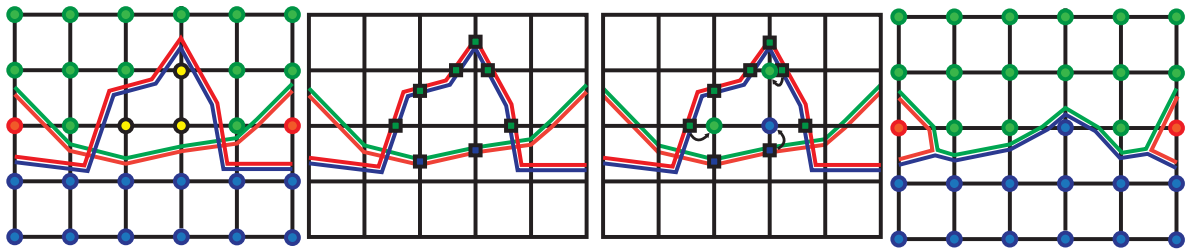

Figure 60: The implicit multimaterial topology change process of Heiss-Synak et al. (2024), illustrated in simplified form in 2D. Far Left: After advection, a region of a multimaterial domain is in an overlapping state. Raycasting along grid lines determines the winding number vectors for each node. Where the material is ambiguous, the node is shown as a yellow circle. Center Left: The edge-crossings (edge/grid intersections) lying between unambiguous and ambiguous nodes are determined, and assigned the unambiguous material. Center Right: Material labels are propagated to assign materials to the ambiguous nodes, based on closest distance. Far Right: With all nodes now assigned unambiguous material labels, a multimaterial marching method can be used on the grid to build a new mesh patch.

§6.1). They leverage the distance values encoded in the regional level set to determine the correct material in the overlap region, using an approach similar to the multiple level set projection scheme of Losasso et al. (2006). A marching cubes approach is used to reconstruct the interface mesh where needed, which is then stitched into place, similar to the two-phase method of Wojtan et al. (2009). Despite working with manifold meshes, Yang et al. (2018) observe that multimaterial extensions of marching cubes can still be necessary to avoid introducing undesirable gaps at junctions between multiple materials, although their results did not do so.

The method above can be seen, in part, as a generalization of the localized implicit remeshing approach of Wojtan et al. (2009) to multiple materials; however, it suffers from the need to make use of both a persistent level set representation and a persistent (multi-layered) triangle mesh representation, across time steps and in different spatial regions. The recent method of Heiss-Synak et al. (2024) more faithfully extends the method of Wojtan et al. (2009) to multiple materials: they use a labeled non-manifold triangle mesh as the primary representation everywhere, and rely on implicit, grid-based reconstruction only temporarily when and where needed to correct proximities and topological inconsistencies in the mesh. Their pipeline proceeds as follows (also illustrated in 2D in Figure 60). After advecting the multimaterial interface mesh, they use raycasting to compute the intersections between grid edges and mesh triangles, and to determine the material (winding number) vector for each grid node. They use this information to detect non-physical overlap regions and cells where the mesh and its grid-sampling meaningfully disagree. They then clip and retriangulate the triangles bordering the cells identified for repair, advocating a constrained Delaunay (re)triangulation approach for improved robustness and to avoid creating zero-area triangles, inspired by the work of Pavić et al. (2010) on mesh Booleans. A new interface mesh patch is constructed in overlap regions given the corrected labels for each node, with the new node labels being determined via a distance-based marching approach. The new multimaterial interface is constructed in overlap regions via simplified multimaterial marching cubes cases, and mesh improvement is performed to yield a smoother resulting interface. Finally, they stitch the new interface back together with the outer clipped triangles. They demonstrate a variety of impressive large-scale applications, including soap bubble dynamics (up to a thousand bubbles), normal flow (offsetting), and Boolean-like combinations of multiple shapes of

differing materials, observing improved efficiency and robustness compared to the Los Topos method of Da et al. (2014).

6.4. Computational Mathematics

The 2D MARS method of Zhang and colleagues was recently extended to handle the multiphase case of three or more materials (Tan et al., 2026). To retain high order accuracy while avoiding overlaps or vacuums between materials, they use cubic spline curves which are carefully stitched together at kinks/junctions and shared by incident material pairs. They do not consider topology changes, but highlight them as an important avenue for future work.

6.5. Cell Biophysics

In the early 2000s, the vertex models developed in materials science began to be adapted for simulating biological behaviors of cells. Nagai and Honda (2001) considered the growth of epithelial tissue in a two-dimensional model, adopting the earlier 2D topology change treatment of Nagai et al. (1988) but changing the physical model. In the same manner, Honda et al. (2004) proposed a three-dimensional model for cell dynamics, adopting the corresponding 3D topology change mechanisms of Nagai et al. (1990) and Fuchizaki et al. (1995). In the biology setting, T1 transitions are often also referred to as *intercalation*, or simply cell rearrangement. We are not aware of any fundamentally new mesh topology change mechanisms developed in cell biology, and the prior vertex model approaches from materials science have continued to be widely used, such as for studying cell migration behaviors (Bi et al., 2015). However, other codes have occasionally been applied as well. For example, Maître et al. (2016) adapted the Los Topos code from computer graphics (Da et al., 2014) to study embryo development, and Thomas and Hopyan (2023) (among others) have adapted the Surface Evolver code (Brakke, 1992) to other biological settings involving topology change.

The specific case of cell division has often been handled using a cut-plane approach that instantly slices apart all the interface edges (2D) or triangles (3D) it intersects and creates new separating interfaces, yielding a pair of daughter cells (Madhikar et al., 2018; Runser et al., 2024; Vetter et al., 2024). However, this operation is more akin to general cutting operations (e.g., (Sifakis et al., 2007)), which we have excluded from consideration in this survey, as they do not arise "naturally" in smooth interface evolution (see discussion in §2.6).

The ongoing evolution and role of vertex models (i.e., explicit interface tracking) for cellular tissue dynamics was recently reviewed by Briñas-Pascual et al. (2024). Among their observations is that multiphase interface tracking-like non-manifold vertex models are inadequate for some cellular simulation scenarios, since the borders of biological cells do not truly consist of a single connected interface; a more faithful *Deformable Cell Model* discretizes each cell as a disjoint deformable closed manifold polygon or polyhedron, allowing for more flexible and accurate behavior, such as accounting for space *between* cells (Jamali et al., 2010; Runser et al., 2024). (This design is similar to that of Yang et al. (2018), who also eschewed non-manifold meshes in favor of double-layered manifold meshes.) Nevertheless, similar merging and splitting operations (as in two-phase interface tracking) can be useful, as supported in the recent 2D PolyHoop method (Vetter et al., 2024).

7. Selected Related Topics

7.1. Mesh Repair

Surface mesh repair is a topic of study within geometry processing and computer graphics whose aim is to correct defects in polygonal models. Such defects have many sources, from human error in meshes created manually by artists to mesh reconstruction artifacts caused by noise in 3D scanning data. Many of the approaches to topology change we have discussed possess fundamental connections to mesh repair techniques. Consider a small selection of examples. Ju (2004) and Bischoff and Kobbelt (2005) used implicit, grid-based techniques to seal gaps and related errors in triangle meshes; these ideas are echoed in several of the implicit schemes for handling topology change we discussed in Section 3. Guéziec et al. (2001) introduced topological cutting and stitching techniques to convert meshes with certain non-manifold features into strictly manifold meshes, which exhibit similarities to some of Surface Evolver's popping operations Brakke (1992) and El Topo's non-manifold vertex separation (Brochu and Bridson, 2009). Both the computation of winding numbers and the solution of Poisson problems, used in interface tracking to resolve topological errors (Helmsen, 1994; Zaharescu et al., 2010; Unverdi and Tryggvason, 1992; Shin and Juric, 2002), have been used to define or correct inside/outside segmentations in damaged or partially open meshes (Jacobson et al., 2013; Kazhdan et al., 2006; Centin et al., 2015). Many other such connections can certainly be made. However, in explicit interface tracking, the relevant types of mesh flaws are generally limited to proximities and/or (self-)intersections among closed, watertight, and possibly non-manifold triangle meshes. Mesh repair methods, by contrast, consider a much more general class of domain-dependent mesh errors, including holes, gaps or intersections between adjacent polygons, geometric or topological "noise", incorrectly oriented polygons, polygon degeneracies such as zero-area triangles, arbitrary non-manifold features, entirely unconnected set of polygons ("polygon soup"), and more (Attene et al., 2013). Mesh repair thus constitutes a broad and deep topic of study in its own right, beyond the scope of this review. Nevertheless, it could be fruitful to explore whether explicit interface tracking methods can benefit from further ideas in mesh repair.

7.2. Mesh Booleans

Another important point of connection between interface tracking and the study of discrete geometry/topology lies in Boolean operations (or Constructive Solid Geometry), which have a long history in computer-aided design, solid modeling, and computational geometry. Classically, these operations can be realized through Nef polyhedra (Bieri and Nef, 1988), a representation which partitions space into closed regions using linear half-spaces (e.g., as implemented in CGAL (The CGAL Project, 2025; Hachenberger et al., 2007)). However, robust numerical realizations of these concepts are notoriously computationally expensive, as they require arbitrary precision rational arithmetic to resolve (near-)degeneracies in difficult cases. One thrust of modern research in geometry processing has therefore sought to resolve Boolean operations on closed triangle meshes while reducing the computational cost and preserving robustness. This goal is often pursued, at least in part, by leveraging floating-point arithmetic to varying degrees. For example, arithmetic filtering techniques are often used to guarantee correct answers to certain geometric predicates, by using adaptive-precision arithmetic only where necessary and otherwise using standard floating point (Shewchuk, 1997); these ideas have been put to good use in Boolean algorithms (Zhou et al., 2016; Cherchi et al., 2020; Guo and Fu,

2024). A few methods have been proposed that work *entirely* in floating-point arithmetic (Smith and Dodgson, 2007; Lalish, 2022), while others use both exact predicates and exact constructions (Lévy, 2024) for maximal robustness. However, a key challenge with exact methods is that the output may no longer be exactly representable in floating-point, and rounding the vertex coordinates back to floating-point can potentially introduce new intersections. This problem is known as snap rounding. It is itself a difficult problem of ongoing interest in computational geometry (Devillers et al., 2020; Valque, 2024), though the recent work of Valque and Lazard (2025) appears promising. Regardless, essentially all mesh Boolean methods rely on mesh co-refinement followed by inside/outside classification of the enclosed volumes, much like the global explicit methods discussed in §4. Therefore, incorporating recent developments in this domain could likely benefit global explicit topology change methods.

Zhang and colleagues in computational mathematics have also highlighted the close connections between Boolean algebra and interface tracking in their MARS framework (Zhang and Fogelson, 2016), for which they developed 2D and 3D models for deforming continua (Zhang and Li, 2020; Liang et al., 2025). Notably, their mathematical theory specifically allows for the singular points and curves (needed to describe the instant of topology change), analogous to the discrete non-manifold triangle mesh configurations allowed (even in the two-phase case) by e.g., El Topo / Los Topos (Brochu and Bridson, 2009; Da et al., 2014) .

8. Discussion

We have primarily categorized topology change mechanisms based on whether they perform implicit conversion and reconstruction ("implicit"), global mesh co-refinement and in-place topology analysis and correction ("global explicit"), or local incremental mesh surgery ("local explicit"). However, various other design choices can be considered when selecting or implementing a method, which may be influenced by the particular problem domain or application. For example, consider the size of the stencil of triangles impacted by a topology operation. Large modifications can yield smoother geometry after the change, such as inserting a well-resolved connecting tube, or cutting and then capping a slender thread with well-resolved hemispheres (Quan and Schmidt, 2007; Quan et al., 2009). By contrast, modifications with small stencils, such as snapping together or pulling apart individual vertices (Brochu and Bridson, 2009; Da et al., 2014) can tend to create sharper or coarser geometries. In a scenario with strong curvature-dependence (e.g., surface tension forces) installing smoother new geometry can cause less severe disruptions; on the other hand, smaller changes that only mildly perturb the geometry are more likely to succeed (i.e., without inadvertently introducing new intersections or topology errors) in the presence of complex flows.

Another question is how resilient the underlying application is to small interface intersections, non-manifold geometry, or poor quality triangles. Some algorithms seek to guarantee no intersections whatsoever, while others may tolerate occasional, small intersections, if the underlying physics can be made robust to such minor inconsistencies. Some algorithms make temporary use of non-manifold geometry to enable smaller incremental edits during topology change (e.g., (Brakke, 1992; Brochu and Bridson, 2009; Da et al., 2014)), while many others prefer to transition instantly from manifold state to manifold state (e.g., (Lachaud and Montanvert, 1996; Bredno et al., 2003) among many others), possibly because either the mesh data structure or the application domain

require it. Methods that rely on marching-cubes-like reconstructions can often yield triangles that are tiny or sliver-shaped; if the interface triangles are to be used directly in a conforming finite element or boundary element calculation, significant and immediate remeshing may be necessary, whereas if the triangles are only used to distinguish interior from exterior for an underlying grid-based finite difference-type solver, somewhat lower quality triangles may suffice.

We have generally assumed that topology change occurs naturally due to surfaces approaching each other, independent of the particular driving flow. However, it is still important to consider whether the underlying flow will create a true intersection of the mesh, or only cause interfaces to come in close proximity. If the entire interface is driven by a single spatially continuous flow field, actual intersections of the interface typically only happen because of numerical error, e.g., due to spatial or temporal discretization. As a result, global explicit methods (§4), which fundamentally require intersections to occur before processing topology changes, may be ill-suited, as they can leave persistent, thin, unresolved gaps in places where merging might have been expected. On the other hand, in a geometric flow such as offsetting (outward normal flow) or curvature flow the motion of a vertex is dictated by local shape rather than its position in a surrounding flow field, so interfaces will naturally cross through one another without issue. In this case, the precision offered by global explicit schemes may make them preferable to implicit or local explicit ones.

Interface tracking and topology change methods are also subject to trade-offs among implementation simplicity, efficiency, and accuracy. The implicit GB FronTier method in its original, basic form (Glimm et al., 1999) (i.e., advect the mesh, sample onto a grid by raycasting, globally reconstruct with marching cubes) is relatively straightforward to both implement and optimize, but it rapidly loses volume and fine-scale details. Increasingly advanced variants of this family can better preserve shape (e.g., Müller (2009); Wojtan et al. (2009, 2010)) and/or volume (e.g., (Aulisa et al., 2003; Shin et al., 2011)) but tend to incur additional geometric complexity and computational expense. Both global and local explicit mesh surgery approaches (§4, §5) suffer even less from implicit reconstruction-induced smoothing, but are generally considered more challenging to implement and make robust (see also our discussion on Boolean operations in §7.2). Thus, it may in some cases be more effective (or at least simpler) to employ a basic but highly optimized (or parallelism-friendly) implementation at a higher interface resolution than to upgrade or replace the core interface tracking method for higher fidelity. Unfortunately, explicit interface tracking and topology change schemes have only seldom been directly compared against one another, having more often been pitted against alternative level set, particle, or VOF approaches (an exception is the work of Heiss-Synak et al. (2024) which carefully compared against Los Topos (Da et al., 2014) for the multimaterial case). A systematic head-to-head study comparing the various available explicit interface tracking schemes would be useful to better characterize the Pareto frontier of design trade-offs within this class of methods.

9. Conclusion and Future Directions

In this work, we have sought to review and summarize the wide range of topology change methods proposed for explicit interface tracking in the relevant literatures. A timeline of the development of 3D topology change methods is given in Figure 61, and a list of publicly available 3D explicit interface tracking codes is given in Table 1. While

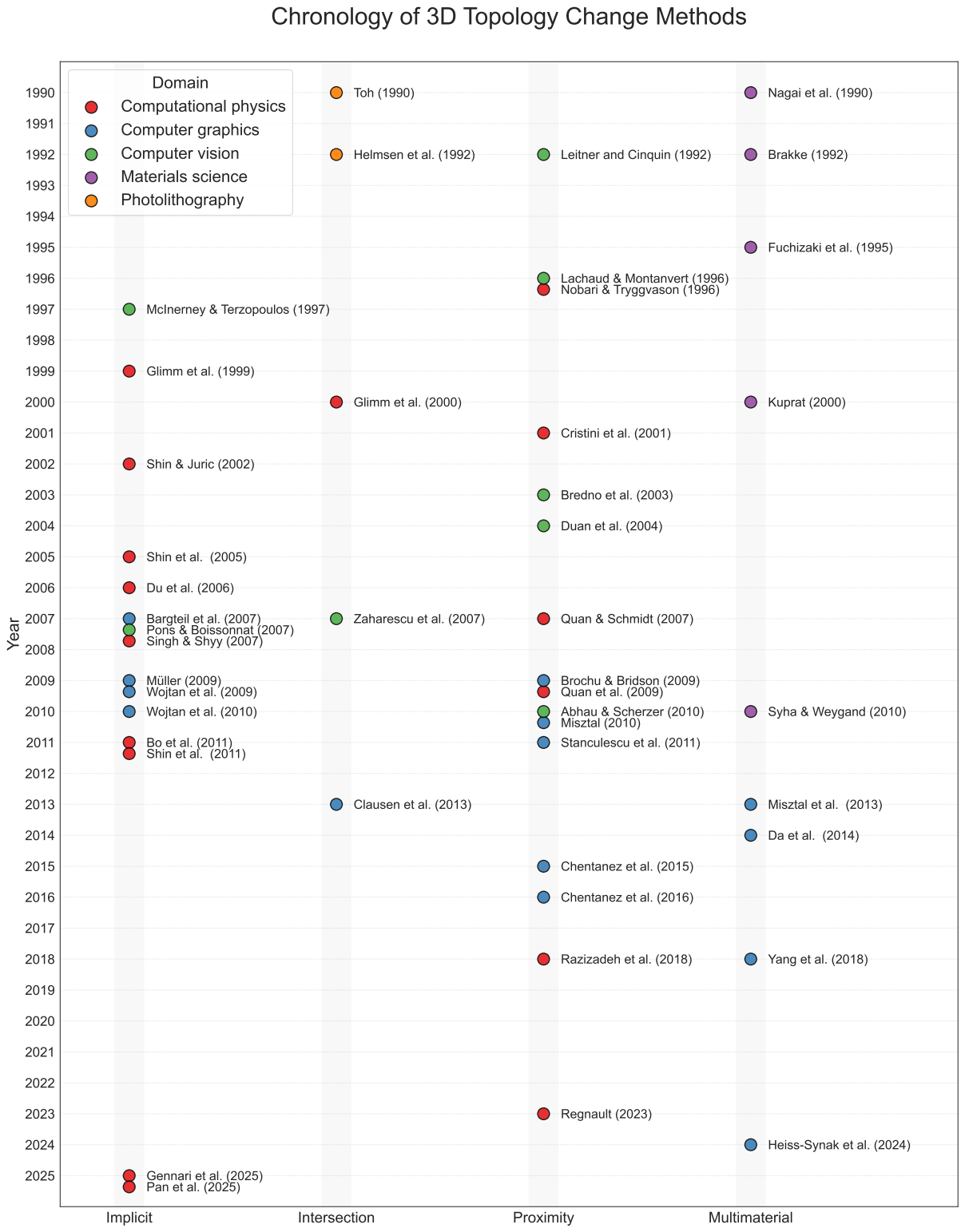

Figure 61: A categorized timeline of 3D topology-adaptive interface tracking methods. To reduce clutter, we include only the first paper in sets of publications by the same author(s) if they describe essentially the same method.

Software Name	Reference Paper	URL
FronTier++	(Bo et al., 2011)	https://github.com/b2tine/FronTierCpp
El Topo	(Brochu and Bridson, 2009)	https://github.com/tysonbrochu/eltopo
Los Topos	(Da et al., 2014)	https://github.com/christopherbatty/MultiTracker
Deformable Simplicial Complex	(Misztal and Bærentzen, 2012)	https://github.com/janba/DSC
SurfaceEvolver	(Brakke, 1992)	https://kenbrakke.com/evolver/evolver.html
SuperDuperTopoFixer	(Heiss-Synak et al., 2024)	https://git.ista.ac.at/psynak/superdupertopofixer

Table 1: Publicly available software for topology-adaptive explicit interface tracking.

this already represents a large body of research, there remains much work to be done to unlock the full potential of topology-adaptive explicit interface tracking, both for topology change and other aspects. We therefore conclude by briefly outlining various possible directions for future investigation, in no particular order.

- Many implicit topology change schemes rely on variations of marching tetrahedra or cubes for isosurface reconstruction. It may be valuable to explore incorporating some of the many other existing isosurfacing variations (De Araújo et al., 2015), such as dual contouring (Ju et al., 2002; Carrera et al., 2026) or schemes with verified accuracy (Etiene et al., 2009, 2011).

- Polygonal Area Mapping schemes (Zhang and Liu, 2008; Zhang and Fogelson, 2014, 2016) are among the only explicit interface tracking schemes that offer exact *local* volume preservation across time steps, but are so far limited to two dimensions, two materials, and no topology change. Can they be extended to 3D with support for topology change or multiple materials?

- The vast majority of explicit interface tracking methods assume piecewise flat interfaces, i.e., using triangle meshes. However, the two-dimensional cubic iPAM and cubic MARS methods offer high order accuracy using smooth cubic splinegon representations (Zhang and Fogelson, 2016; Zhang, 2018). Similarly, Elgeti et al. (2012) and Zwicke et al. (2017) explored NURBS (Non-Uniform Rational B-Splines) for 3D free-surface flow, similar to isogeometric finite element analysis. However, none of these papers considered topological change. Recent work on collision detection and intersection computation for higher-order surface representations could prove useful (Zhang et al., 2023; Yang et al., 2023).

- The methods of Brochu and Bridson (2009) and Da et al. (2014) rely on collision-handling techniques adapted from cloth simulation (Bridson et al., 2002; Harmon et al., 2008) to avoid interpenetrations, which date back to the early 2000s. There have since been significant advances in robust collision-handling that could be brought to bear, such as improved methods for continuous collision detection (Wang et al., 2021) or the incremental potential contact approach for collision response (Li et al., 2020; Chen et al., 2025a; Zheng et al., 2025).

- With a few key exceptions (Bernstein and Wojtan, 2013; Brakke, 1992), the algorithms discussed here assume that explicit interfaces always consist of closed watertight regions. The topic of tracking open interfaces is a promising direction for further study, e.g., to simulate topology change for dynamic soap films stretched over wires or the blowing of soap bubbles, where the notion of "inside" is undefined.

(Surface Evolver naturally handles open surfaces, but does not support collision-induced merging.)

- Unsplit EBIT (Pan et al., 2026) and the method of Bernstein and Wojtan (2013) proposed to use a notion of collision-parity to determine inside/outside labeling in an incremental fashion, which raises many exciting questions. As discussed in §2.7, using the parity (even-odd) winding number rule in the static case is inconsistent with the desired behavior of the positive winding number rule. It is unclear whether or not these collision-parity approaches always produce the desired result for arbitrary deformations; if not, can they be modified to do so? Collision-based approaches also introduce a history dependence that is absent in static winding number-based approaches. These and other tradeoffs (e.g., in terms of complexity, efficiency, robustness, etc.) remain to be evaluated.

- As we have discussed (§8), both proximity and intersection can be appropriate criteria for initiating topology change depending on the scenario or governing flow. It may therefore be useful to develop a unified mesh surgery-based scheme that allows for *both* global explicit (§4) and local explicit (§5) strategies (for example, by combining the method of Zaharescu et al. (2010) with that of Brochu and Bridson (2009)). Implicit methods (§3) can already easily allow both proximity or intersection as triggers, simply by choosing when/where to apply mesh reconstruction.

- Zhang and Fogelson (2016) suggested that it should be possible to achieve higher-order accuracy (with respect to simulation grid resolution) across topological changes, but to our knowledge, this has yet to be clearly demonstrated.

- Modern advances in robustness and efficiency of mesh Booleans/co-refinement could be leveraged to improve global explicit topology change methods (see discussion in §7.2).

- Besides topological change treatments, explicit interface tracking also relies heavily on remeshing to maintain triangle quality during deformation. In recent work in computational physics, Regnault (2023) studied several aspects of mesh adaptation for two-phase flow; Hu et al. (2025) showed how to add and remove vertices in a 2D spline-based approach while retaining high order accuracy; and the recent 3D linear MARS method proposed by Qiu and Zhang (2025) describes local triangle remeshing techniques that retain second and third order accuracy in terms of the Lagrangian and Eulerian length scales, respectively. Can recent developments in remeshing techniques for static geometry (Khan et al., 2020), such as strictly error-bounded remeshing (Cheng et al., 2019) or faster intersection-free remeshing (Liu et al., 2025) offer improvements for the interface tracking context?

- Recent work in elastodynamics considers tightly integrating the task of volumetric tetrahedral remeshing into the time-evolution of the physics (Ferguson et al., 2023; Wen et al., 2025) to achieve high accuracy with few elements. For the same reason, it may likewise be useful to consider integrating interface remeshing and topology change into the interface advection process, rather than treating each in isolation.

- Some authors have considered specifying vertex trajectories in other ways (beyond simply querying a vector field) to reduce mesh deterioration or improve accuracy.

Gorges et al. (2023) discarded tangential velocity components to reduce clustering of vertices during propagation. The face offsetting method of Jiao (2007) supports either wavefrontal or advective interface motion, by offsetting the faces of the mesh and reconstructing the implied vertex positions. This topic remains somewhat underexplored.

- Some of the topology change schemes we discussed have been parallelized or ported to graphics processing units (GPUs) (Chentanez et al., 2016), as have various related or component techniques (e.g., Poisson surface reconstruction (Kazhdan and Hoppe, 2023) and marching cubes (Yang et al., 2024)). Recent frameworks and data structure efforts (Jiang et al., 2022; Mahmoud et al., 2025) also show promise in making dynamic triangle mesh algorithms (of which explicit interface tracking is an important instance) much easier to adapt to parallel or GPU environments, although these authors did not consider topological modifications. High performance shell and cloth simulation, which share similarities with interface tracking, have also been heavily investigated on both CPU and GPU (Lu et al., 2025; Lan et al., 2024). There is room for further study of this topic.

- As discussed in §6, extensions from two phases to multiple phases have been considered for several families of methods, including implicit methods (Heiss-Synak et al., 2024), local explicit triangle-based methods (Da et al., 2014), and local explicit tetrahedra-based methods (Misztal, 2010). It remains to adapt other approaches to the multiphase case, such as the local explicit hole-pairing schemes (Bredno et al., 2003; Chentanez et al., 2015) or global explicit schemes (Zaharescu et al., 2010).

- Bischoff et al. (2005) proposed an active contour method for the evolution of curves lying *on* surfaces, which supports (implicit-style) topology change; the method extends their earlier 2D approaches (Bischoff and Kobbelt, 2004a,b), and thus also has some similarities to EBIT methods. However, there appears to have been little further exploration of this task with explicit interface tracking schemes, despite interest in both general PDEs on surfaces (Dziuk and Elliott, 2013) and alternative level set-based methods for on-surface interface evolution (Chen et al., 2025b; Macdonald and Ruuth, 2008).

- We are not aware of any study that has comprehensively compared or benchmarked the many families of explicit interface tracking methods (or the associated topology change mechanisms). Establishing a set of large-scale standardized benchmarks for the evaluation of explicit interface tracking methods could serve to propel the field forward.

- In terms of new application domains, there has been significant recent interest in visual computing communities around gradient-based mesh optimization, particularly for applications in deep learning, differentiable/inverse rendering, and generative modeling (Liu et al., 2018; Nicolet et al., 2021; Barda et al., 2024, 2023). The approach of Palfinger (2022) essentially uses explicit interface tracking *without* topology changes to iteratively fit a 3D triangle mesh to a shape given a set of 2D images; other methods resort to marching tetrahedra/cubes-like strategies to allow topology change (Shen et al., 2021, 2023; Binninger et al., 2025). Explicit interface

tracking *with* topology change support may be a viable alternative; indeed, the Los Topos code (Da et al., 2014) has recently been used for inverse microscopy to recover the shapes of multicellular biological structures (Ichbiah et al., 2025).

- Similarly, explicit interface tracking has not yet been deployed across as diverse a set of domains as the level set method; it would be interesting to evaluate the former's effectiveness in many of those applications as well. For example, the level set review by Gibou et al. (2018) considers problems in solidification, directed self-assembly, and Parkinson's disease, and Garzon et al. (2022) discuss industrial applications in inkjet printing, electrojetting, and automotive painting.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the author used ChatGPT to aid in revising the text of (only) Section 1, followed by additional manual editing. It was also used to support literature search, in combination with traditional websearch tools and academic databases. Afterwards, the author reviewed and edited the content as needed and takes full responsibility for the content of the article.

Acknowledgments

The author is thankful to Andreas Bærentzen, Qinghai Zhang, Yunhao Qiu, David Cha, and Liang Cui for reading draft versions of the paper and providing helpful feedback. I also gratefully acknowledge research funding provided by the Natural Sciences and Engineering Research Council of Canada (Grants RGPIN-2021-02524, RGPIN-2026-04447).

References

Abhau, J., Scherzer, O., 2010. A combinatorial method for topology adaptations in 3D deformable models. International journal of computer vision 87, 304–315. doi:10.1007/s11263-009-0282-5.

Aboulhasanzadeh, B., Hosoda, S., Tomiyama, A., Tryggvason, G., 2013. A validation of an embedded analytical description approach for the computations of high schmidt number mass transfer from bubbles in liquids. Chemical Engineering Science 101, 165–174. doi:10.1016/j.ces.2013.06.020.

Aboulhasanzadeh, B., Thomas, S., Taeibi-Rahni, M., Tryggvason, G., 2012. Multiscale computations of mass transfer from buoyant bubbles. Chemical Engineering Science 75, 456–467. doi:10.1016/j.ces.2012.04.005.

Alliez, P., Cohen-Steiner, D., Yvinec, M., Desbrun, M., 2005. Variational tetrahedral meshing, in: ACM SIGGRAPH 2005 Papers, pp. 617–625. doi:10.1145/1073204.1073238.

Alliez, P., Ucelli, G., Gotsman, C., Attene, M., 2008. Recent advances in remeshing of surfaces, in: De Floriani, L., Spagnuolo, M. (Eds.), Shape analysis and structuring, Springer Berlin Heidelberg. pp. 53–82. doi:10.1007/978-3-540-33265-7_2.

Angelidis, A., Singh, K., 2006. Space deformations and their application to shape modeling, in: ACM SIGGRAPH 2006 Courses, pp. 10–29. doi:10.1145/1185657.1185672.

Araki, S., Yokoya, N., Iwasa, H., Takemura, H., 1997. Splitting of active contour models based on crossing detection for extraction of multiple objects. Systems and computers in Japan 28, 34–42. doi:10.1002/(SICI)1520-684X(199710)28:11<34::AID-SCJ4>3.0.CO;2-L.

Araki, S., Yokoya, N., Takemura, H., 1999. Real-time tracking of multiple moving objects using split-and-merge contour models based on crossing detection. Systems and Computers in Japan 30, 25–33. doi:10.1002/(SICI)1520-684X(199908)30:9<25::AID-SCJ4>3.0.CO;2-G.

Ashgriz, N., Barbat, T., Wang, G., 2004. A computational Lagrangian–Eulerian advection remap for free surface flows. International Journal for Numerical Methods in Fluids 44, 1–32. doi:10.1002/fld.620.

Attene, M., Campen, M., Kobbelt, L., 2013. Polygon mesh repairing: An application perspective. ACM Computing Surveys (CSUR) 45, 1–33. doi:10.1145/2431211.2431214.

Aulisa, E., Barbi, G., Cervone, A., Chierici, A., Giangolini, F., Manservisi, S., Sirotti, L., et al., 2024. Divergence free velocity interpolation for surface marker tracking, in: WCCM-ECCOMAS CONGRESS, Scipedia SL. pp. 1–12.

Aulisa, E., Manservisi, S., Scardovelli, R., 2003. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. Journal of Computational Physics 188, 611–639. doi:10.1016/S0021-9991(03)00196-7.

Aulisa, E., Manservisi, S., Scardovelli, R., 2004. A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking. Journal of Computational Physics 197, 555–584. doi:10.1016/j.jcp.2003.12.009.

Bærentzen, A., 2025. On the topology of polygonal meshes. arXiv preprint arXiv:2511.11618 .

Balazovjech, M., Mikula, K., Petrášová, M., Urbán, J., 2012. Lagrangean method with topological changes for numerical modelling of forest fire propagation, in: Proceedings of ALGORITMY, pp. 42–52.

Barda, A., Erel, Y., Kasten, Y., Bermano, A.H., 2023. Roar: robust adaptive reconstruction of shapes using planar projections. arXiv preprint arXiv:2307.00690 .

Barda, A., Kim, V., Aigerman, N., Bermano, A.H., Groueix, T., 2024. MagicClay: Sculpting meshes with generative neural fields, in: SIGGRAPH Asia 2024 Conference Papers, pp. 1–10. doi:10.1145/3680528.3687627.

Barequet, G., Sharir, M., 1995. Filling gaps in the boundary of a polyhedron. Computer Aided Geometric Design 12, 207–229. doi:10.1016/0167-8396(94)00011-G.

Bargteil, A.W., Goktekin, T.G., O'Brien, J.F., Strain, J.A., 2006. A semi-Lagrangian contouring method for fluid simulation. ACM Transactions on Graphics (TOG) 25, 19–38. doi:10.1145/1122501.1122503.

Bargteil, A.W., Wojtan, C., Hodgins, J.K., Turk, G., 2007. A finite element method for animating large viscoplastic flow. ACM transactions on graphics (TOG) 26, 16–es. doi:10.1145/1276377.1276397.

Benninghoff, H., Garcke, H., 2014. Efficient image segmentation and restoration using parametric curve evolution with junctions and topology changes. SIAM Journal on Imaging Sciences 7, 1451–1483. doi:10.1137/130932430.

Bernstein, G.L., Wojtan, C., 2013. Putting holes in holey geometry: Topology change for arbitrary surfaces. ACM Transactions on Graphics (TOG) 32, 1–12. doi:10.1145/2461912.2462027.

Bi, D., Lopez, J., Schwarz, J.M., Manning, M.L., 2015. A density-independent rigidity transition in biological tissues. Nature Physics 11, 1074–1079. doi:10.1038/nphys3471.

Bieri, H., Nef, W., 1988. Elementary set operations with d-dimensional polyhedra, in: Workshop on Computational Geometry, Springer. pp. 97–112. doi:10.1007/3-540-50335-8_28.

Binninger, A., Wiersma, R., Herholz, P., Sorkine-Hornung, O., 2025. TetWeave: Isosurface extraction using on-the-fly delaunay tetrahedral grids for gradient-based mesh optimization. ACM Transactions on Graphics (TOG) 44, 1–19. doi:10.1145/3730851.

Bischoff, S., Kobbelt, L., 2004a. Snakes with topology control, in: 4th Israel-Korean Bi-National Conference on Geometric Modeling and Computer Graphics.

Bischoff, S., Kobbelt, L., 2005. Structure preserving CAD model repair, in: Computer Graphics Forum, Amsterdam: North Holland, 1982-. pp. 527–536. doi:10.1111/j.1467-8659.2005.00878.x.

Bischoff, S., Kobbelt, L.P., 2004b. Parameterization-free active contour models with topology control. The Visual Computer 20, 217–228. doi:10.1007/s00371-003-0228-9.

Bischoff, S., Weyand, T., Kobbelt, L., 2005. Snakes on triangle meshes, in: Bildverarbeitung für die Medizin 2005: Algorithmen—Systeme—Anwendungen Proceedings des Workshops vom 13.–15. März 2005 in Heidelberg, Springer. pp. 208–212. doi:10.1007/3-540-26431-0_43.

Bo, W., Liu, X., Glimm, J., Li, X., 2011. A robust front tracking method: verification and application to simulation of the primary breakup of a liquid jet. SIAM Journal on Scientific Computing 33, 1505–1524. doi:10.1137/10079135X.

Boettinger, W.J., Warren, J.A., Beckermann, C., Karma, A., 2002. Phase-field simulation of solidification. Annual review of materials research 32, 163–194. doi:10.1146/annurev.matsci.32.101901.155803.

Botsch, M., Kobbelt, L., 2004. A remeshing approach to multiresolution modeling, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 185–192. doi:10.1145/1057432.1057457.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon mesh processing. CRC press. doi:10.1201/b10688.

Brakke, K., 2004. Surface evolver workshop. https://kenbrakke.com/evolver/workshop/ [Accessed: 2026-02-04].

Brakke, K.A., 1992. The surface evolver. Experimental mathematics 1, 141–165. doi:10.1080/10586458.1992.10504253.

Bredno, J., Lehmann, T.M., Spitzer, K., 2003. A general discrete contour model in two, three, and four dimensions for topology-adaptive multichannel segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 25, 550–563. doi:10.1109/TPAMI.2003.1195990.

Bridson, R., Fedkiw, R., Anderson, J., 2002. Robust treatment of collisions, contact and friction for cloth animation, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 594–603. doi:10.1145/566570.566623.

Briñas-Pascual, N., Cornwall-Scoones, J., O'Hanlon, D.P., Guerrero, P., Perez-Carrasco, R., 2024. No country for old frameworks? vertex models and their ongoing reinvention to study tissue dynamics. Biophysica 4, 586–603. doi:10.3390/biophysica4040039.

Brochu, T., Batty, C., Bridson, R., 2010. Matching fluid simulation elements to surface geometry and topology. ACM Transactions on Graphics (TOG) 29. doi:10.1145/1778765.1778784.

Brochu, T., Bridson, R., 2009. Robust topological operations for dynamic explicit surfaces. SIAM Journal on Scientific Computing 31, 2472–2493. doi:10.1137/08073761.

Brochu, T., Keeler, T., Bridson, R., 2012. Linear-time smoke animation with vortex sheet meshes, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 87–95. doi:10.5555/2422356.2422371.

Brochu, T., Schmidt, R.M., 2017. Geometric modeling of multi-material printed objects, in: Eurographics (Short Papers), pp. 45–48. doi:10.2312/egsh.20171011.

Bronsard, L., Wetton, B.T., 1995. A numerical method for tracking curve networks moving with curvature motion. Journal of Computational Physics 120, 66–87. doi:10.1006/jcph.1995.1149.

Bronson, J., Levine, J.A., Whitaker, R., 2013. Lattice cleaving: A multimaterial tetrahedral meshing algorithm with guarantees. IEEE transactions on visualization and computer graphics 20, 223–237. doi:10.1109/TVCG.2013.115.

Campen, M., Kobbelt, L., 2010. Exact and robust (self-)intersections for polygonal meshes, in: Computer Graphics Forum, Wiley Online Library. pp. 397–406. doi:10.1111/j.1467-8659.2009.01609.x.

Carrera, X., Wang, N., Batty, C., Stein, O., Sellán, S., 2026. Dual contouring of signed distance data. arXiv preprint arXiv:2604.00157 .

Caselles, V., Catté, F., Coll, T., Dibos, F., 1993. A geometric model for active contours in image processing. Numerische mathematik 66, 1–31. doi:10.1007/BF01385685.

Ceniceros, H.D., Roma, A.M., 2005. A multi-phase flow method with a fast, geometry-based fluid indicator. Journal of computational Physics 205, 391–400. doi:10.1016/j.jcp.2004.11.013.

Centin, M., Pezzotti, N., Signoroni, A., 2015. Poisson-driven seamless completion of triangular meshes. Computer Aided Geometric Design 35, 42–55. doi:10.1016/j.cagd.2015.03.006.

Chen, A.H., Hsu, J., Liu, Z., Macklin, M., Yang, Y., Yuksel, C., 2025a. Offset geometric contact. ACM Transactions on Graphics (TOG) 44, 1–21. doi:10.1145/3731205.

Chen, J., Xiao, X., Feng, X., Sheen, D., 2025b. A level set immersed finite element method for parabolic problems on surfaces with moving interfaces. Journal of Computational Physics 531, 113939. doi:10.1016/j.jcp.2025.113939.

Cheng, X.X., Fu, X.M., Zhang, C., Chai, S., 2019. Practical error-bounded remeshing by adaptive refinement. Computers & Graphics 82, 163–173. doi:10.1016/j.cag.2019.05.019.

Chentanez, N., Müller, M., Macklin, M., 2016. GPU accelerated grid-free surface tracking. Computers & Graphics 57, 1–11. doi:10.1016/j.cag.2016.03.002.

Chentanez, N., Müller, M., Macklin, M., Kim, T.Y., 2015. Fast grid-free surface tracking. ACM Transactions on Graphics (TOG) 34, 1–11. doi:10.1145/2766991.

Cherchi, G., Livesu, M., Scateni, R., Attene, M., 2020. Fast and robust mesh arrangements using floating-point arithmetic. ACM Transactions on Graphics (TOG) 39, 1–16. doi:10.1145/3414685.3417818.

Chiaruttini, V., Riolo, V., Feyel, F., 2013. Advanced remeshing techniques for complex 3D crack propagation, in: 13th International conference on fracture, Beijing, China, pp. 547–55.

Chirco, L., Maarek, J., Popinet, S., Zaleski, S., 2022. Manifold death: a volume of fluid implementation of controlled topological changes in thin sheets by the signature method. Journal of Computational Physics 467, 111468. doi:10.1016/j.jcp.2022.111468.

Chirco, L., Zaleski, S., 2023. An edge-based interface-tracking method for multiphase flows. International Journal for Numerical Methods in Fluids 95, 491–497. doi:10.1002/fld.5144.

Choi, W.P., Lam, K.M., Siu, W.C., 2001. An adaptive active contour model for highly irregular boundaries. Pattern Recognition 34, 323–331. doi:10.1016/S0031-3203(99)00231-9.

Christiansen, A.N., Nobel-Jørgensen, M., Aage, N., Sigmund, O., Bærentzen, J.A., 2014. Topology optimization using an explicit interface representation. Structural and Multidisciplinary Optimization 49, 387–399. doi:10.1007/s00158-013-0983-9.

Clausen, P., Wicke, M., Shewchuk, J.R., O'brien, J.F., 2013. Simulating liquids and solid-liquid interactions with lagrangian meshes. ACM Transactions on Graphics (TOG) 32, 1–15. doi:10.1145/2451236.2451243.

Cohen, I., Cohen, L., 1993. Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images . IEEE Transactions on Pattern Analysis And Machine Intelligence 15, 1131–1147. doi:10.1109/34.244675.

Cohen, L.D., 1991. On active contour models and balloons. CVGIP: Image understanding 53, 211–218. doi:10.1016/1049-9660(91)90028-N.

Cristini, V., Bławzdziewicz, J., Loewenberg, M., 2001. An adaptive mesh algorithm for evolving surfaces: simulations of drop breakup and coalescence. Journal of Computational Physics 168, 445–463. doi:10.1006/jcph.2001.6713.

Da, F., Batty, C., Grinspun, E., 2014. Multimaterial mesh-based surface tracking. ACM Transactions on Graphics (TOG) 33, 112–1. doi:10.1145/2601097.2601146.

Da, F., Batty, C., Wojtan, C., Grinspun, E., 2015. Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams. ACM Transactions on Graphics (TOG) 34, 1–9. doi:10.1145/2767003.

Da, F., Hahn, D., Batty, C., Wojtan, C., Grinspun, E., 2016. Surface-only liquids. ACM Transactions on Graphics (TOG) 35, 1–12. doi:10.1145/2897824.2925899.

De Araújo, B.R., Lopes, D.S., Jepp, P., Jorge, J.A., Wyvill, B., 2015. A survey on implicit surface polygonization. ACM Computing Surveys (CSUR) 47, 1–39. doi:10.1145/2732197.

De Berg, M., Cheong, O., Van Kreveld, M., Overmars, M., 2008. Computational geometry: algorithms and applications. Springer. doi:10.1007/978-3-540-77974-2.

DeBar, R.B., 1974. Fundamentals of the KRAKEN code.[Eulerian hydrodynamics code for compressible nonviscous flow of several fluids in two-dimensional (axially symmetric) region]. Technical Report. California Univ., Livermore (USA). Lawrence Livermore Lab.

DeCarlo, D., Gallier, J., 1996. Topological evolution of surfaces, in: Graphics Interface, pp. 194–203. doi:10.5555/241020.241075.

Delingette, H., Montagnat, J., 2000. New algorithms for controlling active contours shape and topology, in: European conference on computer vision, Springer. pp. 381–395. doi:10.1007/3-540-45053-X_25.

Devillers, O., Lazard, S., Lenhart, W.J., 2020. Rounding meshes in 3D. Discrete & Computational Geometry 64, 37–62. doi:10.1007/s00454-020-00202-2.

Doğan, G., Morin, P., Nochetto, R.H., 2008. A variational shape optimization approach for image segmentation with a Mumford–Shah functional. SIAM Journal on Scientific Computing 30, 3028–3049. doi:10.1137/070692066.

Doi, A., Koide, A., 1991. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. IEICE TRANSACTIONS on Information and Systems 74, 214–224.

Du, J., Fix, B., Glimm, J., Jia, X., Li, X., Li, Y., Wu, L., 2006. A simple package for front tracking. Journal of Computational Physics 213, 613–628. doi:10.1016/j.jcp.2005.08.034.

Duan, Y., Yang, L., Qin, H., Samaras, D., 2004. Shape reconstruction from 3D and 2D data using PDE-based deformable surfaces, in: European conference on computer vision, Springer. pp. 238–251. doi:10.1007/978-3-540-24672-5_19.

Dunyach, M., Vanderhaeghe, D., Barthe, L., Botsch, M., 2013. Adaptive remeshing for real-time mesh deformation, in: 34th Annual Conference of the European Association for Computer Graphics (Eurographics 2013), The Eurographics Association. doi:https://doi.org/10.2312/conf/EG2013/short/029-032.

Ďurikovič, R., Kaneda, K., Yamashita, H., 1995. Dynamic contour: a texture approach and contour operations. The Visual Computer 11, 277–289. doi:10.1007/BF01898405.

Dyadechko, V., Shashkov, M., 2005. Moment-of-fluid interface reconstruction. Los Alamos Report LA-UR-05-7571 49.

Dziuk, G., Elliott, C.M., 2013. Finite element methods for surface pdes. Acta Numerica 22, 289–396. doi:10.1017/S0962492913000056.

Eberly, D., 2021. Robust and error-free geometric computing. CRC Press. doi:10.1201/9780429330506.

Elgeti, S., Sauerland, H., Pauli, L., Behr, M., 2012. On the usage of NURBS as interface representation in free-surface flows. International journal for numerical methods in fluids 69, 73–87. doi:10.1002/fld.2537.

Enomoto, Y., Kawasaki, K., Nagai, T., 1989. Two-dimensional vertex model with local friction coefficient. International Journal of Modern Physics B 3, 163–169. doi:10.1142/S0217979289000142.

Ericson, C., 2004. Real-time collision detection. CRC Press.

Etiene, T., Nonato, L.G., Scheidegger, C., Tienry, J., Peters, T.J., Pascucci, V., Kirby, R.M., Silva, C.T., 2011. Topology verification for isosurface extraction. IEEE Transactions on Visualization and Computer Graphics 18, 952–965. doi:10.1109/TVCG.2011.109.

Etiene, T., Scheidegger, C., Nonato, L.G., Kirby, R.M., Silva, C., 2009. Verifiable visualization for isosurface extraction. IEEE Transactions on Visualization and Computer Graphics 15, 1227–1234. doi:10.1109/TVCG.2009.194.

Ferguson, Z., Schneider, T., Kaufman, D., Panozzo, D., 2023. In-timestep remeshing for contacting elastodynamics. ACM Transactions on Graphics 42. doi:10.1145/3592428.

Fradkov, V., Glicksman, M., Palmer, M., Nordberg, J., Rajan, K., 1993. Topological rearrangements during 2D normal grain growth. Physica D: Nonlinear Phenomena 66, 50–60. doi:10.1016/0167-2789(93)90223-N.

Freitag, L.A., Ollivier-Gooch, C., 1997. Tetrahedral mesh improvement using swapping and smoothing. International Journal for Numerical Methods in Engineering 40, 3979–4002. doi:10.1002/(SICI)1097-0207(19971115)40:21<3979::AID-NME251>3.0.CO;2-9.

Frost, H., Thompson, C., Howe, C., Whang, J., 1988. A two-dimensional computer simulation of capillarity-driven grain growth: Preliminary results. Scripta Metallurgica 22, 65–70. doi:10.1016/S0036-9748(88)80307-7.

Fuchizaki, K., Kusaba, T., Kawasaki, K., 1995. Computer modelling of three-dimensional cellular pattern growth. Philosophical Magazine B 71, 333–357. doi:10.1080/13642819508239038.

Garzon, M., Saye, R.I., Sethian, J.A., 2022. Efficient algorithms for tracking moving interfaces in industrial applications: Inkjet plotters, electrojetting, industrial foams, and rotary bell painting, in: Recent Advances in Industrial and Applied Mathematics, Springer. pp. 173–194. doi:10.1007/978-3-030-86236-7_10.

Gennari, G., Gorges, C., Denner, F., Wachem, B., 2025. A marching cubes based method for topology changes in three-dimensional two-phase flows with front tracking. Journal of Computational Physics 540, 114284. doi:10.1016/j.jcp.2025.114284.

Gibou, F., Fedkiw, R., Osher, S., 2018. A review of level-set methods and some recent applications. Journal of Computational Physics 353, 82–109. doi:10.1016/j.jcp.2017.10.006.

Glimm, J., Grove, J., Lindquist, B., McBryan, O.A., Tryggvason, G., 1988. The bifurcation of tracked scalar waves. SIAM Journal on Scientific and Statistical Computing 9, 61–79. doi:10.1137/0909006.

Glimm, J., Grove, J.W., Li, X., Zhao, N., 1999. Simple front tracking. Contemporary mathematics 238, 133–149.

Glimm, J., Grove, J.W., Li, X.L., Tan, D.C., 2000. Robust computational algorithms for dynamic interface tracking in three dimensions. SIAM Journal on Scientific Computing 21, 2240–2256. doi:10.1137/S1064827598340500.

Gorges, C., Evrard, F., van Wachem, B., Denner, F., 2022. Reducing volume and shape errors in front tracking by divergence-preserving velocity interpolation and parabolic fit vertex positioning. Journal of Computational Physics 457, 111072. doi:10.1016/j.jcp.2022.111072.

Gorges, C., Hodžić, A., Evrard, F., van Wachem, B., Velte, C.M., Denner, F., 2023. Efficient reduction of vertex clustering using front tracking with surface normal propagation restriction. Journal of Computational Physics 491, 112406. doi:10.1016/j.jcp.2023.112406.

Gottstein, G., Shvindlerman, L.S., 2009. Grain boundary migration in metals: thermodynamics, kinetics, applications. CRC press.

Guéziec, A., Taubin, G., Lazarus, F., Hom, B., 2001. Cutting and stitching: Converting sets of polygons to manifold surfaces. IEEE Transactions on Visualization and Computer Graphics 7, 136–151. doi:10.1109/2945.928166.

Guo, J.P., Fu, X.M., 2024. Exact and efficient intersection resolution for mesh arrangements. ACM Transactions on Graphics (TOG) 43, 1–14. doi:10.1145/3687925.

Hachenberger, P., Kettner, L., Mehlhorn, K., 2007. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. Computational Geometry 38, 64–99. doi:10.1016/j.comgeo.2006.11.009.

Harlow, F.H., Welch, J.E., et al., 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Physics of fluids 8, 2182. doi:10.1063/1.1761178.

Harmon, D., Vouga, E., Tamstorf, R., Grinspun, E., 2008. Robust treatment of simultaneous collisions. ACM Transactions on Graphics (TOG) 27, 1–4. doi:10.1145/1360612.1360622.

Heiss-Synak, P., Kalinov, A., Strugaru, M., Etemadi, A., Yang, H., Wojtan, C., 2024. Multi-material mesh-based surface tracking with implicit topology changes. ACM Transactions on Graphics (TOG) 43, 1–14. doi:10.1145/365822.

Helmsen, J.J., 1994. A comparison of three dimensional photolithography simulators. University of California, Berkeley.

Helmsen, J.J., Scheckler, E., Neureuther, A., Sequin, C., 1992. An efficient loop detection and removal algorithm for 3D surface-based lithography simulation, in: NUPAD IV. Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits,, IEEE. pp. 3–8. doi:10.1109/NUPAD.1992.673838.

Hirt, C.W., Nichols, B.D., 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of computational physics 39, 201–225. doi:10.1016/0021-9991(81)90145-5.

Ho, C.C., Wu, F.C., Chen, B.Y., Chuang, Y.Y., Ouhyoung, M., et al., 2005. Cubical marching squares: Adaptive feature preserving surface extraction from volume data, in: Computer graphics forum, Amsterdam: North Holland, 1982-. pp. 537–546. doi:10.1111/j.1467-8659.2005.00879.x.

Ho, M., Yeoh, G., Reizes, J., Timchenko, V., 2016. The intersection marker method for 3D interface tracking of deformable surfaces in finite volumes. International Journal for Numerical Methods in Fluids 81, 220–244. doi:10.1002/fld.4182.

Honda, H., Tanemura, M., Nagai, T., 2004. A three-dimensional vertex dynamics cell model of space-filling polyhedra simulating cell behavior in a cell aggregate. Journal of theoretical biology 226, 439–453. doi:10.1016/j.jtbi.2003.10.001.

Hormann, K., Agathos, A., 2001. The point in polygon problem for arbitrary polygons. Computational geometry 20, 131–144. doi:10.1016/S0925-7721(01)00012-8.

Hu, D., Liang, K., Ying, L., Li, S., Zhang, Q., 2025. ARMS: Adding and removing markers on splines for high-order general interface tracking under the MARS framework. Journal of Computational Physics 521, 113574. doi:10.1016/j.jcp.2024.113574.

Ichbiah, S., Sinha, A., Delbary, F., Turlier, H., 2025. Inverse 3D microscopy rendering for cell shape inference with active mesh, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 26987–26998.

Ishida, S., Synak, P., Narita, F., Hachisuka, T., Wojtan, C., 2020. A model for soap film dynamics with evolving thickness. ACM Transactions on Graphics (TOG) 39, 31–1. doi:10.1145/3386569.3392405.

Jabbour, C., Quintard, M., Bertin, H., Robin, M., 1996. Oil recovery by steam injection: three-phase flow effects. Journal of Petroleum Science and Engineering 16, 109–130. doi:10.1016/0920-4105(96)00013-7.

Jacobson, A., Kavan, L., Sorkine-Hornung, O., 2013. Robust inside-outside segmentation using generalized winding numbers. ACM Transactions on Graphics (TOG) 32, 1–12. doi:10.1145/2461912.2461916.

Jamali, Y., Azimi, M., Mofrad, M.R., 2010. A sub-cellular viscoelastic model for cell population mechanics. PLoS One 5, e12097. doi:10.1371/journal.pone.0012097.

Ji, L., Yan, H., 2002. Robust topology-adaptive snakes for image segmentation. Image and Vision Computing 20, 147–164. doi:10.1109/ICIP.2001.958614.

Jiang, Z., Dai, J., Hu, Y., Zhou, Y., Dumas, J., Zhou, Q., Bajwa, G.S., Zorin, D., Panozzo, D., Schneider, T., 2022. Declarative specification for unstructured mesh editing algorithms. ACM Transactions on Graphics (TOG) 41, 251–1. doi:10.1145/3550454.3555513.

Jiao, X., 2006. Volume and feature preservation in surface mesh optimization, in: Proceedings of the 15th International Meshing Roundtable, Springer. pp. 359–373. doi:10.1007/978-3-540-34958-7_21.

Jiao, X., 2007. Face offsetting: A unified approach for explicit moving interfaces. Journal of computational physics 220, 612–625. doi:10.1016/j.jcp.2006.05.021.

Jiao, X., Zha, H., 2008. Consistent computation of first-and second-order differential quantities for surface meshes, in: Proceedings of the 2008 ACM symposium on Solid and physical modeling, pp. 159–170. doi:10.1145/1364901.1364924.

Ju, T., 2004. Robust repair of polygonal models. ACM Transactions on Graphics (TOG) 23, 888–895. doi:10.1145/1015706.1015815.

Ju, T., Losasso, F., Schaefer, S., Warren, J., 2002. Dual contouring of Hermite data, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 339–346. doi:10.1145/566570.566586.

Jung, W., Shin, H., Choi, B.K., 2004. Self-intersection removal in triangular mesh offsetting. Computer-Aided Design and Applications 1, 477–484. doi:10.1080/16864360.2004.10738290.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. International journal of computer vision 1, 321–331. doi:10.1007/BF00133570.

Kawasaki, K., Nagai, T., Nakashima, K., 1989. Vertex models for two-dimensional grain growth. Philosophical Magazine B 60, 399–421. doi:10.1080/13642818908205916.

Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 61–70. doi:10.2312/SGP/SGP06/061-070.

Kazhdan, M., Hoppe, H., 2023. Distributed Poisson surface reconstruction, in: Computer graphics forum, Wiley Online Library. p. e14925. doi:10.1111/cgf.14925.

Keeler, T., Bridson, R., 2015. Ocean waves animation using boundary integral equations and explicit mesh tracking, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association. p. 11–19. doi:10.2312/sca.20141118.

Kermode, J., Weaire, D., 1990. 2D-FROTH: a program for the investigation of 2-dimensional froths. Computer Physics Communications 60, 75–109. doi:10.1016/0010-4655(90)90080-K.

Khan, D., Plopski, A., Fujimoto, Y., Kanbara, M., Jabeen, G., Zhang, Y.J., Zhang, X., Kato, H., 2020. Surface remeshing: A systematic literature review of methods and research directions. IEEE transactions on visualization and computer graphics 28, 1680–1713. doi:10.1109/TVCG.2020.3016645.

Kinderlehrer, D., Livshits, I., Ta'Asan, S., 2006. A variational approach to modeling and simulation of grain growth. SIAM Journal on Scientific Computing 28, 1694–1715. doi:10.1137/030601971.

Klingner, B.M., Shewchuk, J.R., 2007. Aggressive tetrahedral mesh improvement, in: Proceedings of the 16th international meshing roundtable, Springer. pp. 3–23. doi:10.1007/978-3-540-75103-8_1.

Kuprat, A., 2000a. Inspecting and repairing physical topology in a moving grid grain growth simulation, in: Proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations, Whistler, Canada, International Society of Grid Generation, Mississippi State University, pp. 541–550.

Kuprat, A., 2000b. Modeling microstructure evolution using gradient-weighted moving finite elements. Siam Journal on Scientific Computing 22, 535–560. doi:10.1137/S1064827598348374.

Kuprat, A., George, D., Straub, G., Demirel, M.C., 2003. Modeling microstructure evolution in three dimensions with Grain3D and LaGriT. Computational Materials Science 28, 199–208. doi:10.1016/S0927-0256(03)00107-1.

Kuprat, A., Khamayseh, A., George, D., Larkey, L., 2001. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. Journal of Computational Physics 172, 99–118. doi:10.1006/jcph.2001.6816.

Lachaud, J.O., Montanvert, A., 1996. Volumic segmentation using hierarchical representation and triangulated surface, in: European Conference on Computer Vision, Springer. pp. 137–146. doi:10.1007/BFb0015530.

Lachaud, J.O., Montanvert, A., 1999. Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. Medical Image Analysis 3, 187–207. doi:10.1016/S1361-8415(99)80006-1.

Lachaud, J.O., Taton, B., 2003. Deformable model with adaptive mesh and automated topology changes, in: Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings., IEEE. pp. 12–19. doi:10.1109/IM.2003.1240227.

Laidlaw, D.H., Trumbore, W.B., Hughes, J.F., 1986. Constructive solid geometry for polyhedral objects, in: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 161–170. doi:10.1145/15886.159.

Lalish, E., 2022. Manifold: a library dedicated to creating and operating on manifold triangle meshes. URL: https://github.com/elalish/manifold.

Lan, L., Lu, Z., Long, J., Yuan, C., Li, X., He, X., Wang, H., Jiang, C., Yang, Y., 2024. Efficient gpu cloth simulation with non-distance barriers and subspace reuse. ACM Transactions on Graphics (TOG) 43. doi:10.1145/3687760.

Lazar, E.A., 2011. The evolution of cellular structures via curvature flow. Ph.D. thesis. Princeton.

Lazar, E.A., Mason, J.K., MacPherson, R.D., Srolovitz, D.J., 2011. A more accurate three-dimensional grain growth algorithm. Acta Materialia 59, 6837–6847. doi:10.1016/j.actamat.2011.07.052.

Le Chenadec, V., Pitsch, H., 2013. A 3D unsplit forward/backward volume-of-fluid approach and coupling to the level set method. Journal of computational physics 233, 10–33. doi:10.1016/j.jcp.2012.07.019.

Leitner, F., Cinquin, P., 1992. Complex topology 3-D objects segmentation, in: Model-Based Vision Development and Tools, SPIE. pp. 16–26. doi:10.1117/12.57119.

Lévy, B., 2024. Exact predicates, exact constructions and combinatorics for mesh CSG. ACM Transactions on Graphics doi:10.1145/3744642.

Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., Kaufman, D.M., 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. ACM transactions on graphics doi:10.1145/3386569.3392425.

Li, X., He, X., Liu, X., Liu, B., Wu, E., 2014. Multiphase surface tracking with explicit contouring, in: Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology, pp. 31–40. doi:10.1145/2671015.2671017.

Li, X., He, X., Liu, X., Zhang, J.J., Liu, B., Wu, E., 2016. Multiphase interface tracking with fast semi-Lagrangian contouring. IEEE Transactions on Visualization and Computer Graphics 22, 1973–1986. doi:10.1109/TVCG.2015.2476788.

Liang, H., Qiu, Y., Tan, Y., Zhang, Q., 2025. On topology of three-dimensional continua with singular points. arXiv preprint arXiv:2512.02385 .

Liu, H.T.D., Tao, M., Jacobson, A., 2018. Paparazzi: surface editing by way of multi-view image processing. ACM Transactions on Graphics (TOG) 37, 221. doi:10.1145/3272127.3275047.

Liu, T., Ye, H., Chen, J., 2025. Fast intersection-free remeshing of triangular meshes. IEEE Transactions on Visualization and Computer Graphics doi:10.1109/TVCG.2025.3569926.

Lorensen, W.E., Cline, H.E., 1987. Marching cubes: A high resolution 3D surface construction algorithm, in: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, p. 163–169. doi:10.1145/37401.37422.

Losasso, F., Shinar, T., Selle, A., Fedkiw, R., 2006. Multiple interacting liquids. ACM Transactions on Graphics (TOG) 25, 812–819. doi:10.1145/1141911.1141960.

Lu, Z., Liu, Z., Lan, L., Wang, H., Ishiwaka, Y., Jiang, C., Wu, K., Yang, Y., 2025. High-performance CPU cloth simulation using domain-decomposed projective dynamics. ACM Transactions on Graphics 44. doi:10.1145/3731182.

Macdonald, C.B., Ruuth, S.J., 2008. Level set equations on surfaces via the closest point method. Journal of Scientific Computing 35, 219–240. doi:10.1007/s10915-008-9196-6.

Madhikar, P., Åström, J., Westerholm, J., Karttunen, M., 2018. CellSim3D: GPU accelerated software for simulations of cellular growth and division in three dimensions. Computer Physics Communications 232, 206–213. doi:10.1016/j.cpc.2018.05.024.

Mahmoud, A.H., Porumbescu, S.D., Owens, J.D., 2025. Dynamic mesh processing on the GPU. ACM Transactions on Graphics (TOG) 44, 1–19. doi:10.1145/3731162.

Maître, J.L., Turlier, H., Illukkumbura, R., Eismann, B., Niwayama, R., Nédélec, F., Hiiragi, T., 2016. Asymmetric division of contractile domains couples cell positioning and fate specification. Nature 536, 344–348. doi:10.1038/nature18958.

Marić, T., Kothe, D.B., Bothe, D., 2020. Unstructured un-split geometrical Volume-of-Fluid methods–a review. Journal of Computational Physics 420, 109695. doi:10.1016/j.jcp.2020.109695.

Marić, T., Marschall, H., Bothe, D., 2015. lentFoam–a hybrid level set/front tracking method on unstructured meshes. Computers & Fluids 113, 20–31. doi:10.1016/j.compfluid.2014.12.019.

Marthinsen, K., Hunderi, O., Ryum, N., 1996. The influence of spatial grain size correlation and topology on normal grain growth in two dimensions. Acta materialia 44, 1681–1689. doi:10.1016/1359-6454(95)00262-6.

McInerney, T., Terzopoulos, D., 1995. Topologically adaptable snakes, in: Proceedings of IEEE International Conference on Computer Vision, IEEE. pp. 840–845. doi:10.1109/ICCV.1995.466850.

McInerney, T., Terzopoulos, D., 1997. Medical image segmentation using topologically adaptable surfaces, in: International Conference on Computer Vision, Virtual Reality, and Robotics in Medicine, Springer. pp. 23–32. doi:10.1007/BFb0029221.

McInerney, T., Terzopoulos, D., 1999. Topology adaptive deformable surfaces for medical image volume segmentation. IEEE Transactions on Medical Imaging 18, 840–850. doi:10.1109/42.811261.

McInerney, T., Terzopoulos, D., 2000. T-snakes: Topology adaptive snakes. Medical image analysis 4, 73–91. doi:10.1016/S1361-8415(00)00008-6.

Meyer, M., Desbrun, M., Schröder, P., Barr, A.H., 2003. Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and mathematics III. Springer, pp. 35–57. doi:10.1007/978-3-662-05105-4_2.

Meyer, M., Whitaker, R., Kirby, R.M., Ledergerber, C., Pfister, H., 2008. Particle-based sampling and meshing of surfaces in multimaterial volumes. IEEE Transactions on Visualization and Computer Graphics 14, 1539–1546. doi:10.1109/TVCG.2008.154.

Mikula, K., Urbán, J., 2012. New fast and stable Lagrangean method for image segmentation, in: 2012 5th International Congress on Image and Signal Processing, IEEE. pp. 688–696. doi:10.1109/CISP.2012.6469852.

Misztal, M.K., 2010. Deformable simplicial complexes. Ph.D. thesis. Technical University of Denmark.

Misztal, M.K., Bærentzen, J.A., 2012. Topology-adaptive interface tracking using the deformable simplicial complex. ACM Transactions on Graphics (TOG) 31, 1–12. doi:10.1145/2167076.2167082.

Misztal, M.K., Erleben, K., Bargteil, A., Fursund, J., Christensen, B.B., Bærentzen, J.A., Bridson, R., 2013. Multiphase flow of immiscible fluids on unstructured moving meshes. IEEE transactions on visualization and computer graphics 20, 4–16. doi:10.1109/TVCG.2013.97.

Moes, N., Remacle, J.F., Lambrechts, J., Lé, B., Chevaugeon, N., 2023. The eXtreme Mesh deformation approach (X-MESH) for the Stefan phase change model. Journal of Computational Physics 477, 111878. doi:10.2139/ssrn.4097530.

Monaghan, J.J., 1994. Simulating free surface flows with SPH. Journal of computational physics 110, 399–406. doi:https://doi.org/10.1006/jcph.1994.1034.

Monnereau, C., Vignes-Adler, M., Pittet, N., 1999. Coarsening of a three-dimensional reconstructed foam under surface evolver. Philosophical Magazine B 79, 1213–1222. doi:10.1080/13642819908218316.

Moore, M., Wilhelms, J., 1988. Collision detection and response for computer animation, in: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, pp. 289–298. doi:10.1145/378456.378528.

Mor, A.B., Kanade, T., 2000. Modifying soft tissue models: Progressive cutting with minimal new element creation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. pp. 598–607. doi:10.1007/978-3-540-40899-4_61.

Mora, L.B., Gottstein, G., Shvindlerman, L., 2008. Three-dimensional grain growth: Analytical approaches and computer simulations. Acta Materialia 56, 5915–5926. doi:10.1016/j.actamat.2008.08.006.

Mosso, S.J., Swartz, B.K., Kothe, D.B., Ferrell, R.C., 1997. A parallel, volume-tracking algorithm for unstructured meshes, in: Parallel Computational Fluid Dynamics 1996. Elsevier, pp. 368–375. doi:10.1016/B978-044482327-4/50113-3.

Müller, M., 2009. Fast and robust tracking of fluid surfaces, in: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 237–245. doi:10.1145/1599470.1599501.

Muradoglu, M., Tryggvason, G., 2008. A front-tracking method for computation of interfacial flows with soluble surfactants. Journal of computational physics 227, 2238–2262. doi:10.1016/j.jcp.2007.10.003.

Nagai, T., Honda, H., 2001. A dynamic cell model for the formation of epithelial tissues. Philosophical Magazine B 81, 699–719. doi:10.1080/13642810108205772.

Nagai, T., Kawasaki, K., Nakamura, K., 1988. Vertex dynamics of two-dimensional cellular patterns. Journal of the physical society of Japan 57, 2221–2224. doi:10.1143/JPSJ.57.2221.

Nagai, T., Ohta, S., Kawasaki, K., Okuzono, T., 1990. Computer simulation of cellular pattern growth in two and three dimensions. Phase Transitions: A Multinational Journal 28, 177–211. doi:10.1080/01411599008207938.

Nakhmani, A., Tannenbaum, A., 2012. Self-crossing detection and location for parametric active contours. IEEE Transactions on Image Processing 21, 3150–3156. doi:10.1109/TIP.2012.2188808.

Ngoi, K.P., Jia, J., 1999. An active contour model for colour region extraction in natural scenes. Image and vision computing 17, 955–966. doi:10.1016/S0262-8856(98)00169-3.

Nguyen, T.T., Dahl, V.A., Bærentzen, J.A., 2019. Multi-phase image segmentation with the adaptive deformable mesh. Pattern Recognition Letters 117, 97–103. doi:10.1016/j.patrec.2018.12.009.

Nguyen, T.T., Dahl, V.A., Bærentzen, J.A., Trinderup, C.H., 2018. Multi-phase volume segmentation with tetrahedral mesh., in: BMVC, p. 258.

Nicolet, B., Jacobson, A., Jakob, W., 2021. Large steps in inverse rendering of geometry. ACM Transactions on Graphics (TOG) 40. doi:10.1145/3478513.3480501.

Nobari, M., Tryggvason, G., 1996. Numerical simulations of three-dimensional drop collisions. AIAA journal 34, 750–755. doi:10.2514/3.13136.

Nochetto, R.H., Walker, S.W., 2010. A hybrid variational front tracking-level set mesh generator for problems exhibiting large deformations and topological changes. Journal of Computational Physics 229, 6243–6269. doi:10.1016/j.jcp.2010.04.035.

Noh, W.F., Woodward, P., 1976. SLIC (simple line interface calculation), in: van de Vooren, A.I., Zandbergen, P.J. (Eds.), Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 330–340. doi:10.1007/3-540-08004-X_336.

O'Rourke, J., 1998. Computational geometry in C. Cambridge university press. doi:10.1017/CBO9780511804120.

Osher, S., Fedkiw, R., 2003. Level Set Methods and Dynamic Implicit Surfaces. Springer. doi:10.1007/b98879.

Palfinger, W., 2022. Continuous remeshing for inverse rendering. Computer Animation and Virtual Worlds 33, e2101. doi:10.1002/cav.2101.

Pan, H., Choi, Y.K., Liu, Y., Hu, W., Du, Q., Polthier, K., Zhang, C., Wang, W., 2012. Robust modeling of constant mean curvature surfaces. ACM Transactions on Graphics (TOG) 31, 1–11. doi:10.1145/2185520.2185581.

Pan, J., Long, T., Chirco, L., Scardovelli, R., Popinet, S., Zaleski, S., 2024. An edge-based interface tracking (EBIT) method for multiphase-flow simulation with surface tension. Journal of Computational Physics 508, 113016. doi:10.1016/j.jcp.2024.113016.

Pan, J., Long, T., Scardovelli, R., Popinet, S., Zaleski, S., 2025. A three-dimensional edge-based interface tracking (EBIT) method for multiphase-flow simulations. arXiv preprint arXiv:2509.26361 .

Pan, J., Long, T., Scardovelli, R., Zaleski, S., 2026. An unsplit scheme for interface advection in the edge-based interface tracking (EBIT) method. Journal of Computational Physics 557, 114838. doi:10.1016/j.jcp.2026.114838.

Panchal, D., Jayaswal, D., 2021. Computational paradigms for direct triangular surface remeshing. Computers & Graphics 94, 87–110. doi:10.1016/j.cag.2020.11.001.

Patkar, S., Aanjaneya, M., Karpman, D., Fedkiw, R., 2013. A hybrid Lagrangian-Eulerian formulation for bubble generation and dynamics, in: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 105–114. doi:10.1145/2485895.2485912.

Pauš, P., Beneš, M., 2009. Algorithm for topological changes of parametrically described curves, in: Proceedings of ALGORITMY, pp. 176–184.

Pavić, D., Campen, M., Kobbelt, L., 2010. Hybrid Booleans, in: Computer Graphics Forum, Wiley Online Library. pp. 75–87. doi:10.1111/j.1467-8659.2009.01545.x.

Peskin, C.S., 2002. The immersed boundary method. Acta numerica 11, 479–517. doi:10.1017/S0962492902000077.

Pons, J.P., Boissonnat, J.D., 2007a. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 1–8. doi:10.1109/CVPR.2007.383019.

Pons, J.P., Boissonnat, J.D., 2007b. A Lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space, in: Computer Graphics Forum, Wiley Online Library. pp. 227–239. doi:10.1111/j.1467-8659.2007.01029.x.

Popinet, S., coauthors, 2026. Basilisk. URL: https://basilisk.fr/.

Pozrikidis, C., 1992. Boundary integral and singularity methods for linearized viscous flow. Cambridge university press. doi:10.1017/CBO9780511624124.

Provot, X., 1997. Collision and self-collision handling in cloth model dedicated to design garments, in: Computer Animation and Simulation'97: Proceedings of the Eurographics Workshop in Budapest, Hungary, September 2–3, 1997, Springer. pp. 177–189. doi:10.1007/978-3-7091-6874-5_13.

Qiu, Y., Zhang, Q., 2025. A linear mars method for three-dimensional interface tracking. arXiv preprint arXiv:2512.07524 .

Quan, S., Lou, J., Schmidt, D.P., 2009. Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations. Journal of Computational Physics 228, 2660–2675. doi:10.1016/j.jcp.2008.12.029.

Quan, S., Schmidt, D.P., 2007. A moving mesh interface tracking method for 3D incompressible two-phase flows. Journal of Computational Physics 221, 761–780. doi:10.1016/j.jcp.2006.06.044.

Quiriny, A., Lambrechts, J., Moës, N., Remacle, J.F., 2024. X-Mesh: A new approach for the simulation of two-phase flow with sharp interface. Journal of Computational Physics 501, 112775. doi:10.1016/j.jcp.2024.112775.

Rajkotwala, A., Mirsandi, H., Peters, E., Baltussen, M., Van der Geld, C., Kuerten, J., Kuipers, J., 2018. Extension of local front reconstruction method with controlled coalescence model. Physics of Fluids 30. doi:10.1063/1.5008371.

Ramey, N.A., Corso, J.J., Lau, W.W., Burschka, D., Hager, G.D., 2004. Real-time 3D surface tracking and its applications, in: 2004 Conference on Computer Vision and Pattern Recognition Workshop, IEEE. pp. 34–34. doi:10.1109/CVPR.2004.425.

Razizadeh, M., Mortazavi, S., Shahin, H., 2018. Drop breakup and drop pair coalescence using front-tracking method in three dimensions. Acta Mechanica 229, 1021–1043. doi:10.1007/s00707-017-1958-5.

Regnault, P., 2023. Front-Tracking mesh adaptation for the simulation of two-phase flows with coalescence and breakup. Ph.D. thesis. Université Gustave Eiffel. doi:10.70675/d07a8699z29b3z4a23z8094z66203fbf8c59.

Rider, W.J., Kothe, D.B., 1998. Reconstructing volume tracking. Journal of computational physics 141, 112–152. doi:10.1006/jcph.1998.5906.

Runser, S., Vetter, R., Iber, D., 2024. SimuCell3D: three-dimensional simulation of tissue mechanics with cell polarization. Nature Computational Science 4, 299–309. doi:10.1038/s43588-024-00620-9.

Samadani, R., 1989. Changes in connectivity in active contour models, in: [1989] Proceedings. Workshop on Visual Motion, IEEE. pp. 337–343. doi:10.1109/WVM.1989.47127.

Sauter, S.A., Schwab, C., 2010. Boundary element methods. Springer. doi:10.1007/978-3-540-68093-2.

Saye, R.I., Sethian, J.A., 2012. Analysis and applications of the voronoi implicit interface method. Journal of Computational Physics 231, 6051–6085. doi:10.1016/j.jcp.2012.04.004.

Saye, R.I., Sethian, J.A., 2013. Multiscale modeling of membrane rearrangement, drainage, and rupture in evolving foams. Science 340, 720–724. doi:10.1126/science.1230623.

Scardovelli, R., Zaleski, S., 2000. Analytical relations connecting linear interfaces and volume fractions in rectangular grids. Journal of Computational Physics 164, 228–237. doi:10.1006/jcph.2000.6567.

Semushin, S., 1988. Flow calculation with a contact surface on a rectangular lattice.

Sethian, J.A., et al., 1999. Level set methods and fast marching methods. 2 ed., Cambridge University Press.

Shahbazi, K., Paraschivoiu, M., Mostaghimi, J., 2003. Second order accurate volume tracking based on remapping for triangular meshes. Journal of Computational Physics 188, 100–122. doi:10.1016/S0021-9991(03)00156-6.

Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S., 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3D shape synthesis. Advances in Neural Information Processing Systems 34, 6087–6101.

Shen, T., Munkberg, J., Hasselgren, J., Yin, K., Wang, Z., Chen, W., Gojcic, Z., Fidler, S., Sharp, N., Gao, J., 2023. Flexible isosurface extraction for gradient-based mesh optimization. ACM Transactions on Graphics (TOG) 42, 1–16. doi:10.1145/3592430.

Shewchuk, J.R., 1997. Adaptive precision floating-point arithmetic and fast robust geometric predicates. Discrete & Computational Geometry 18, 305–363. doi:10.1007/PL00009321.

Shewchuk, J.R., 1999. Lecture notes on geometric robustness. Eleventh International Meshing Roundtable , 115–126.

Shin, S., Abdel-Khalik, S., Daru, V., Juric, D., 2005. Accurate representation of surface tension using the level contour reconstruction method. Journal of Computational Physics 203, 493–516. doi:10.1016/j.jcp.2004.09.003.

Shin, S., Juric, D., 2002. Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. Journal of Computational Physics 180, 427–470. doi:10.1006/jcph.2002.7086.

Shin, S., Juric, D., 2007. High order level contour reconstruction method. Journal of mechanical science and technology 21, 311–326. doi:10.1007/BF02916292.

Shin, S., Juric, D., 2009. A hybrid interface method for three-dimensional multiphase flows based on front tracking and level set techniques. International Journal for Numerical Methods in Fluids 60, 753–778. doi:10.1002/fld.1912.

Shin, S., Yoon, I., Juric, D., 2011. The local front reconstruction method for direct simulation of two-and three-dimensional multiphase flows. Journal of Computational Physics 230, 6605–6646. doi:10.1016/j.jcp.2011.04.040.

Shirley, P., Ashikhmin, M., Marschner, S., 2009. Fundamentals of computer graphics. AK Peters/CRC Press. doi:10.1201/9781003050339.

Sifakis, E., Der, K.G., Fedkiw, R., 2007. Arbitrary cutting of deformable tetrahedralized objects, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 73–80. doi:10.5555/1272690.1272701.

Singh, R., Shyy, W., 2007. Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction. Journal of Computational Physics 224, 150–167. doi:10.1016/j.jcp.2006.12.026.

Smith, J.M., Dodgson, N.A., 2007. A topologically robust algorithm for boolean operations on polyhedral shapes using approximate arithmetic. Computer-Aided Design 39, 149–163. doi:10.1016/j.cad.2006.11.003.

Soares, A., Ferro, A., Fortes, M., 1985. Computer simulation of grain growth in a bidimensional polycrystal. Scripta metallurgica 19, 1491–1496. doi:10.1016/0036-9748(85)90157-7.

Stanculescu, L., Chaine, R., Cani, M.P., 2011. Freestyle: Sculpting meshes with self-adaptive topology. Computers & Graphics 35, 614–622. doi:10.1016/j.cag.2011.03.033.

Stanculescu, L., Chaine, R., Cani, M.P., Singh, K., 2013. Sculpting multi-dimensional nested structures. Computers & graphics 37, 753–763. doi:10.1016/j.cag.2013.05.010.

Stock, M.J., Dahm, W.J., Tryggvason, G., 2008. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. Journal of Computational Physics 227, 9021–9043. doi:10.1016/j.jcp.2008.05.022.

Stoeter, S.A., Papanikolopoulos, N., 2004. Closed dynamic contour models that split and merge, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, IEEE. pp. 3883–3888. doi:10.1109/ROBOT.2004.1308873.

Strain, J., 2001. A fast semi-Lagrangian contouring method for moving interfaces. Journal of Computational Physics 170, 373–394. doi:10.1006/jcph.2001.6740.

Sun, Y., Beckermann, C., 2007. Sharp interface tracking using the phase-field equation. Journal of Computational Physics 220, 626–653. doi:10.1016/j.jcp.2006.05.025.

Sussman, M., Puckett, E.G., 2000. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. Journal of computational physics 162, 301–337. doi:10.1006/jcph.2000.6537.

Syha, M., Weygand, D., 2010. A generalized vertex dynamics model for grain growth in three dimensions. Modelling and Simulation in Materials Science and Engineering 18, 015010. doi:10.1088/0965-0393/18/1/015010.

Szeliski, R., Tonnesen, D., Terzopoulos, D., 1993. Modeling surfaces of arbitrary topology with dynamic particles, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 82–87. doi:10.1109/CVPR.1993.340975.

Tan, Y., Qian, Y., Li, Z., Zhang, Q., 2026. A multiphase cubic mars method for fourth-and higher-order interface tracking of two or more materials with arbitrary topology and geometry. Journal of Scientific Computing 107, 77. doi:10.1007/s10915-026-03284-x.

The CGAL Project, 2025. CGAL User and Reference Manual. 6.1 ed., CGAL Editorial Board. URL: https://doc.cgal.org/6.1/Manual/packages.html.

Thomas, E.C., Hopyan, S., 2023. Shape-driven confluent rigidity transition in curved biological tissues. Biophysical Journal 122, 4264–4273. doi:10.1016/j.bpj.2023.10.001.

Toh, K.K.H., 1990. Algorithms for three-dimensional simulation of photoresist development. University of California, Berkeley.

Torres, D., Brackbill, J., 2000. The point-set method: front-tracking without connectivity. Journal of Computational Physics 165, 620–644. doi:10.1006/jcph.2000.6635.

Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., Jan, Y.J., 2001. A front-tracking method for the computations of multiphase flow. Journal of computational physics 169, 708–759. doi:10.1006/jcph.2001.6726.

Unverdi, S.O., Tryggvason, G., 1992. A front-tracking method for viscous, incompressible, multi-fluid flows. Journal of computational physics 100, 25–37. doi:10.1016/0021-9991(92)90307-K.

Valque, L., 2024. 3D Snap Rounding. Ph.D. thesis. Université de lorraine.

Valque, L., Lazard, S., 2025. Resolving self-intersections in 3D meshes while preserving floating-point coordinates, in: Computer Graphics Forum, Wiley Online Library. p. e70197. doi:10.1111/cgf.70197.

VanDerWoude, G.W., 2000. A three-dimensional front tracking algorithm for etching and deposition processes. State University of New York at Stony Brook.

Varanasi, K., Zaharescu, A., Boyer, E., Horaud, R., 2008. Temporal surface tracking using mesh evolution, in: European conference on computer vision, Springer. pp. 30–43. doi:10.1007/978-3-540-88688-4_3.

Vetter, R., Runser, S.V., Iber, D., 2024. PolyHoop: Soft particle and tissue dynamics with topological transitions. Computer Physics Communications 299, 109128. doi:10.1016/j.cpc.2024.109128.

Vorsatz, J., Rössl, C., Seidel, H.P., 2003. Dynamic remeshing and applications, in: Proceedings of the eighth ACM symposium on Solid modeling and applications, pp. 167–175. doi:10.1145/781606.781633.

Wakai, F., Enomoto, N., Ogawa, H., 2000. Three-dimensional microstructural evolution in ideal grain growth—general statistics. Acta Materialia 48, 1297–1311. doi:10.1016/S1359-6454(99)00405-X.

Wang, B., Ferguson, Z., Schneider, T., Jiang, X., Attene, M., Panozzo, D., 2021. A large-scale benchmark and an inclusion-based algorithm for continuous collision detection. ACM Transactions on Graphics (TOG) 40, 1–16. doi:10.1145/3460775.

Wang, M., Cong, M., Zhu, B., 2025. An interface tracking method with triangle edge cuts. Journal of Computational Physics 520, 113504. doi:10.1016/j.jcp.2024.113504.

Wang, T., Ling, H., Lang, C., Feng, S., Hou, X., 2019. Deformable surface tracking by graph matching, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 901–910. doi:10.1109/ICCV.2019.00099.

Weaire, D., Kermode, J., 1983. Computer simulation of a two-dimensional soap froth: I. method and motivation. Philosophical Magazine B 48, 245–259. doi:10.1080/13642818308228287.

Weaire, D.L., Hutzler, S., 1999. The physics of foams. Oxford University Press. doi:10.1093/oso/9780198505518.001.0001.

Weiler, K.J., 1986. Topological structures for geometric modeling. Rensselaer Polytechnic Institute.

Wen, J., Barbič, J., Kaufman, D.M., 2025. Optimal r-adaptive in-timestep remeshing for elastodynamics. ACM Transactions on Graphics (TOG) 44, 1–19. doi:10.1145/3731204.

Weygand, D., Brechet, Y., Lépinoux, J., 1998. A vertex dynamics simulation of grain growth in two dimensions. Philosophical Magazine B 78, 329–352. doi:10.1080/13642819808206731.

Weygand, D., Bréchet, Y., Lépinoux, J., Gust, W., 1999. Three-dimensional grain growth: a vertex dynamics simulation. Philosophical Magazine B 79, 703–716. doi:10.1080/13642819908205744.

Wicke, M., Ritchie, D., Klingner, B.M., Burke, S., Shewchuk, J.R., O'Brien, J.F., 2010. Dynamic local remeshing for elastoplastic simulation. ACM Transactions on graphics (TOG) 29, 1–11. doi:10.1145/1778765.1778786.

Wojtan, C., Müller-Fischer, M., Brochu, T., 2011. Liquid simulation with mesh-based surface tracking, in: ACM SIGGRAPH 2011 Courses, pp. 1–84. doi:10.1145/2037636.2037644.

Wojtan, C., Thürey, N., Gross, M., Turk, G., 2009. Deforming meshes that split and merge, in: ACM SIGGRAPH 2009 papers, pp. 1–10. doi:10.1145/1576246.1531382.

Wojtan, C., Thürey, N., Gross, M., Turk, G., 2010. Physics-inspired topology changes for thin fluid features. ACM Transactions on Graphics (TOG) 29, 1–8. doi:10.1145/1778765.1778787.

Wojtan, C., Turk, G., 2008. Fast viscoelastic behavior with thin features, in: ACM SIGGRAPH 2008 papers, pp. 1–8. doi:10.1145/1399504.1360646.

Woop, S., Benthin, C., Wald, I., 2013. Watertight ray/triangle intersection. Journal of Computer Graphics Techniques (JCGT) 2, 65–82.

Wyvill, G., McPheeters, C., Wyvill, B., 1986. Data structure for soft objects. The visual computer 2, 227–234. doi:10.1007/BF01900346.

Yang, J., Jia, X., Yan, D.M., 2023. Topology guaranteed B-spline surface/surface intersection. ACM Transactions on Graphics (TOG) 42, 1–16. doi:10.1145/3618349.

Yang, M., Ye, J., Ding, F., Zhang, Y., Yan, D.M., 2018. A semi-explicit surface tracking mechanism for multi-phase immiscible liquids. IEEE Transactions on Visualization and Computer Graphics 25, 2873–2885. doi:10.1109/TVCG.2018.2864283.

Yang, X., Yun, S., Guan, Q., Gao, H., 2024. rupMC: a ray-unit parallel marching cubes algorithm on CPU/GPU heterogeneous architectures. International Journal of Digital Earth 17, 2340583. doi:10.1080/17538947.2024.2340583.

Yoon, I., Shin, S., 2010. Tetra-marching procedure for high order level contour reconstruction method. WIT Transactions on Engineering Sciences 69, 507–518. doi:10.2495/AFM100441.

Youngs, D.L., 1982. Time-dependent multi-material flow with large fluid distortion. Numerical methods for fluid dynamics .

Youngs, D.L., 1984. An interface tracking method for a 3D Eulerian hydrodynamics code. Atomic Weapons Research Establishment (AWRE) Technical Report 44, 35.

Zaharescu, A., Boyer, E., Horaud, R., 2007. Transformesh: a topology-adaptive mesh-based approach to surface evolution, in: Asian Conference on Computer Vision, Springer. pp. 166–175. doi:10.1007/978-3-540-76390-1_17.

Zaharescu, A., Boyer, E., Horaud, R., 2010. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 823–837. doi:10.1109/TPAMI.2010.116.

Zhang, P., Law, C.K., 2011. An analysis of head-on droplet collision with large deformation in gaseous medium. Physics of Fluids 23. doi:10.1063/1.3580754.

Zhang, P., Marschner, Z., Solomon, J., Tamstorf, R., 2023. Sum-of-squares collision detection for curved shapes and paths, in: ACM SIGGRAPH 2023 conference proceedings, pp. 1–11. doi:10.1145/3588432.3591507.

Zhang, Q., 2018. Fourth-and higher-order interface tracking via mapping and adjusting regular semianalytic sets represented by cubic splines. SIAM Journal on Scientific Computing 40, A3755–A3788. doi:10.1137/17M1149328.

Zhang, Q., Fogelson, A., 2014. Fourth-order interface tracking in two dimensions via an improved polygonal area mapping method. SIAM Journal on Scientific Computing 36, A2369–A2400. doi:10.1137/140951886.

Zhang, Q., Fogelson, A., 2016. MARS: An analytic framework of interface tracking via mapping and adjusting regular semialgebraic sets. SIAM Journal on Numerical Analysis 54, 530–560. doi:10.1137/140966812.

Zhang, Q., Li, Z., 2020. Boolean algebra of two-dimensional continua with arbitrarily complex topology. Mathematics of Computation 89, 2333–2364. doi:10.1090/mcom/3539.

Zhang, Q., Liu, P.L.F., 2008. A new interface tracking method: The polygonal area mapping method. Journal of Computational Physics 227, 4063–4088. doi:10.1016/j.jcp.2007.12.014.

Zheng, J., Luo, Z., Li, M., 2025. Robust and efficient penetration-free elastodynamics without barriers. ACM Transactions on Graphics doi:10.1145/3811035.

Zheng, S., 2010. An intensive restraint topology adaptive snake model and its application in tracking dynamic image sequence. Information Sciences 180, 2940–2959. doi:10.1016/j.ins.2010.04.030.

Zheng, W., Yong, J.H., Paul, J.C., 2009. Simulation of bubbles. Graphical Models 71, 229–239. doi:10.1016/j.gmod.2009.08.001.

Zhou, Q., Grinspun, E., Zorin, D., Jacobson, A., 2016. Mesh arrangements for solid geometry. ACM Transactions on Graphics (TOG) 35, 1–15. doi:10.1145/2897824.2925901.

Zwicke, F., Eusterholz, S., Elgeti, S., 2017. Boundary-conforming free-surface flow computations: Interface tracking for linear, higher-order and isogeometric finite elements. Computer Methods in Applied Mechanics and Engineering 326, 175–192. doi:10.1016/j.cma.2017.08.022.