Title
Smooth, Volume-Accurate Material Interface Reconstruction

Journal
IEEE Transactions on Visualization and Computer Graphics, 16

Authors
Anderson, John C.
Garth, Christoph
Duchaineau, Mark A.
et al.

Publication Date
2010

Peer reviewed

Smooth, Volume-Accurate Material Interface Reconstruction

John C. Anderson, *Member, IEEE*, Christoph Garth,
Mark A. Duchaineau, *Member, IEEE*, and Kenneth I. Joy, *Member, IEEE*

Abstract—A new material interface reconstruction method for volume fraction data is presented. Our method is comprised of two components: first, we generate initial interface topology; then, using a combination of smoothing and volumetric forces within an active interface model, we iteratively transform the initial material interfaces into high-quality surfaces that accurately approximate the problem's volume fractions. Unlike all previous work, our new method produces material interfaces that are smooth, continuous across cell boundaries, and segment cells into regions with proper volume. These properties are critical during visualization and analysis. Generating high-quality mesh representations of material interfaces is required for accurate calculations of interface statistics, and dramatically increases the utility of material boundary visualizations.

Index Terms—Material interface reconstruction, volume fractions, embedded boundary, active interfaces, segmentation.

1 Introduction

Interface reconstruction and tracking is an important problem with broad application. Interface problems arise in combustion applications, climate studies, and astrophysics. Scientific simulations of fluid transport naturally produce interfaces – the boundaries between sloshing fluids in tanks, between cavity and casting in mold filling applications, and waves breaking on shorelines.

The material interface reconstruction (MIR) problem is to construct smooth, continuous boundary interfaces between regions of homogeneous materials. Depending on the type of input data two major MIR problems exist: reconstruction based on labeled voxels or vertices (MIRLV), and reconstruction based on cell-based volume fractions (MIRVF). This paper addresses the latter.

In many of the above applications it is necessary to reconstruct interfaces between multiple material regions of known volume. Multi-fluid hydrodynamics simulation codes, for example, often produce datasets in which cell-centered scalar quantities report the fractional volume of each material contained within the cell. In this paper we develop a method to create high-quality MIRVF reconstructions – i.e., boundary interfaces between different material regions – for visualization and analysis purposes in applications that depend upon volume fraction information.

Volume fraction data exists within a spatial domain that has been decomposed into a finite grid of cells C. In an n-material setting, each cell $c \in C$ has an associated tuple $V_c = (v_1, \ldots, v_n)$, where the value v_i is the fractional volume of material i within the cell. Volume fractions are non-negative ($v_i \geq 0$), and form a partition of unity over the volume of the cell ($\sum_{i=1}^{n} v_i = 1$). *Mixed* cells have multiple non-zero volume fractions, while *pure* cells contain only a single material.

- *John C. Anderson is with Makai Ocean Engineering, Inc. E-mail: John.Anderson@makai.com.*
- *Christoph Garth and Kenneth I. Joy are with the Institute for Data Analysis and Visualization at University of California, Davis. E-mail: {cgarth,kijoy}@ucdavis.edu.*
- *Mark A. Duchaineau is with the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. E-mail: duchaine@llnl.gov.*

Material interface reconstruction using volume fractions is an under-constrained problem. There are an infinite number of boundary interfaces that can satisfy a given set of volume fractions; consider Figure 1. Several attempts have been made to recast the MIRVF problem into a MIRLV, isosurface-like problem (as discussed in Section 2); but this creates significant errors and reconstructions that are inconsistent with the input data. After some study, one realizes that material interface reconstruction based on volume fractions is fundamentally different from the problem of reconstructing material interfaces from labeled voxels without volume fraction information. The primary difference is that there are no explicit material labels at the mesh vertices, only cell-based volume fractions. Volume fraction information must be used by MIRVF algorithms to generate interfaces that segment space into homogeneous material regions with specific volume.

Our goal is to develop a high-quality MIRVF algorithm for visualization and analysis, that can be broadly applied to volume fraction data from many sources. In this setting, the reconstructed material interfaces should be piecewise linear, continuous across cell boundaries, and segment cells into regions that approximate the given volume fractions with low error. Generating high-quality, volume-accurate material interfaces allows for meaningful calculations of interface surface statistics such as area and curvature, allows us to texture accurately the interface mesh with other data items from the simulations (temperature, pressure, etc.), and dramatically increases the utility of material interface visualizations.

We introduce a new material interface reconstruction method based upon a volume-adaptive active interface model that adjusts an initial boundary approximation toward a better solution. Our algorithm consists of two general stages:

- We generate the topology of the material interfaces using a pipeline that includes a rule-based examination of mixed cells, and a robust discretization method for ambiguous regions; and,
- We employ a *volume-adaptive* active interface model that balances internal curve/surface metrics with external volume objectives to iterate the interfaces toward a low-error solution.

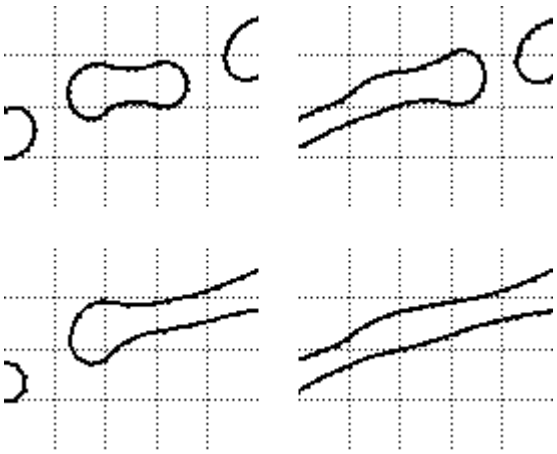


Fig. 1. Four reconstructions for the same volume fraction data.

Section 3 begins by describing some of the difficulties of topology initialization: each mixed cell must be segmented into a number of homogeneous material regions based upon its volume fractions; adjacent pure cells of different materials must be separated; and interfaces should be continuous across cell boundaries. Further, the under-determined nature of this problem that leads to topological ambiguity must be addressed. We detail a robust topology generation pipeline which satisfies these requirements for two- and three-dimensional problems with multiple materials.

Unlike previous methods that force an interface topology and sacrifice volume-preservation (e.g., by recasting MIRVF as an isosurfacing-like problem), our approach largely decouples topology generation from volume preservation. The initial interface topology that we generate is simply a starting point from which we can improve our MIRVF solution.

From the initial cell-level interface topology, we create a piecewise linear surface mesh, as described in Section 4. We then develop a *volume-adaptive* active interface model in Section 5 to update the location of the mesh control points such that the interfaces approach a low-error solution to the MIRVF problem. At the heart of this model are smoothing and volumetric forces that iteratively adjust the surface mesh, simultaneously attempting to satisfy the volume fractions prescribed by the initial problem while improving the mesh quality. Results for 2D and 3D problems are presented in Section 6.

2 Related Work

Our work focuses upon the reconstruction of material interfaces from volume fraction data by means of evolving, dynamic interfaces. Section 2.1 discusses past work in binary- and multi-label surface extraction (MIRLV), as well as simulation- and visualization-oriented methods for reconstructing surfaces from volume fraction information – rather than labels – over a meshed domain (MIRVF). We also discuss dynamic surface models in Section 2.2 to familiarize readers with active contour/interface methods – an important component of our approach.

2.1 Material Interface Reconstruction

Segmented grid data are abundant. The reconstruction of material interfaces from segmented grid data has been a staple problem in the visualization community for many years; the problem, simply stated, is to partition a labeled mesh into regions with homogeneous material labels (MIRLV). Binary labelings can be dealt with algorithms based on Marching Cubes [1], while multi-label data can be handled with additional rules and extended lookup tables [2], [3], [4], [5]. Further, the topic of computing multiple isosurfaces through a mesh has been studied by Nielson and Sung [6]. Beyond case tables, adaptive particle-based sampling has recently been used to segment labeled, multi-material data [7].

The problem we address in this paper, on the other hand, is the reconstruction of surfaces based upon volumetric constraints (MIRVF). This class of material interface reconstruction algorithms has received less attention from the visualization community, however it is an essential component in the Volume-of-Fluid (VOF) method for Eulerian and arbitrary Lagrangian-Eulerian multi-fluid hydrodynamic flows [8], [9], [10]. In a VOF simulation, fractional material volumes are maintained for each cell. To advance the simulation, interface geometry is reconstructed in order to calculate the flux of material between cells. Storing per-cell volumes, rather than explicit boundary interface geometry, eases the simulation of complicated flows and provides for a straightforward means of mass conservation.

Material interface reconstruction methods can be broadly split into simulation and visualization approaches. The reconstruction methods employed within VOF simulation are a crucial part of accurately advecting materials [11], and understandably the focus is upon volume conservation and convergence. In the visualization setting, the goal is to produce high-quality surface meshes that approximate the volume fractions with low error and integrate into the visualization and analysis framework of existing tools (e.g., [12]).

One of the first simulation methods is Simple Line Interface Calculation (SLIC), described by Noh and Woodward [13], where cells are simply partitioned with axis-aligned planes, such that the total material volume in each cell is correct. The Piecewise Linear Interface Calculation (PLIC) algorithm of Youngs [14] is similar to SLIC, however each cell is instead partitioned by planes aligned to local material "gradients." While PLIC is fast and preserves volume fractions, the reconstruction is discontinuous across cell boundaries and thus not suitable for visualization (see Figure 7(a)). Furthermore, PLIC is ambiguous for three or more materials due to the ordering of its binary segmentations.

Pilliod and Pucket [11] describe two modifications to the PLIC method, both of which use least-squares to minimize the error of approximately linear interfaces. Garimella et al. [15] demonstrate how to fix certain local topological inconsistencies in PLIC reconstructions. Dyadechko and Shashkov [16] and Schofield et al. [17] present interface reconstruction algorithms for volume fraction data augmented with material centroid information. These methods either inherit the partitioning ambiguity of PLIC, or produce discontinuous interface

boundaries.

While simulation methods such as PLIC minimize interface geometry and have useful convergence properties, they are ill-suited for visualization purposes due to the "soup" of cell-cutting geometry produced. As a result, there are a number of visualization-oriented MIRVF algorithms. Several such methods have attempted to recast MIRVF as a multi-labeled segmentation problem (MIRLV). Bonnell et al. [18], [19] move the segmentation problem to the dual mesh by assigning volume fractions from the original mesh's cells to vertices of the dual mesh. They calculate intersections using barycentric interpolation in the space of the volume fractions. One problem with this approach is that interfaces are not calculated from the original data, but from a dual grid, which induces significant error. Meredith averages volume fractions to mesh vertices, and then utilizes a multi-label isosurfacing-like approach [20]. Both methods, however, miss small scale features entirely (e.g., filaments and thin shells), have problems with many materials, and do not preserve volume fractions.

Anderson et al. [21] take a discrete approach to MIRVF based upon optimizing the labeling of fractional volume elements in a discretization of the original spatial domain. This approach produces useful interface topology with relatively low volume error, but their final boundary surfaces are "blocky" and not suitable for analysis. Later in this work, we consider the use of a labeled discretization as a means to disambiguate interface topology in complex, mixed material regions.

2.2 Dynamic Surface Models

Active contours – or *snakes* – were first proposed by Kass et al. [22]. These contours are parametric curves constructed with an energy functional that achieves a minimum value near a "boundary." Snakes have been utilized in numerous applications due to their segmentation capabilities [23], and in three dimensions snakes become an alternative surface reconstruction method. Cohen and Cohen [24], [25] published the initial extensions to achieve active surfaces – called *balloons* – and used them in three dimensions to extract facial skin from MRI volume data. Takanashi et al. [26], [27] directly extended the definition of snakes from curves to surfaces, creating an "active net" method, which was used to segment and extract muscle tissue from the visible human dataset. Gibson [28] uses a three-dimensional surface net to segment binary data smoothly. Ahlberg [29] also provides a good discussion of the extension of active contours to three dimensions. In this paper, we use the term *active interface* to mean an explicit, dynamic surface in either two or three dimensions.

The level set method, introduced by Osher and Sethian [30], represents dynamic interfaces as the zero level set of a time-dependent, implicit function. By solving the equations of motion in an adaptive, narrow-band Eulerian grid [31], the level set method is a computationally efficient solution for a number of fluid simulation, surface reconstruction, and segmentation applications – especially where surface topology change is required (see [32], [33], [34], [35]). Like active contours, the level set method scales well from two to three dimensions, and recent work has focused upon handling multiple spatial regions or fluids; in particular: by using one level set per region (see [36], [37], [38], [39]), or combinatorically using n level sets to represent 2^n material regions as done by Vese and Chan [40].

Both explicit and implicit dynamic surface models have been widely used in a range of applications. We have elected to use mesh-based active interfaces [29]. As we show in Section 5, an explicit scheme allows for straightforward calculations of local forces, and updates to the current per-cell reconstruction error; moreover, meshed surfaces are ideally suited for later use in the visualization and analysis pipeline. The choice of an explicit versus implicit surface model is further discussed in Section 7.

3 Material Interface Topology

Generating the initial surface topology for a volume fraction problem comprises the first stage of our reconstruction process. The *topology* of a surface describes its fundamental shape, such as the number of components and holes in the surface. The *embedding* of the surface refers to a geometric representation with specific spatial location, often represented explicitly as a mesh of vertices and faces. In the MIRVF problem, our initial focus is on generating cell-level interface topology – i.e., the general configuration of boundary surfaces within each cell – for mixed cells in two and three dimensions.

Consider a cell with two materials A and B: possible, valid topologies might range from a simple continuous interface between A and B, to multiple disconnected "islands" of A within B. The embedding of the continuous interface topology could be linear or curved, and the islands might be round or ellipsoidal. These are unknowns that make material interface reconstruction difficult: for any mixed material cell, there are limitless topologies and embeddings that might satisfy the volume fractions.

Figure 1 shows four possible reconstructions for the same volume fraction data. Our solution to the topology problem is a pipeline of three topology generation methods to initialize per-cell material interface topology. In the first pass, we use the *boundary* method to extract coarse, mesh-aligned interface topology. Next, the *rule-based marching* method – which can handle common binary- and multi-labeled segmentation cases – is applied across the mixed cells of the volume fraction data. This stage initializes the topology in cells containing few materials and simple interface configurations. Finally, the *discretized boundary* method is applied in complex regions where ambiguity caused by fine-scale features or many materials makes case-table analysis difficult. Figure 2 provides a working example of this pipeline.

3.1 Boundary Method

Our topology generation pipeline starts with the extraction of the most basic material interfaces. *Pure* cells contain only a single material; we start by extracting interfaces between pure cells of different material. Given two neighboring cells c_1 and c_2, with volume fractions of 1.0 for materials A and B, respectively, the shared face f between the cells is part

of the *A*-*B* material interface. Such interfaces often arise in simulation, where it is common to initialize each cell as a homogeneous material; mixed cells are then the result of advected material interfaces [8], [9]. These trivial boundaries are often required to form complete, closed material regions, and extracting them at this stage simplifies the next stage of the topology generation pipeline.

3.2 Rule-based Marching Method

While MIRVF is a fundamentally different problem than isosurfacing, it is often the case that simple topologies are sufficient for interface reconstruction. In many simulation codes, for example, a goal is to have approximately linear interfaces within each cell [11]; correspondingly, the average cell size is set (or adapted) based upon the expected interface curvature and complexity. In this situation, the vast majority of cells in a volume fraction dataset will either have no interface, or relatively simple interfaces between a few materials. In this section, we develop a rule-based "marching"-style segmentation method to generate cell-level interface topology based upon this observation.

The *rule-based marching* method applies a small set of rules to a problem's volume fractions in order to derive a partial vertex labeling, along with edge and cell characteristics. From this labeling, it is possible to derive material interface configurations from binary- and multi-label segmentation lookup tables. One of the primary differences between our *rule-based marching* method and previous isosurfacing approaches for MIRVF is that vertex labels are implied, rather than forced, from the volume fraction data. Bonnell et al. [18], [19] cast volume fraction data onto a dual mesh, while Meredith [20] forces vertex labels through volume averaging. Our approach, on the other hand, is to construct a partial labeling through the application of labeling rules; cell-level topologies from a lookup table are only used when they are suitable in the context of the problem's volume fractions.

Consider the labeling L, where $L(v)$ is the label of vertex v. In a complete labeling, each vertex in an n material problem has a known material label in the set $\{1,\ldots,n\}$. Partial labelings, however, allow vertices to have an unknown material:

$$L(v) = \begin{cases} 1,\ldots,n & \text{if the material at vertex } v \text{ is known, or} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

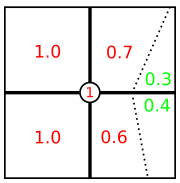
Volume fraction data is without vertex information, and thus the initial labeling of each vertex is undefined.

The first step of the *rule-based marching* method is to label mesh vertices using the volume fraction data. For each vertex in the mesh we evaluate the following rules:

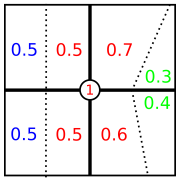
Pure Cells If all pure cells neighboring vertex v are of material i, then $L(v) = i$; if there are neighboring pure cells of different material, however, then $L(v)$ is undefined.

Neighborhood Consider the set of materials M common to all (pure and mixed) cells neighboring vertex v. If M contains a single material then the vertex label is set to that material.

These vertex labeling rules are illustrated for the central vertex in the following diagrams (where material one is red, and the dashed lines are hypothetical interfaces):



Pure Cells



Neighborhood

The **pure cells** rule is designed to "push" the material of pure cells onto neighboring vertices, without promoting arbitrary material precedence. (Note that interfaces between pure cells have already been extracted by the *boundary* method.) The **neighborhood** vertex labeling rule is useful for labeling vertices in the presence of thin filaments and shells, where the feature in question spans one or more vertices but not an entire cell.

From vertices, we move on to the edges of the mesh. Consider a mesh edge e between vertices v_1 and v_2. In typical marching schemes, the number of interface crossings χ along e is known *a priori* based upon the vertex labeling: edges with matching labels have no intersection ($\chi = 0$), while edges with mismatched labels have one intersection ($\chi = 1$). In the MIRVF problem more flexibility is required; thin filaments, shells, fine-scale features, and multiple materials can produce *multiple* intersections along a single mesh edge. Here, edges with matching vertex labels are either not intersected, or intersected *more* than once ($\chi = 0$ or $\chi \geq 2$); edges with mismatched vertex labels must have one or more intersections ($\chi \geq 1$).

Edge labeling rules are used to determine – to the extent possible – the number of interface crossings along each mesh edge. In our work, we use the following pair of rules:

Pure Cells The edge e has no intersections if any neighboring cell along e is pure.

Neighborhood Consider the set of materials M common to all (pure and mixed) neighboring cells along the edge e. If M contains a single material then e has no intersection.

Largely analogous to the vertex labeling rules above, edge labeling rules are based on the observation that edge intersections are building-blocks for defining material regions that span cells; i.e., if an edge is intersected by an interface, then there are at least two materials along that edge and those materials must be present in the neighboring cells. The **neighborhood** rule can also be used to derive certain vertex labels: if edge e (connecting vertex v_1 to vertex v_2) has no intersections, then the labels of its end vertices must match; i.e., $L(v_1) = L(v_2)$. For three-dimensional problems these rules translate naturally to cell faces as well.

Once we have applied the above rules we are left with a partially labeled mesh. Figure 2(b) shows the results of this labeling process when applied to the volume fractions derived from the three-material model problem shown in 2(a). The majority of vertices were labeled using the **pure cells** vertex labeling rule, however the circled vertices were labeled with

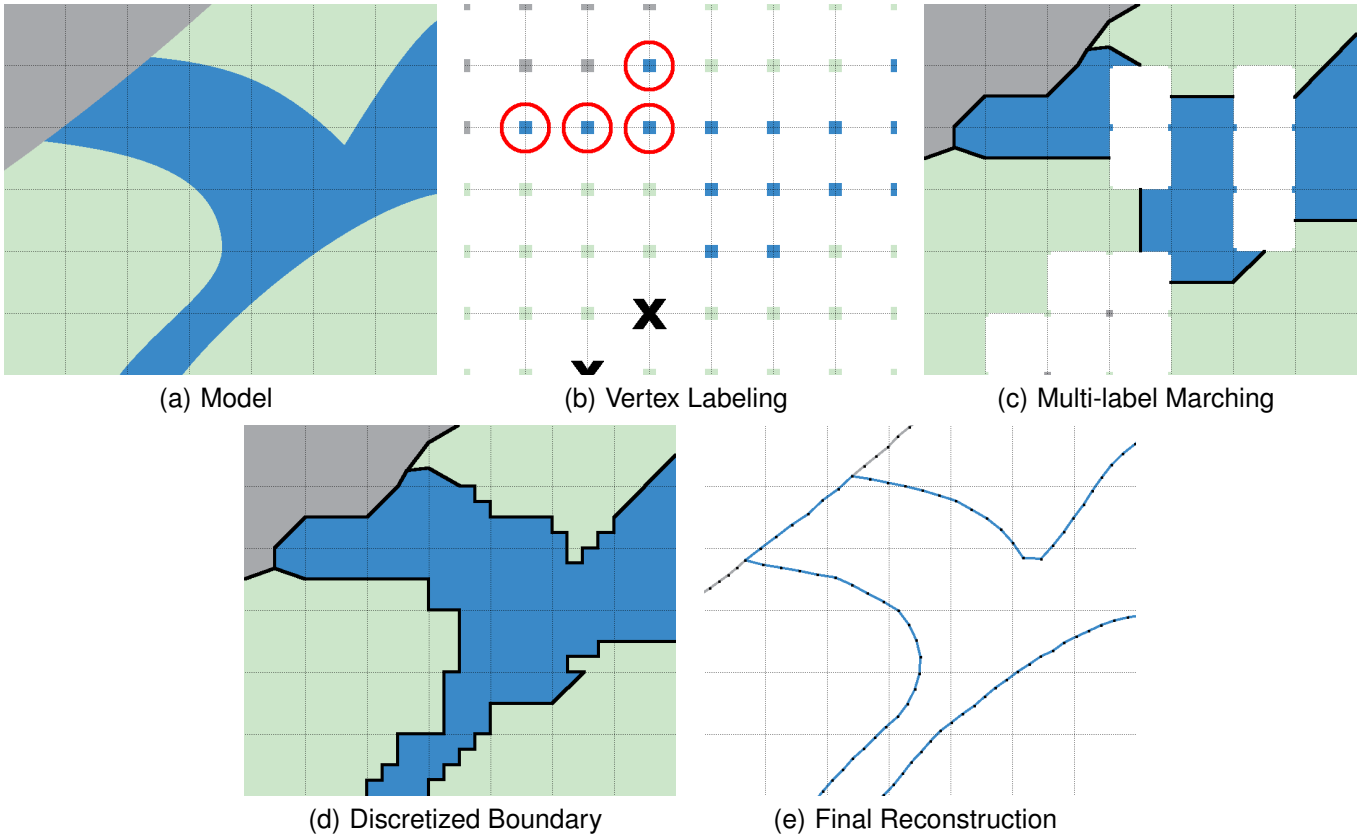


Fig. 2. The first step toward a material interface solution is the generation of initial surface mesh topology. In (a), we show an example 3-material problem. The *rule-based marching* method is used to (b) label the vertices and (c) fill in topology from binary- and multi-label segmentation lookup tables. In (b) the majority of vertices were labeled using the **pure cells** vertex labeling rule, however the circled vertices were labeled with the **neighborhood** rule, and the two vertices marked with an "X" could not be labeled using our rules. In (c), topology has been filled from the case table shown in Figure 3; blank cells could not be initialized due to ambiguity. The *discretized boundary* method is applied in (d) to generate topology for the remaining, uninitialized cells. In (e) we show our proposed reconstruction after applying volume-adaptive active interface optimization.

the **neighborhood** rule, and the two vertices marked with an "X" could not be labeled using our rules.

The next step in the *rule-based marching* is to generate per-cell interface topology. As with other marching-style segmentation methods, we iterate over each cell in the mesh incrementally adding surface fragments to our material interfaces. To find the appropriate surface fragment(s) to add to the interface for each cell, we perform a lookup using the cell's labeling configuration in a binary- or multi-label segmentation table.

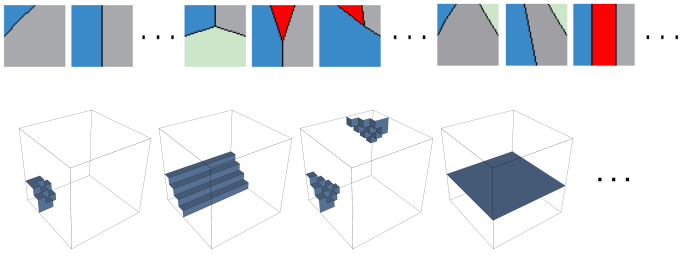


Fig. 3. Partial 2D (top) and 3D (bottom) case tables for per-cell interface generation used in the *rule-based marching* method.

Consider the cell c. If c has an unlabeled vertex v – i.e., $L(v)$ is undefined – then the cell topology is left uninitialized. Otherwise, if all vertices are labeled, then we perform a lookup in the segmentation case table. In most cases, the returned surface fragment(s) from the case table will properly segment the cell into enough regions to provide for a valid MIRVF solution. Unlike typical binary- and multi-label segmentation, however, two situations might occur: the case table might indicate ambiguity, or more materials might be needed. In both of these situations the cell's interface topology is left uninitialized. To test if more materials are required, let M be the set of materials with nonzero volume fractions in cell c, and let M' be the set of material labels for the vertices of the cell. If $M \neq M'$ then c has "residual" material. The implication is that there are either internal "pockets" of material within the cell or there are multiple intersections along one or more of the edges (or faces) of the cell.

Figure 3 partially illustrates the case tables we use to generate interfaces in 2D (top) and 3D (bottom). We have created our 2D table to allow multiple intersections along mesh edges, and for extra material regions within cells in certain configurations – e.g., a triple point where one edge of the cell is intersected twice. In the 2D case table in Figure 3, residual material regions have been colored red. We also mark cases in the table as "ambiguous" if there are multiple valid

ways to segment a labeling (consider the ambiguity addressed by Nielson and Hamann for marching cubes [41], and the complexity introduced by allowing multiple intersections per edge). To construct the 2D table we use a recursive scheme that first labels vertices, then intersects edges, and finally connects those intersections with interface segments – this process is straightforward (and finite), because at each stage there is a limited set of actions that can be performed given a cell's current configuration. Constructing the 3D case table is slightly more involved – however, because its construction depends upon a more complete view of our topology generation process, we postpone that discussion to Section 4.

To generate cell-level interface topology in this stage, the cell's vertices must be labeled, the case table must not indicate ambiguity, and the materials provided by the returned segmentation must match the cell's material requirements. Figure 2(c) shows the result of *rule-based marching* over our running 2D example. Interface configurations typically seen in isosurfacing problems, as well as some "T" junctions between three materials have been captured. Several cells remain uninitialized, however, either as a result of uncertain vertex labels, or because an ambiguous choice would have been required given the available segmentation topologies. In Figure 2(c), blank cells represent those that could not be initialized; the final stage of our pipeline, however, is able to generate topology in these cells.

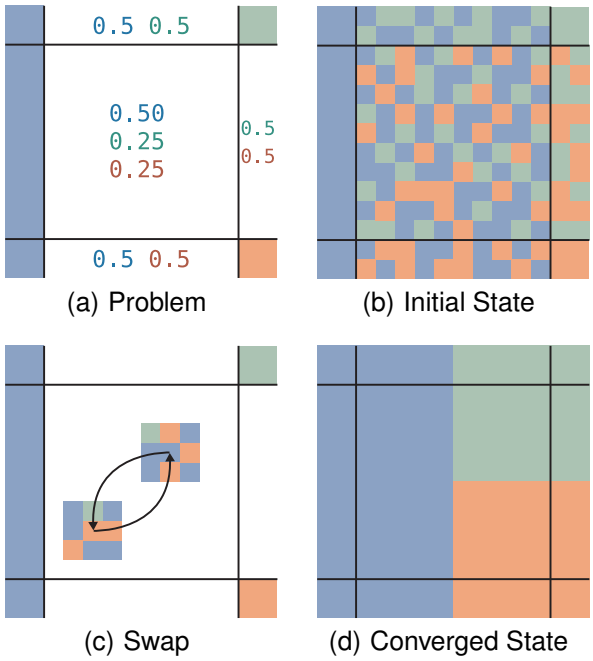


Fig. 4. Overview of the discrete boundary method introduced by Anderson et al. [21]. The mixed cell with given volume fractions (a) is discretized and labeled (b). Optimization of this labeling through swaps (c) results in coalesced regions that define material boundaries.

3.3 Discretized Boundary Method

We solve for the topology of any remaining uninitialized cells using a *discretized boundary* method. In this stage of the topology generation pipeline, we apply the discretized labeling method of Anderson et al. [21] on a per-cell level. Figure 4 provides an overview of this topology generation method. Cells that have been left uninitialized by previous stages in the pipeline are subdivided into small, fractional volume elements at a discretization resolution D. These "subcells" are then labeled by a material, where the number of subcells with label i is proportional to the volume of material i within the cell. Next, we perform energy minimizing optimization to improve the labeling; by using a "quenched" Potts-model energy, the labeling energy will monotonically decrease and rapidly converge. Here, "quenched" refers to simulated annealing with a 0 temperature (see [21] for details). Once the labeling has converged, coincident faces between subcells with different labels are extracted to become part of the initial material interfaces. Figure 2(d) illustrates the use of the *discretized boundary* method to generate topology within ambiguous cells left over from the *rule-based marching* method.

4 Initialization

At this point in the reconstruction process, each mixed cell has been attributed an initial cell-level interface topology. Before applying our volume-adaptive active interface model to improve the topological embedding, however, a single mesh-based representation of the interfaces must be created. To represent the interface, we use a mesh data structure with vertex-face incidence information (see [42]), which facilitates merging cell-level interface topologies into a single material interface mesh.

We begin by combining the interfaces produced by the *boundary* and *rule-based marching* methods. These interfaces are disjoint and thus easy to merge – the former are derived from shared cell faces between pure cells, while the latter are formulated within and between mixed cells. To perform the merge, the vertices and faces produced during these two stages are inserted into a single mesh data structure. Next, however, we must integrate the interfaces produced by the *discretized boundary* stage.

In two dimensions, adding the discretized interfaces is simple. First, we construct a closed, mesh-based representation of sub-cellular faces produced by the *discretized boundary* method. Next, we create the union of this mesh with the mesh derived from the *boundary* and *rule-based marching* stages. Discarding any degenerate faces – i.e., faces that create a partition between regions of the same material – will then yield a consistent material interface mesh. While some mesh processing is required, the operations to merge topologies in 2D are fairly straightforward. Figure 2(d) shows material interfaces (in black) that result from combining interfaces from the *rule-based marching* and *discretized boundary* methods.

In 3D, however, merging meshes can be more difficult; depending upon the output of the *rule-based marching* stage significant re-meshing might be needed to obtain a crack-free interface. To simplify our implementation and avoid re-meshing, we construct our 3D case table in such a way that the surfaces returned from *rule-based marching* can be easily joined with interfaces produced during the *discrete boundary* stage. This construction process is directly based upon the

work of Hege et al. [2] (however, without a simplification step); here we describe the initialization of a single entry in the 3D case table.

Given a hexahedral input cell, we label the cell's eight vertices according to the table entry being considered. Next, the cell is discretized at a resolution D matching the resolution used in the *discrete boundary* stage of topology initialization. The vertex labels are then used to interpolate a probability function for each label to the inside of the cell (sampled at the subcells in the discretization). This probability function indicates how strongly each subcell "belongs" to a label. After interpolation, each subcell is attributed a material corresponding to the label with maximal probability. (Note that for binary-labeling cases we slightly bias the probabilities toward the label with lower index to produce manifold topologies that match standard Marching Cubes cases; this is not done in [2].) The final surfaces inserted into the case table are those faces that separate subcells with different material labels.

Repeating this process for valid binary- and multi-label configurations produces a "discretized" 3D case table for *rule-based marching*. The reader is referred to the work of Hege et al. [2] for additional details, and a full case table for two and three materials – albeit with simplification and non-manifold binary-labeled cases. In this paper, we highlight select cases from our full table on the bottom row of Figure 3.

By carefully constructing our 3D case table for *rule-based marching* to contain discretized interface geometry at the same resolution used by the *discrete boundary* method, the task of merging the output from different topology generation stages in 3D becomes straightforward. As with 2D, we can take the union of meshes from each stage – now without significant re-meshing – and discard faces that create degenerate boundaries. Note that while material interfaces produced from our discretized lookup table (and by the *discrete boundary* method) will be "blocky", all interfaces will be optimized in Section 5.

After creating a single material interface mesh data structure from the results of each topology initialization stage, two bookkeeping tasks remain:

- Every surface mesh face – i.e., segment in 2D, or triangle in 3D – is marked with the two materials that it segments. This makes it simple to determine the orientation of mesh faces and the material regions that they bound.
- We next calculate the per-cell, per-material volume prescribed by the initial surface mesh. The *boundary* method separates only pure cells, and thus no volume calculation is necessary; in the *rule-based marching* method we precompute the per-material volume of each cell segmentation during the lookup table initialization; and in the *discretized boundary* method, summation over the discretized labeling yields the correct volume for each material.

5 Volume-Adaptive Active Interfaces

In this section, we develop a *volume-adaptive* active interface model that iteratively refines the initial interfaces created in Section 3 toward a better reconstruction. The basis of our model is to deform the material interfaces under the influence of local forces. At each iteration step we pick a random control point, and move that point to satisfy our model's objectives better: surface quality and volume accuracy. We continuously update the volume error of the material interfaces, and discuss methods to monitor and enhance convergence.

Active interfaces are represented as piecewise linear curves or surfaces composed of a set of k control points $P = \{p_1, \ldots, p_k\}$ connected by line segments or triangles. Our volume-adaptive model defines two local forces that can be computed per-vertex in the interface mesh. The first is an internal *smoothing force* that attempts to reduce the curvature of the mesh and make control points equidistant; the second is an external *volumetric force* normal to the surface that adjusts the mesh to better fit the volume fraction data. Both forces are computed directly from local interface mesh information, and thus scale with the mesh.

Laplacian smoothing [43] is widely used to approximate internal surface forces within active interface models [28], [29]. The smoothing operation is performed by iteratively replacing a point by the average of its neighboring vertices. Given a sequence of points $p_1, \ldots, p_n$ representing a piecewise linear curve, Laplacian smoothing at point i replaces p_i by the average of p_{i-1} and p_{i+1}. In the case of a surface in three dimensions, p_i is replaced by the average of the points in the 1-ring of point i.

We use Laplacian smoothing to determine the internal force pushing on each point of the interface. Let p_i be a control point defining the piecewise linear interface, then the force is determined to be

$$F_{\text{int}} = d - p_i,$$

where d is the average position of the control points that neighbor p_i.

One difficulty with Laplacian smoothing is that it tends to shrink a curve. Historically, shrinking has been counterbalanced by an external force normal to the curve (see Cohen and Cohen [24], [25]). In a material interface context, however, the shrinking of a curve bounding one material region corresponds to an increase in other materials' volumes. This observation motivates the addition of an external force that does not simply try to increase every bounded region's volume, but instead adaptively pushes the interface toward a better local match to the volume fractions.

Let $\varepsilon(c,A)$ be the signed error between the known and reconstructed fractional volume of material A within the cell c containing control point p_i. If the signed error is positive, then there is too much volume assigned to material A, and surfaces bounding material A within the cell should shrink to reduce the volume. On the other hand, when the error is negative, the bounding surface should grow such that A is awarded more volume within the cell.

To capture this desired behavior, we define the volume-adaptive external force at p_i for material A to be the oriented average of the normals of the faces surrounding p_i:

$$F_{\text{ext}}(A) = \left(\varepsilon(c,A) \sum_{f \in R_A} \perp_A (f) \right),$$

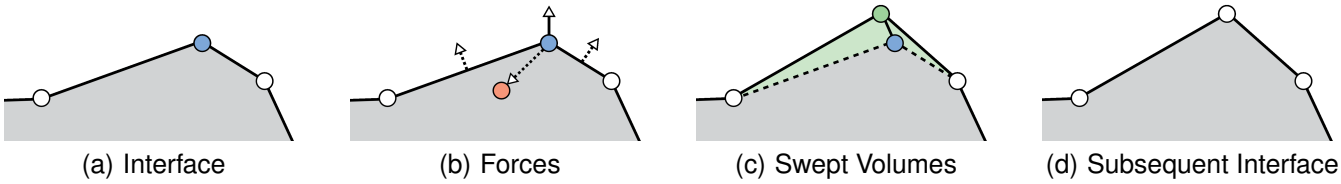


Fig. 5. An interface point is updated within our volume-adaptive active interface model. Consider the blue control point in (a), part of the interface between the empty white region and the shaded grey region. We use local forces – shown in (b) – to update the position of the control point. The *smoothing force* is the offset needed to move the control point to the average of its neighboring control points (shown in orange); the *volume force* is the average of the oriented normals for the faces surrounding the current control point. After moving the control point in (c), we update the volume of each material region by analyzing the area swept by the control point (shaded green). As this update step is repeated for other points on the interface the reconstruction improves.

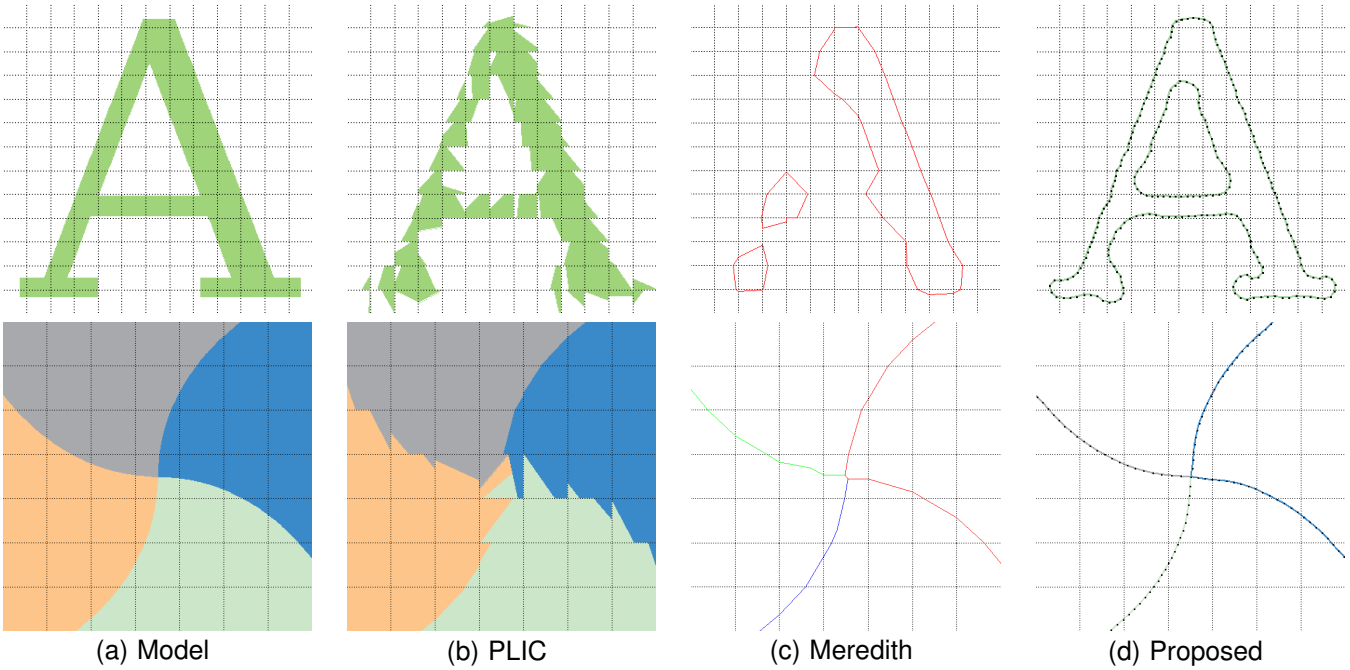


Fig. 6. Two 2-dimensional volume fraction problems: the letter "A", and a four-material junction between intersecting curved interfaces. We show the model interfaces in (a) from which we calculate volume fractions for reconstruction. The PLIC reconstruction of these problems is shown in (b), an isosurfacing-like reconstruction using the method by Meredith is shown in (c), and our proposed reconstruction method in (d).

where R_A is the set of faces surrounding p_i that are marked with material A in cell c, and $\perp_A(f)$ is the normal of face f oriented into material region A. Multiplying by $\varepsilon(c,A)$ adaptively orients the external force to shrink or grow the region bounding material A as needed to better match the problem's volume fractions.

We now combine these forces – illustrated in Figure 5(b) – to update the positions of points that define the piecewise linear material interfaces. Consider a control point p_i within the mixed cell c. At p_i there is a single smoothing force, but there will be multiple volumetric forces: two if the control point's neighborhood is a topological disk separating two materials; more if the neighborhood is multi-material junction. We address this issue stochastically in our active contour model by only considering a single material's volumetric force per iteration.

For a randomly chosen material ξ, the total local force at p_i becomes:

$$F = \alpha F_{\text{int}} + \beta F_{\text{ext}}(\xi),$$

where α is the weighting of the internal force within the active interface model, and β is the weighting of the external force. Finally, we update the position of the control point:

$$p_i' = p_i + F.$$

After moving a control point it is necessary to update the current error of the reconstruction ε in order to maintain the accuracy of the volume-adaptive force. We calculate the volume swept by the 1-ring of p_i as it moves to p_i': in two dimensions line segments are swept to form triangles as shown in Figure 5(c); triangles are swept into tetrahedra in three dimensions. The swept triangles or tetrahedra are then clipped against mesh cells in the local neighborhood, and the oriented volume of the clipped triangles or tetrahedra is used to update the current error of the reconstruction. Incrementally

maintaining material volumes is much faster than recomputing the entire volume each update.

To obtain useful results with any active interface model, the forces within the system must be controlled. In our implementation, a simple yet effective rule is to set the weight of the internal force proportional to that of the external force

$$\alpha \propto \beta,$$

and then monotonically decrease α and β at a rate relative to the change in error until the material interfaces converge. Managing the weights in this manner is analogous to gradually reducing the temperature parameter of simulated annealing [44].

Finally, we note that in order to obtain both volume-accuracy *and* smoothness it can help to subdivide large faces on the boundary surface. A threshold parameter σ is introduced to control the maximum surface face size – i.e., maximum line segment length or triangle area. Using a very large value for σ will ensure that subdivision is not used in cases where coarser surfaces are desired.

6 Results

This section evaluates our method – and compares it to existing methods – over multiple volume fraction datasets. We have generated synthetic data in two and three dimensions to compare against model reconstruction solutions. In three dimensions we present two real-world volume fraction examples. The first example is from a CFD simulation using the Volume-of-Fluid (VOF) method. The second is from an Embedded Boundary (EB) problem. In EB problems, a fixed boundary surface is represented using volume fractions – rather than geometry – in order to facilitate multi-physics simulation around the boundary. Results were obtained on an Apple MacBook Pro notebook computer with a 2.33 GHz Intel Core 2 Duo processor and 2 GB of memory.

TABLE 1
Problem sizes, surface complexity, and reconstruction times.

	Materials	Extents	Mixed	Faces	Time (m:ss)
A	2	20^2	17.5%	242	0:10
Junction	4	7^2	42.86%	130	0:13
3D Bubble	2	64^3	2.65%	143,640	3:42
SF Bay	2	$258^2 \times 10$	1.99%	117,610	5:17
Swirler	2	64^3	6.13%	356,904	4:24
Sphere/Box	3	13^3	15.66%	15,014	3:02
Spheres	6	13^3	58.03%	73,392	2:56

As a reference, Table 1 provides summary information about the datasets and our reconstructions, including: number of materials, dataset size, percent of mixed cells, number of faces in the reconstructed interface, and convergence time. Table 2 plots on a logarithmic scale the average reconstruction error of our method, the isosurfacing-like reconstruction of Meredith [20] (as implemented in VisIt [12]), and the discrete method of Anderson et al. [21]. Error is computed as the average maximal volume fraction differential between a reconstruction and the problem's known volume fractions:

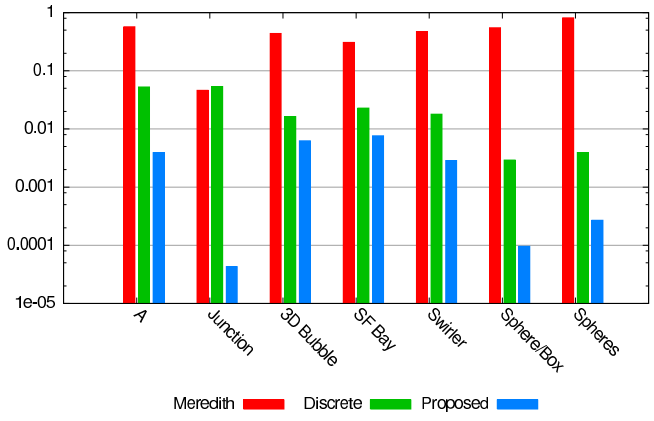
$$E = \frac{\sum_c \max_{i\in(1,\ldots,n)} |\varepsilon(c,i)|}{|\text{Mixed Cells}|}.$$

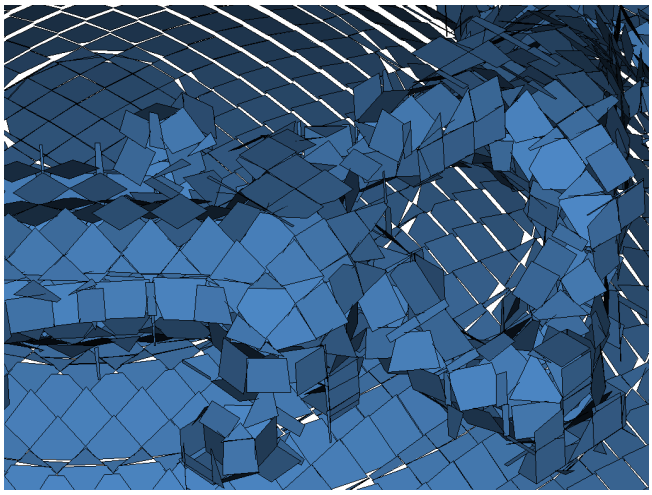
For example, an error of 0.01 indicates that the expected maximum misclassification of material in each mixed cell is 1% of the cell's volume.

Our first set of tests are over two-dimensional synthetic volume fraction data for which we have a model solution. We compute the volume fractions from the model images in 6(a), and attempt to reconstruct the boundaries using PLIC [14], the method of Meredith [20], and our proposed reconstruction method. The top row in Figure 6 shows reconstructions of a letter "A" in serif font embedded within a 20^2 cell grid. Our topology generation pipeline correctly captures the thin filament structures, and our volume-adaptive active interface model iterates the boundary to an average mixed cell error of only 0.3924%. The bottom row of Figure 6 shows reconstructions of a problem in which multiple curved interfaces intersect at a 4-material "junction." This problem also requires the generation of non-trivial topology. PLIC does a poor job of reconstructing this interface because the correct topology cannot be represented by manifold, binary segmenting surfaces. The *rule-based marching* method is able to extract the curved 2-material interfaces, although the center cell containing four materials remains uninitialized under that method. Running the *discretized boundary* method upon the center cell produced the correct topology. Another valid topology that is occasionally generated by the *discretized boundary* method has two 3-material junctions, similar to the reconstruction produced by the method of Meredith [20]. Our final reconstruction has an average mixed cell error of 0.0043%.

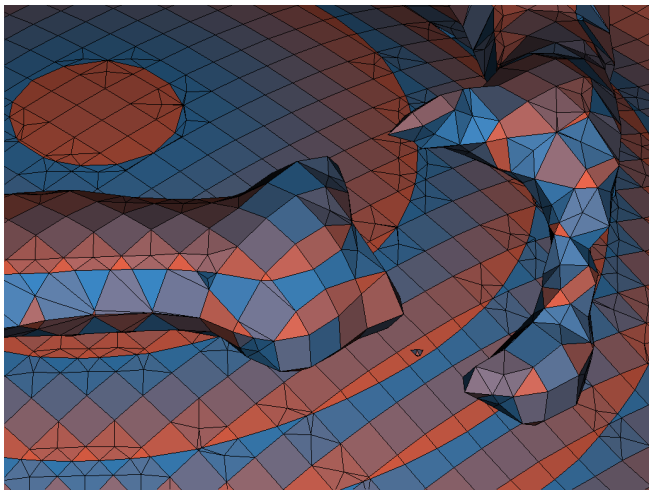
Next, we consider a three-dimensional Volume-of-Fluid simulation of a low density bubble rising through a denser fluid. The computational domain was 64^3. After the bubble reaches the surface, it bursts as a result of surface tension. Figure 7 provides a closeup view of this dataset after the bubble has burst: in 7(a) we show a 3D version of the PLIC algorithm, the result of which is a set of disconnected polygons that partition each mixed cell into two material regions; in 7(b) we show the reconstruction by Meredith [20]; and in 7(c) we show the surface mesh generated by our proposed reconstruction. Pseudocolor has been used to visualize the local, cell-level error on reconstructed interfaces in 3D problems. A

TABLE 2
Log plot of average per-cell error for various methods.

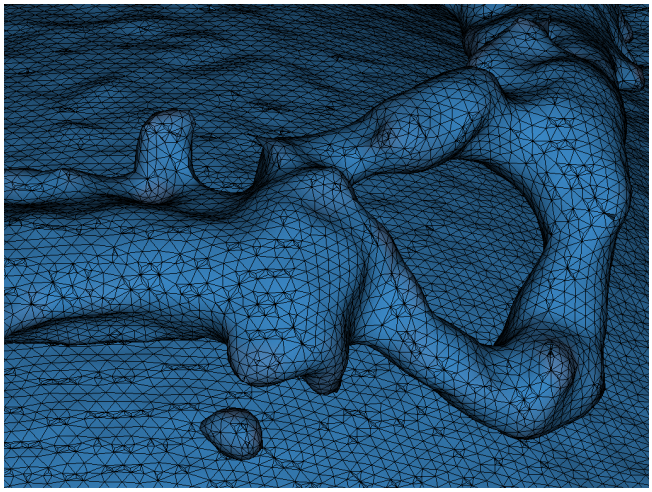




(a) PLIC



(b) Meredith



(c) Proposed

Fig. 7. Reconstructions in three-dimensions of a low-density bubble of fluid rising through a higher density fluid after the bubble has burst through the surface. In (a), we show the PLIC interface reconstruction; (b) shows an isosurfacing-like reconstruction using Meredith's method; (c) is our proposed reconstruction. Interfaces have been pseudocolored by the per-cell reconstruction error.

blue-to-red colormap is used for the range of $[0.0, 1.0]$ volume error: blue indicates low error with little misclassification of material, while red indicates higher error.

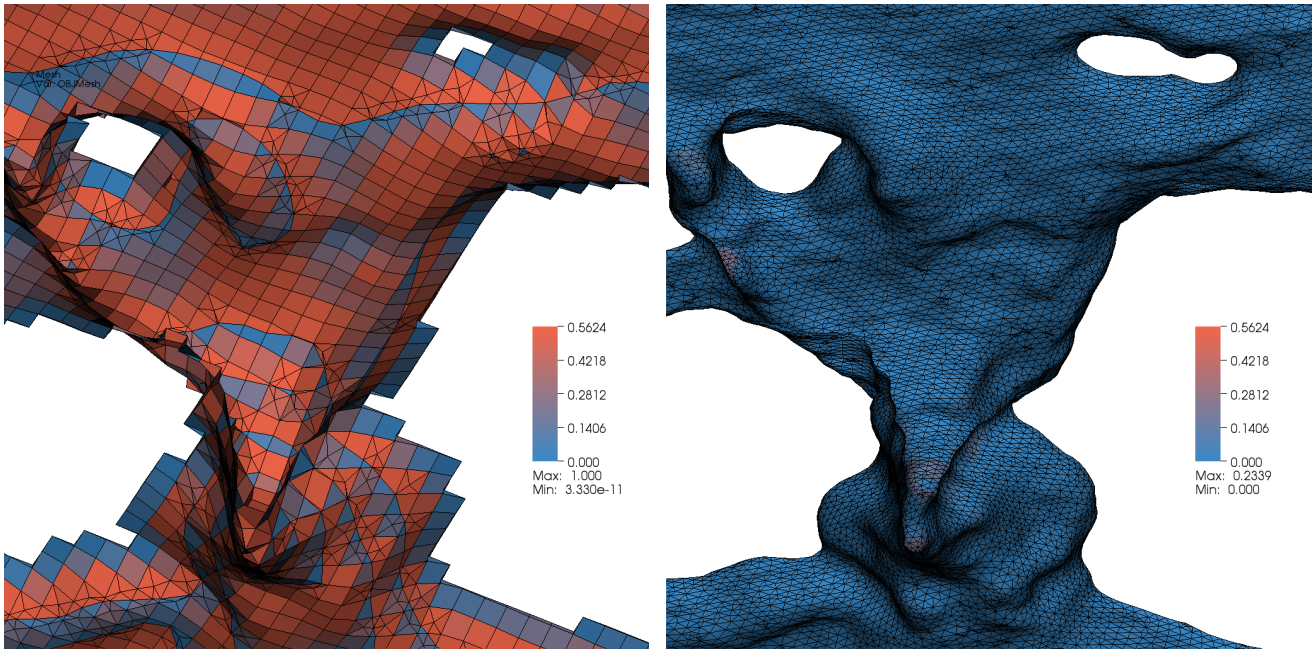
We now turn to a three-dimensional embedded boundary representation of bathymetry – or underwater depth – for the San Francisco Bay. This type of data can be used in a wide range of simulations, from ocean currents and tidal flow to modeling oil spills. The dataset is a $258^2 \times 10$ rectilinear grid with one explicit volume fraction, representing two materials: above and below the boundary. Figure 8(a) shows the reconstruction produced by the Meredith's algorithm [20], while our proposed reconstruction is shown in 8(b). The colormap in this figure is also blue-to-red and indicative of volume error, but the range is $[0.0, 0.5624]$. Our reconstruction produces smoother interfaces with much lower average per-cell error: 0.7640% per mixed cell for our method, versus 43.8410% for the method by Meredith [20].

Next, we consider reconstructing a "low-swirl burner." Low-swirl combustion is an aerodynamic flame stabilization method that produces ultra-lean flames with low emission, and is largely used for industrial heating and gas turbines [45]. The embedded boundary dataset we use is a 128^3 rectilinear grid of volume fractions derived from the constructive solid geometry definition of a low-swirl burner. Our reconstructions focus upon the lower 64^3 octant of the full dataset due to the symmetric nature of the geometry. Figure 9 compares the reconstruction by Meredith's method (left) to our proposed reconstruction (right); notice that our method fully captures the swirler blades and central screen. In our reconstruction, the *rule-based marching* method generates interface topology for 71% of the mixed cells (mostly on the large cylindrical portions of the burner), while the *discrete boundary* method generated interfaces for the swirler blades and central screen.

Finally, constructive solid geometry has been used to create analytic test datasets in three-dimensions. Two datasets are shown in the left column of Figure 10: on top, a box intersected by a sphere (box: $(0.2, 0.2, 0.2)$ to $(0.6, 0.6, 0.6)$; sphere: center $(0.6, 0.6, 0.6)$, radius 0.2); on bottom, five concentric spheres (centers $(0.5, 0.5, 0.5)$, radii $\frac{1}{13}$, $\frac{2.25}{13}$, $\frac{3.5}{13}$, $\frac{4.75}{13}$, and $\frac{6}{13}$). The sphere/box problem has 3 materials, while the multiple spheres problem has 6 materials – in both problems, "empty" space is another material. In the middle column of Figure 10 we show the reconstructions performed using the algorithm of Meredith [20], while our proposed reconstructions are shown in the right column. Error results are listed in Table 2, however the figure also includes horizontal lines to help illustrate differences between the reconstructions and the problem models.

7 Conclusion

We have described a new material interface reconstruction method that produces high-quality boundary meshes with low error in two and three dimensions. Our approach separates interface topology generation from iterative surface improvement using a volume-adaptive active interface model. Experimental results show our approach to be very well-behaved: per-cell error tends to be significantly less than 1%, while producing continuous, piecewise linear meshes.



(a) Meredith (b) Proposed

Fig. 8. Reconstructions of an embedded boundary (EB) representation of bathymetry – or underwater depth – data from the San Francisco Bay using: (a) Meredith's method, and (b) our active interface reconstruction method. Interfaces have been pseudocolored by the per-cell reconstruction error.

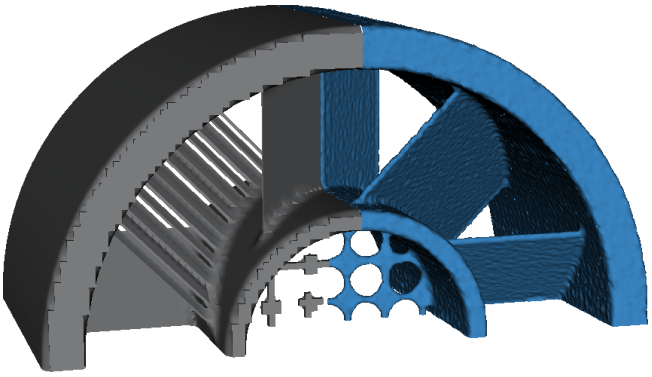


Fig. 9. Comparison of Meredith's reconstruction (left) to our proposed reconstruction (right) for one quadrant of the swirler dataset.

Numerous directions for future work remain:

- Develop methods to generate a parameterized range of topology solutions. For example, users should be able to control – on a per-material basis – whether they want interfaces that tend toward thin filaments/shells or multiple disconnected blobs.
- Explore the use of level set methods [30] as an alternative to our current mesh-based active interface model, especially in the presence of parameterized topology control.
- Support unstructured meshes and adaptive mesh refinement (AMR) grids. In our proposed framework, this would require the extension of our topology generation pipeline to support different mesh types; the volume-adaptive active interface model presented is largely independent of the underlying mesh.
- In this work we have focused on generating high-quality material interfaces (in terms of geometric quality and low volumetric error), rather than running time. It is not uncommon for volume fraction datasets to contain billions of elements; additional work should focus upon the speed of our algorithm through parallelism, simpler surface meshes, and possibly multi-resolution methods.
- Volume fraction data is often dumped on a coarse timescale from fine-grained simulations. Interface tracking at the visualization time scale should be investigated to encourage timestep-to-timestep topology consistency when possible.
- Parameter optimization remains an open topic; the following parameters control our algorithm:
  - D is the discretization parameter (for the *discrete boundary* method, and 3D case table construction);
  - α and β are local smoothing and volumetric weights in the active interface model, respectively; and,
  - σ is a size threshold parameter.

  We are continuing to investigate optimal methods to set and tune these parameters based upon the problem's dimension, number of materials, and interface complexity. As a general guide, however, we have found that setting $D = 5$, $\sigma = 0$, and $\alpha = 1.5\beta$ (decreased during run-time) to be a good starting point for these parameters.
- The "discretized" case table discussed in Section 4 dramatically eases the implementation of our method. However, in the future we plan to implement and test re-meshing capabilities that would merge interfaces produced by existing multi-material MIRLV algorithms with our *discrete boundary* output.

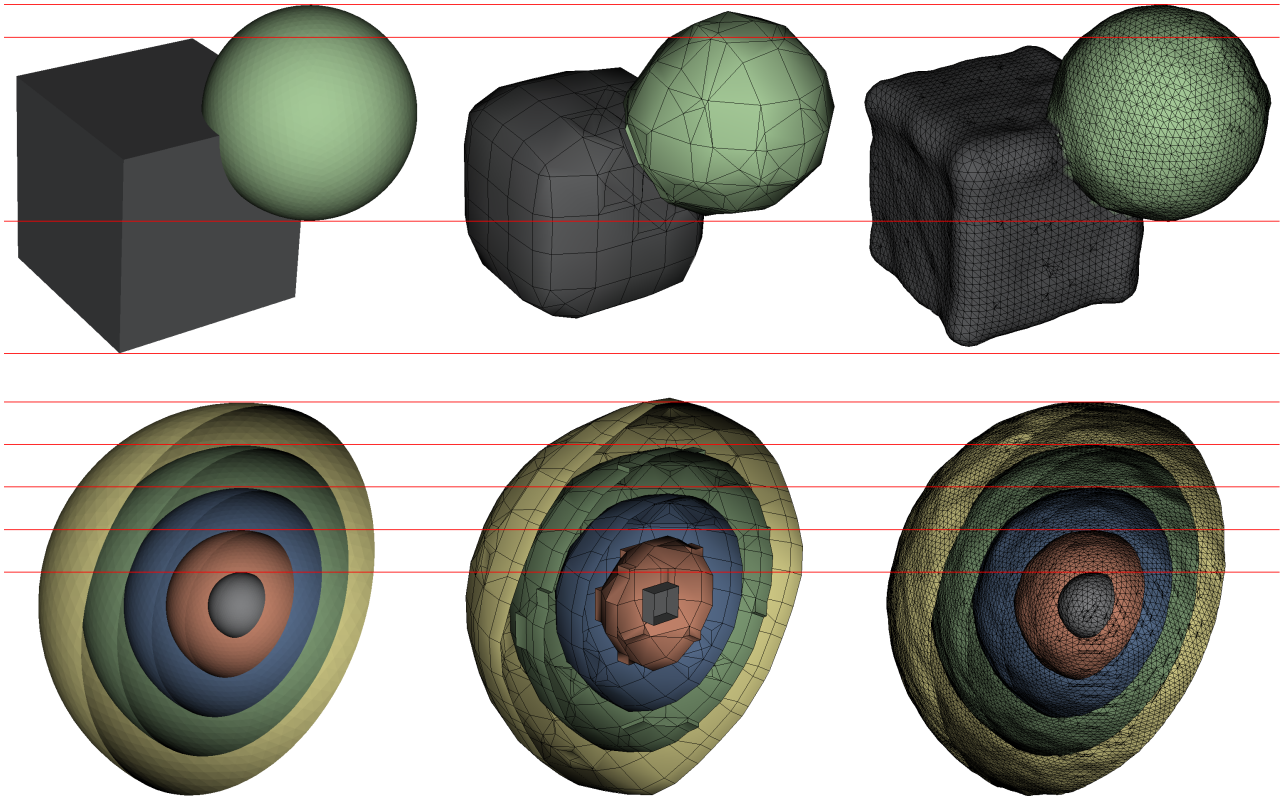


Fig. 10. Analytic three-dimensional problems with multiple materials: on top, a box intersected by a sphere (3 materials); on bottom, five concentric spheres (6 materials). Horizontal lines help to illustrate the differences between reconstructions; note the volume loss with an isosurfacing-like approach compared to our proposed method.

- Finally, we note that Laplacian smoothing is only one possible choice for the internal surface force in an active interface model. Our choice of Laplacian smoothing reflects our goal to promote smoothness and general surface quality [43]; other force models could be explored to promote or preserve sharp corners while smoothing.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and supported by the Lawrence Scholar Program. This work was also supported by the Office of Science, U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET).

The swirler dataset was graciously provided by Terry J. Ligocki and Phillip Colella at Lawrence Berkeley National Laboratory; more information about low-swirl combustion is available at http://eetd.lbl.gov/aet/combustion/LSC-info/. The bubble dataset was generated using the open-source fluid solver Gerris [46], available at http://gfs.sf.net. We have also used VisIt [12] (http://www.llnl.gov/visit), a free interactive parallel visualization and graphical analysis tool, to produce certain visualizations.

The authors would additionally like to thank Jeremey Meredith from Oak Ridge National Laboratory, and colleagues in the Institute for Data Analysis and Visualization (IDAV) at UC Davis.

REFERENCES

[1] W. E. Lorensen and H. E. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm," in *Proc. of SIGGRAPH*, 1987, pp. 163–169.

[2] H.-C. Hege, M. Seebaß, D. Stalling, and M. Zöckler, "A generalized Marching Cubes algorithm based on non-binary classifications," Konrad-Zuse-Zentrum für Informationstechnik Berlin, Tech. Rep. SC-97-05, 1997.

[3] Z. Wu and J. M. Sullivan, Jr., "Multiple material Marching Cubes algorithm," *International Journal for Numerical Methods in Engineering*, vol. 58, no. 2, pp. 189–207, Jul. 2003.

[4] D. C. Banks and S. Linton, "Counting cases in Marching Cubes: Toward a generic algorithm for producing substitopes," in *Proc. of IEEE Visualization*, Oct. 2003, pp. 51–58.

[5] G. M. Nielson and R. Franke, "Computing the separating surface for segmented data," in *Proc. of IEEE Visualization*, Oct. 1997, pp. 229–233.

[6] G. M. Nielson and J. Sung, "Interval volume tetrahedrization," in *Proc. of IEEE Visualization*, 1997, pp. 221–228.

[7] M. Meyer, R. Whitaker, R. M. Kirby, C. Ledergerber, and H. Pfister, "Particle-based sampling and meshing of surfaces in multimaterial volumes," *IEEE Trans. on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1539–1546, 2008.

[8] C. Hirt and B. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *J. Computational Physics*, vol. 39, pp. 201–225, 1981.

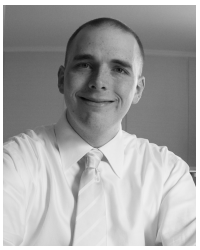
[9] W. J. Rider and D. B. Kothe, "Reconstructing volume tracking," *J. Computational Physics*, vol. 141, no. 2, pp. 112–152, 1998.

[10] D. J. Benson, "Volume of fluid interface reconstruction methods for multi-material problems," *Applied Mechanics Reviews*, vol. 55, no. 2, pp. 151–165, Mar. 2002.

[11] J. E. Pilliod, Jr. and E. G. Puckett, "Second-order accurate volume-of-fluid algorithms for tracking material interfaces," *J. Computational Physics*, vol. 199, no. 2, pp. 465–502, 2004.

[12] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max, "A contract-based system for large data visualization," in *Proc. of IEEE Visualization*, 2005, pp. 190–198.

[13] W. F. Noh and P. Woodward, "SLIC (Simple Line Interface Calculation)," in *LNP Vol. 59: Some Methods of Resolution of Free Surface Problems*, A. I. van de Vooren and P. J. Zandbergen, Eds., 1976, pp. 330–340.
[14] D. L. Youngs, "Time-dependent multi-material flow with large fluid distortion," in *Numerical Methods for Fluid Dynamics*. Academic Press, 1982, pp. 273–285.
[15] R. V. Garimella, V. Dyadechko, B. K. Swartz, and M. J. Shashkov, "Interface reconstruction in multi-fluid, multi-phase flow simulations," in *Proc. of International Meshing Roundtable*, Sep. 2005, pp. 19–32.
[16] V. Dyadechko and M. Shashkov, "Reconstruction of multi-material interfaces from moment data," *J. of Computational Physics*, vol. 227, no. 11, pp. 5361–5384, May 2008.
[17] S. P. Schofield, R. V. Garimella, M. M. Francois, and R. Loubre, "Material order-independent interface reconstruction using power diagrams," *International Journal for Numerical Methods in Fluids*, vol. 56, no. 6, pp. 643–659, 2008.
[18] K. S. Bonnell, D. A. Schikore, M. A. Duchaineau, B. Hamann, and K. I. Joy, "Constructing material interfaces from data sets with volume-fraction information," in *Proc. of IEEE Visualization*, Oct. 2000, pp. 367–372.
[19] K. S. Bonnell, M. A. Duchaineau, D. R. Schikore, B. Hamann, and K. I. Joy, "Material interface reconstruction," *IEEE Trans. on Visualization and Computer Graphics*, vol. 9, no. 4, pp. 500–511, 2003.
[20] J. S. Meredith, "Material interface reconstruction in VisIt," Lawrence Livermore National Laboratory, Tech. Rep. UCRL-CONF-209351, 2004.
[21] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy, "Discrete multi-material interface reconstruction for volume fraction data," *Computer Graphics Forum*, vol. 27, no. 3, pp. 1015–1022, May 2008.
[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes – active contour models," *International J. of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.
[23] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey," *Medical Image Analysis*, vol. 1, pp. 91–108, Jun. 1996.
[24] L. D. Cohen, "On active contour models and balloons," *Computer Vision, Graphics, and Image Processing*, vol. 53, no. 2, pp. 211–218, 1991.
[25] L. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1131–1147, Nov. 1993.
[26] I. Takanashi, S. Muraki, A. Doi, and A. Kaufman, "3D active net: 3D volume extraction," *Institute of Image Information and Television Engineers*, vol. 51, no. 12, pp. 2097–2106, 1997.
[27] ——, "Three-dimensional active net for volume extraction," in *Proc. of SPIE*, vol. 3298, 1998, pp. 184–193.
[28] S. F. F. Gibson, "Constrained elastic surface nets: Generating smooth surfaces from binary segmented data," *Lecture Notes in Computer Science*, vol. 1496, pp. 888–900, 1998.
[29] J. Ahlberg, "Active contours in three dimensions," Master's thesis, Linköping University, Sep. 1996, LiTH-ISY-EX-1708.
[30] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. of Computational Physics*, vol. 79, pp. 12–49, 1988.
[31] D. Adalsteinsson and J. Sethian, "A fast level set method for propagating interfaces," *J. of Computational Physics*, vol. 118, no. 2, pp. 269–277, 1995.
[32] J. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
[33] J. A. Sethian, "Evolution, implementation, and application of level set and fast marching methods for advancing fronts," *J. Computational Physics*, vol. 169, no. 2, pp. 503–555, 2001.
[34] S. Osher and R. Fedkiw, "Level set methods: An overview and some recent results," *J. Computational Physics*, vol. 169, pp. 463–502, 2001.
[35] ——, *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
[36] B. Merriman, J. K. Bence, and S. J. Osher, "Motion of multiple junctions: a level set approach," *J. Computational Physics*, vol. 112, no. 2, pp. 334–363, 1994.
[37] S. J. Ruuth, "A diffusion-generated approach to multiphase motion," *J. Computational Physics*, vol. 145, no. 1, pp. 166–192, 1998.
[38] K. A. Smith, F. J. Souls, and D. L. Chopp, "A projection method for motion of triple junctions by level sets," *Interfaces and Free Boundaries*, vol. 4, no. 3, pp. 263–276, 2002.
[39] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," *ACM Trans. on Graphics*, vol. 25, no. 3, pp. 812–819, 2006.
[40] L. A. Vese, Tony, and F. Chan, "A multiphase level set framework for image segmentation using the Mumford and Shah model," *International J. of Computer Vision*, vol. 50, pp. 271–293, 2002.
[41] G. M. Nielson and B. Hamann, "The asymptotic decider: resolving the ambiguity in Marching Cubes," in *IEEE Visualization*, 1991, pp. 83–91.
[42] R. V. Garimella, "Mesh data structure selection for mesh generation and FEA applications," *International J. for Numerical Methods in Engineering*, vol. 55, no. 4, pp. 451–478, 2002.
[43] D. A. Field, "Laplacian smoothing and Delaunay triangulations," *Communications in Applied Numerical Methods*, vol. 4, no. 6, pp. 709–712, 1988.
[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, 4598, no. 4598, pp. 671–680, May 1983.
[45] R. K. Cheng, "Low swirl combustion," in *Gas Turbine Handbook*. U.S. Department of Energy, 2006.
[46] S. Popinet, "Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries," *J. Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.



John C. Anderson received a B.S. in Computer Science and B.A. in History from University of the Pacific in 2004, and a Ph.D. in Computer Science from UC Davis in 2009. While at UC Davis, he collaborated with researchers from Lawrence Livermore National Laboratory on surface extraction, multi-variate analysis, and query-driven visualization methods. Dr. Anderson now works on visualization problems in the earth sciences at Makai Ocean Engineering, Inc. in Waimanalo, Hawaii.

Christoph Garth received the Diplom (M.Sc.) in Mathematics and Computer Science and Dr. rer. nat. (Ph.D.) in Computer Science from the University of Kaiserslautern in 2003 and 2007, respectively. Dr. Garth is currently a postdoctoral researcher with the Institute for Data Analysis and Visualization at UC Davis. His research focuses on scientific visualization, analysis methods for vector and tensor fields, topological methods, query-driven visualization, parallel/scalable algorithms, and computer graphics.

Mark A. Duchaineau is a research scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. He received both an M.S. (1996) and Ph.D. (1996) in Computer Science from UC Davis. Dr. Duchaineau's recent research foci include the visualization of terrascale scientific simulations, multi-resolution and multi-variate methods, and large-scale 3D reconstruction of aerial imagery.

Kenneth I. Joy is a Professor in the Department of Computer Science at UC Davis, and the Director of the Institute for Data Analysis and Visualization. In this position, Prof. Joy leads research efforts in scientific visualization, geometric modeling, and computer graphics. He is a faculty computer scientist at Lawrence Berkeley National Laboratory and participating guest researcher at Lawrence Livermore National Laboratory. Prof. Joy received a B.A. (1968) and M.A. (1972) in Mathematics from UCLA, and a Ph.D. (1976) from the University of Colorado, Boulder.