Lawrence Berkeley National Laboratory
Lawrence Berkeley National Laboratory

Title
Visualization and Analysis-Oriented Reconstruction of Material Interfaces

Permalink
https://escholarship.org/uc/item/2nc2z8jh

Author
Childs, Henry R.

Publication Date
2010-06-01

Visualization and Analysis-Oriented Reconstruction of Material Interfaces

Jeremy S. Meredith[1] and Hank Childs[2]

[1]Oak Ridge National Laboratory, USA
[2]Lawrence Berkeley National Laboratory and University of California, Davis, USA

Abstract

Reconstructing boundaries along material interfaces from volume fractions is a difficult problem, especially because the under-resolved nature of the input data allows for many correct interpretations. Worse, algorithms widely accepted as appropriate for simulation are inappropriate for visualization. In this paper, we describe a new algorithm that is specifically intended for reconstructing material interfaces for visualization and analysis requirements. The algorithm performs well with respect to memory footprint and execution time, has desirable properties in various accuracy metrics, and also produces smooth surfaces with few artifacts, even when faced with more than two materials per cell.

Categories and Subject Descriptors: I.3.6 [Computer Graphics] Methodology and Technologies

1   Introduction

Many important classes of computer simulations of physical phenomena require support for "materials," i.e. discrete regions of space with different physical properties. For example, a simulation of tidal waves needs to partition space into water and air, and a simulation of an automobile accident must model glass, metal, and rubber. There are two approaches to supporting materials on a computational mesh: Lagrangian (where each cell contains exactly one material for the entire simulation) and Eulerian (where the materials are allowed to flow through the mesh). Although the Lagrangian approach is simpler to implement, the Eulerian approach is often used because of its flexibility. The Eulerian approach is ideal for computations requiring a static mesh while materials move, for materials that bend and twist so significantly that they can't be represented easily with normal mesh elements, or simply to model materials at a higher resolution than the mesh to maintain accuracy. The result is that cells in the computational mesh will be "mixed," i.e. containing two or more materials.
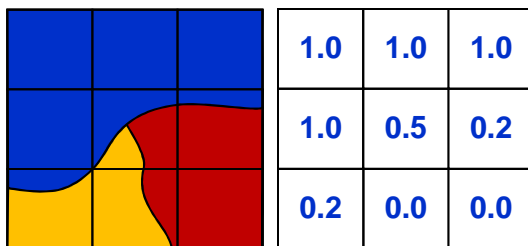

Figure 1: *Original problem materials (left) and the volume fractions of the blue material in each cell (right).*

Material information is stored on a per-cell basis, with a scalar for each material representing the "volume fraction" (VF) for that material, i.e. the percent of the cell occupied by that material. Figure 1 shows an example of three materials in two-dimensional space along with a nine-cell computational grid, as well as the VFs in each cell for the upper-most (blue) material. In this example, the four cells with a blue VF of 1.0 are "clean," containing only the blue material, and the three cells with a VF between 0 and 1 are mixed and should contain the interface of the blue material.

There are multiple "correct" solutions and many criteria for a good reconstruction: Does it honor the volume fractions? Does it place materials from neighboring cells next to each other? Does it create large discontinuities? Although simulations reconstruct interfaces themselves, their primary concerns include advecting materials through the mesh correctly or specific physical properties like conservation of mass, not visualization and analysis. Their reconstructions often lead to inaccurate analysis and poor aesthetics. In this paper, we introduce a new algorithm that is well suited for visualization and analysis.

The rest of the paper is organized as follows. In section 2, we describe previous work in this area. Section 3 describes our new approach and variations therein. In section 4, we perform a comparative evaluation with several other types of interface reconstruction techniques, and we present paths for future work and our conclusions in section 5.

2   Previous work

One of the first techniques for material interface reconstruction (MIR) is a method that uses tracking particles to define the interface [Ams66]. Later methods tracked interfaces using level set methods [OS88]. An early method for creating linear geometric material boundaries out of volume fractions arose in the simple line interface calculation (SLIC) [NW76] and volume-of-fluid (VOF) method [HN81]. SLIC and VOF are piecewise-constant/stair-stepped algorithms, aligning material interface boundaries with one of the major coordinate axes. An improvement to these approaches came in the piecewise linear interface calculation (PLIC) [PY92], which loops over the materials in each cell, first choosing an orientation then finding the intersection position which results in the

correct VF. By supporting non-axis aligned orientations, PLIC removed one of the major obstacles to a more realistic reconstruction. Modifications to the basic PLIC algorithm include the ordering of materials and method for calculating interface orientations [AS07]. Again, these methods are largely concerned with not merely the reconstruction of the material interfaces but also the roles these interfaces play in simulation codes, such as tracking and propagating the surface following physical laws, and thus often neglect features important for visualization and analysis.

Other algorithms have been devised which are more applicable for analysis and visualization. One algorithm implemented in visualization programs is an isosurface algorithm, which creates surfaces based on interpolated volume fraction values, either on re-centered values or on a dual mesh. [RBW99] This approach generates smooth surfaces, but has serious flaws in supporting multiple materials. Bonnell et al. describe an approach which remaps geometry into barycentric coordinate space, leading to more accurate reconstructions for more than two materials [BJH*00]. However, when faced with additional materials, the output has a high geometric complexity and the algorithm only applies to triangle/tetrahedral grids. The algorithm described by [AGD*08] subdivides cells into a discretized grid and iteratively shifts material sub-voxels to minimize an energy function, generating good results with fixed error bounds. However, the subdivision is expensive in output complexity and performance time. Further, the output is restricted to generating axis-aligned boundary surfaces. Another category of approaches could be described as multi-material marching cubes variants [WS03, BL03, HSS*97], but these approaches add too many constraints, such as requiring material choices to be made at each corner, or supporting intersections only at edge midpoints. Finally, the algorithm described in this paper represents a significant extension to and analysis of the algorithm described in [Mer04], in particular adding an iterative step to improve accuracy and an analysis comparing its effectiveness to other algorithms.

3 Algorithm description
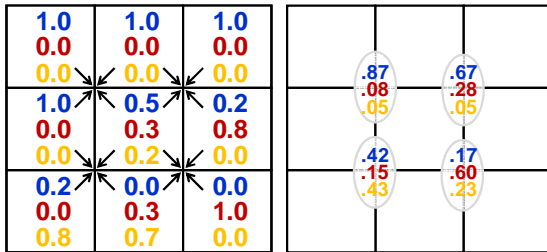
3.1 Core algorithm

Figure 2: *Volume fractions for each material (left) are averaged to the vertices of the mesh (right).*

The algorithm begins by averaging the material VFs to the mesh vertices. This ultimately enables continuity since the algorithm depends on volume fractions along cell edges, and each edge will now have the same starting values, regardless of which adjacent cell we are reconstructing. We illustrate each step of our algorithm with an example where we study the reconstruction on the central cell of a nine-cell grid with three materials: blue, red, and yellow. The first step, seen in Figure 2, demonstrates the averaging step.

Our algorithm deals with multiple materials in a cell by adding each material one at a time. Each cell is initialized with a single material, then, when adding each successive material, it splits out portions of the cell to belong to the new material. Figure 3 shows the first iteration in this process. The center cell has been initialized to be yellow and we demonstrate how a second material (red) is added. This is done in two steps: evaluation and reconstruction. For the evaluation step, we focus on the two selected materials (red and yellow) and ignore all other materials (blue). For each edge in the cell, we test to see if there is a location along the edge where the interpolated VFs of the materials are equal. In our example, the red and yellow materials' interpolated VFs are equal near the top of the left edge, where both have a value of approximately 0.087, and near the middle of the bottom edge, where both have a value of approximately 0.336.
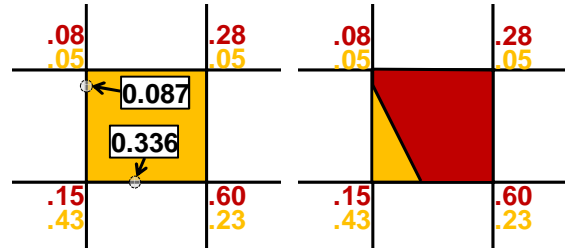

Figure 3: *Intersection points for red/yellow evaluation (left) and the clipped volumes (right).*

If an edge contains an intersection point, we split this edge into two new edges: yellow (for the new edge incident to the yellow-dominant vertex) and red (for the new edge incident to the red-dominant vertex). After the edge splits, we perform the actual reconstruction. In two dimensions, the output is made up of polygons, each containing a single material (red or yellow). The vertices of a yellow triangle, for example, will consist of only yellow-dominant vertices or intersection points. We have implemented two variants, one of which operates only on triangles and requires the input to be triangulated, and one which can input and output both triangles and quadrilaterals. The rules for reconstruction are table-based and resemble a Marching cubes table for isosurfacing [LC87]. Further, when considering only two materials (as we have so far) the resemblance to isosurfacing extends to the entire algorithm. This includes ambiguities that may arise for cases where dominant nodes are diagonal from each other (e.g. red-dominant upper left and lower right, yellow-dominant upper right and lower left). We handle these ambiguities by choosing one material to span the middle and the other to get separated, again much like an isosurface algorithm.

3.1.1 Three dimensional extension

Our three dimensional case is similarly done by table, although we have again implemented two variants. In the first variant, we must tetrahedralize the input and then group output vertices of the same material together, along

with neighboring intersection points, into one or more tetrahedrons. In the second variant, we leave the input cells whole and strive to reduce the number of output cells by grouping tetrahedrons together into pyramids, wedges, and hexahedrons when possible. We refer to this as the "zoo" variant since it uses the elements of the finite element zoo. Once again like a Marching Cubes algorithm, we have worked out all 2^n cases for each n-node primitive so that a simple table lookup determines the output shapes for each reconstruction step. To illustrate the difference between the tetrahedron and "zoo" variants, consider the case of a hexahedron that has four yellow dominant nodes in its top half and four red dominant nodes in its bottom half. The tetrahedral variant would need ten or more tetrahedrons to represent the reconstruction, while the zoo variant would do it with two hexahedrons. This fact has implications for surface smoothness we explore later. Note that three-dimensional reconstruction introduces a new problem: as the faces of the original cell are being divided into triangles and quadrilaterals, it is important that the neighboring cell makes matching decisions regarding how it divides the same face. Otherwise, the connectivity of the output mesh will be incorrect. This potential problem is prevented through careful selection of the table entries, as well as some consistent indexing.

3.1.2 Isosurfaces and Equisurfaces

For a two-material problem, the boundary between the reconstructed materials is indeed an isosurface of value 0.5. However, this boundary is not technically an isosurface. An isosurface is a surface of constant value. Our surface is rather a surface where the volume fractions of two materials are equal. When there are more than two materials, the value along the boundary will vary. Figure 3 demonstrates this, as the boundary has interpolated VFs ranging from 0.087 to 0.336. In recognition of this key difference from an isosurface, we refer to this surface as an "equi-surface."

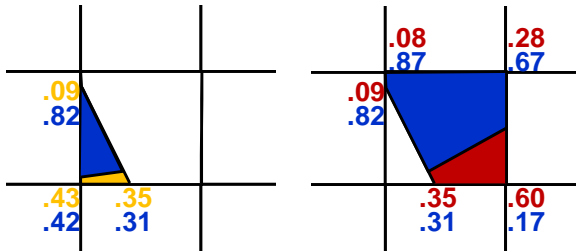### 3.1.3 Reincorporation of Additional Materials

Figure 4: *Evaluating the third (blue) material against the first (yellow) and second (red) materials.*

At this stage, we now have two output cells, one for each material visited. We take each of these cells and evaluate it against the next material (the blue one) as seen in Figure 4. To do so, note that we must have values at all nodes for the blue material as well, and as such must also interpolate VF values for our blue material to the new node locations. This is where the process repeats: we clip our yellow cell from the previous step into a blue and a yellow cell, and the red cell into a blue and a red cell. As mentioned, we have two

variants of this clipping process: one which outputs two triangles for the red material in Figure 4, for example, and the other which outputs a single quadrilateral.

Note that by construction, the edges of these two new blue material cells must meet each other at the exact same points. This same phenomenon occurs at edges between cells, and for this reason this algorithm guarantees continuity for all material interfaces. The results of the reconstruction are shown in Figure 5. Note this algorithm is applicable to any cell type, although we depend on a table that covers the 2^n intersection cases (where n is the number of points in the cell).
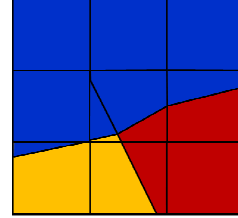

Figure 5: *The final reconstruction for all three materials.*

3.2 Global iteration scheme

As described in section 3.1, our algorithm makes use of volume fractions only implicitly, using them not as a hard criterion, but as input to a process which uses them as values to define geometric clipping planes. However, there is still a direct correlation between the two: for example, a larger input volume fraction of a material will result in the generated volume for that material being larger. We will explore the actual correspondence between input and output volume fractions in Section 4, but we will note here simply that the relationship is not necessarily a trivial linear one: as can even be seen in the example three-material diagrams above, there are many circumstances in which the input and output VFs are not an exact match.

However, we can use this proportionality between input and output volume fractions to improve our scheme. The process we use is one of a global iterative modification of the VFs used as input to the reconstruction. Specifically:

1. Initialize the reconstruction algorithm VF input values to the desired VFs (i.e. the VFs from the original data set).
2. Perform the reconstruction using the current input VFs.
3. Calculate the output VFs achieved by the reconstruction.
4. For each material in each cell, modify the input VF by some percentage of the difference between the desired VF and the achieved output VF.
5. Repeat steps #2, #3, and #4 until iteration stops.

For example, suppose our data has a VF of 0.2 material A and 0.8 material B in some cell, but we achieve 0.1 and 0.9, respectively, after our first reconstruction. With a 40% percentage used in step 4, we will modify these target VFs to be 0.24 and 0.76 and begin the next iteration.

Note that because modified VF values are averaged from cells to nodes during the iteration, modifications to one cell will affect neighboring cells. Therefore, this is not a local adjustment and it prevents a simple analytic solution in each

	PLIC	Discrete	Isovolume-Z	Equi-T	Equi-Z	Equi-Z/iterative
4.1 Mesh Type Support	all	rectilinear only	all	all	all	all
4.2 Geometric Connectivity	no	yes (axis-aligned)	no	yes	yes	yes
4.3 Multi Material Support	yes (sensitive)	yes	no	yes	yes	yes
4.4 Smooth, Low-Artifact Output	no	no	yes	partial	yes	yes
4.5 Volume Accuracy	yes (exact)	yes (bounded)	no	no	no	yes (not guaranteed)
4.6 Surface Accuracy	no	no	yes	yes	yes	yes
4.7 Memory Complexity	moderate	poor	excellent	moderate	excellent	excellent
4.8 Runtime Performance	poor	poor	moderate	excellent	excellent	moderate

Table 1: *Summary of comparative evaluation of reconstruction algorithms.*

cell, also removing guarantees of convergence. However, this global aspect is necessary; without this connection between cells, connectivity and smoothness would be lost.

In Step #4, the percentage of the VF difference used to modify the target VFs can be thought of as a damping factor: a larger value will approach the desired VF values more quickly, but as such can result in oscillations which prevent convergence at all. We have found empirically that values between 10% and 40% generally approach the target VFs quickly while avoiding oscillations. Also, as convergence is not guaranteed, the scheme is most useful with the number of iterations specified manually. We investigate these convergence properties in Section 4.5.1.

4   Evaluation

In this section, we compare our algorithm against other algorithms, both visualization- and simulation-oriented. Each of the algorithms is implemented inside the VisIt visualization and analysis application [CBB*05], and we use these implementations for our comparisons. These include a piecewise-linear interface construction ("PLIC") algorithm, an implementation of the algorithm from Anderson et al. [AGD*08] ("Discrete"), and a volumetric variant of an isosurface algorithm ("Isovolume"). For our technique which clips cells based on equi-surfaces, we show results using both reconstruction variants: one using the pure triangular/tetrahedral approach ("Equi-T"), and one using a the more complete selection of cell types from the finite element zoo ("Equi-Z"). The latter variant can utilize our global iteration scheme with a fixed number of iterations (e.g. "Equi-Z/i5" for five iterations). Table 1 summarizes the results of the comparative evaluation. We explore these results in detail below in light of visualization and analysis requirements, including overall capabilities (4.1-4.4), accuracy (4.5-4.6), and performance (4.7-4.8).

4.1   Mesh type support

All algorithms compared here work in both two- and three-dimensions. Furthermore, all work with regular, structured, or unstructured data sets, with the exception of the Discrete algorithm, which is limited to regular grids. We use all types of data sets below, using a two-pronged approach to evaluate these algorithms: use well-defined synthetic test data to examine critical algorithmic features on a small scale, and larger simulation data to evaluate the methods in a real-world setting.

4.2   Geometric connectivity

For many geometric algorithms, good cell connectivity is required – this is why unstructured meshes are generally not

sets of independent cells, but instead use data structures that share points among neighboring cells. Visualization and analysis examples requiring good connectivity include calculating external faces of a data set or algorithms which utilize the set of cells adjacent to a point. Not every interface reconstruction technique is inherently capable of generating correct connectivity.

For example, a standard PLIC algorithm generates an interface in each cell independently of its neighbors, and the intersection point between adjacent cells will not match up. This means that generating a closed surface without duplicate geometry is impossible without a complex algorithm which can split the reconstructed geometry many extra times. See Figure 6 for an example. In practice, this is a major liability for using PLIC style reconstruction for the many analysis algorithms requiring a well-defined surface. However, for the purpose of the fairest analysis of PLIC algorithms, we implemented a duplicate surface geometry removal algorithm, which allows us take measurements based on the ideal surface theoretically achievable with PLIC-style reconstruction.
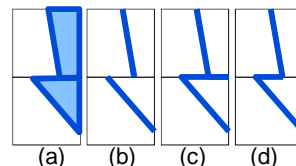

Figure 6: *(a) Volumetric reconstruction with PLIC style algorithms. (b) A surface-only PLIC reconstruction leaves holes. (c) Generating the surface from a volumetric reconstruction leaves duplicate geometry. (d) The best result has sharp angles and requires additional expense to generate. Although duplicate partial surface removal (i.e. (d)) is not a standard part of PLIC reconstructions, we implemented it for the comparisons in this paper.*

The Discrete algorithm quantizes space into sub-voxels. As such, good connectivity is somewhat straightforward in that the edges or faces between sub-voxels with differing materials define the interface, but the connectivity is limited to only axis-aligned geometry (like a rectilinear grid), imitating some weaknesses of a SLIC algorithm.

A standard isosurface based approach that creates only lines in two dimensions and surfaces in three dimensions cannot be used for much analysis other than generating images; to perform analysis on the reconstructed result requires a more complex volumetric algorithm. And, like a normal isosurface algorithm, this type of reconstruction can easily generate good connectivity between adjacent cells. However, a standard isovolume algorithm requires one pass for each material, creating volumes at the 50% iso-level.

This is not conducive to generating connectivity between the materials occupying a single cell, since each material's geometry is generated in isolation. In a two-material case this could be remedied, as the 50% levels happen to match up, but in general this is not possible – see the multi-material support section (4.3) below for more details.

As described in Section 3, our algorithm shares some similarities with an isovolume algorithm in that adjacent cells will naturally share intersection points. However, our approach also solves the inter-material connectivity problem: it does not use a fixed value, but instead defines a single point of intersection between each pair of materials. By recursing through the materials in a pairwise evaluation, this results in even the many-material case having inherently correct connectivity across cells and materials.
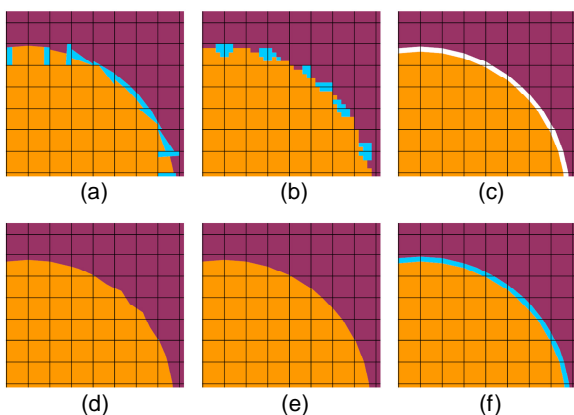
4.3 Multi-material support

Figure 7: *Thin-shell torus test data set. Close-up of reconstruction using (a) PLIC (b) Discrete (c) Isovolume (d) Equi-T (e) Equi-Z (f) Equi-Z/i20.*

In general, material-supporting simulation codes operate on many materials, and so correctly supporting multiple materials within a single cell is a key requirement for correctness in a reconstruction algorithm. Failure can take the form of a total inability to handle more than two materials in one cell to a gross shape mischaracterization. To explore this capability, we use the "Torus" data set. This is a three-material data set, where an inner circle of the first material is surrounded by a thin toroidal shell of a second material, in turn surrounded by a third material as background. This is also useful for studying the algorithms under low-volume fraction conditions. For example, Figure 7 shows a close-up from the results of this data set reconstructed by each algorithm. The PLIC algorithm must reconstruct cells working from the outside-in, which means it can potentially get the material ordering incorrect in cases with three or more materials; this has occurred in some cells, breaking any chance at a complete shell. The Discrete algorithm tends to group pieces of materials together, resulting here in uneven thickness. The Isovolume approach misses the thin material entirely, leaving a hole due to its lack of three-material support. Without iteration, our new algorithm misses the thin material as well, but does not leave a hole. When several iterations are added, our

algorithm successfully reconstructs the thin shell, leaving it smoothly connected through the entire 360°.

4.4 Smoothness and physical realism

As previously mentioned, the material interface problem is under-constrained. A reconstruction algorithm is free to choose any surface within the cell as long as it reflects the volume fractions. From a visualization standpoint, the choice of surface within a cell is more than aesthetics; if a jagged edge or lumpy surface is used when a smooth edge or flat surface would have met the requirements imposed by the volume fractions, then the reconstruction algorithm has introduced artifacts and assumed information which did not exist in the data. These artifacts can be distracting, physically unrealistic, or even misleading. Thus, in the absence of extra information, smooth or straight surfaces are desirable, as they introduce the least new information. Though it does manifest in some analysis calculations (such as surface area, which we explore in Section 4.6), this form of realism is hard to quantify. However, it can be subjectively apparent. For example, in Figure 7(f), the ability of our new algorithm to correctly generate a thin shell, and particularly to do so with a smooth continuous interface, is a strong example of its realism and minimal introduction of artifacts compared to other algorithms.
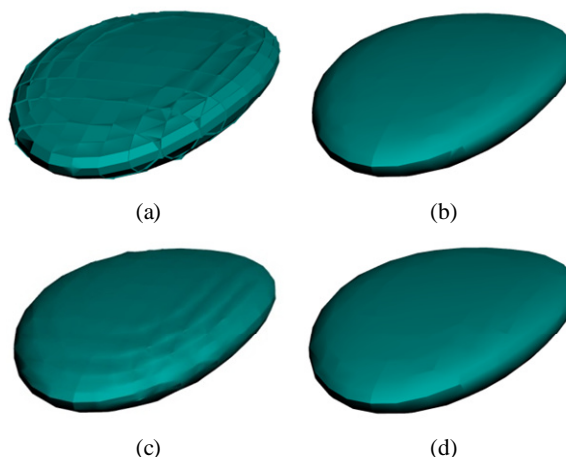

Figure 8: *Unstructured mesh ovoid reconstruction using (a) PLIC (b) Isovolume (c) Equi-T (d) Equi-Z/i5.*

Figure 8 explores these artifacts in three dimensions with a deformed ovoid material in an unstructured grid. The Discrete algorithm does not function on unstructured grids, so we have not evaluated it here, but we note that all angles in this algorithm will be 90 degrees, the worst possible. Note again the sharp surface irregularities when using a PLIC algorithm; these will be present on most nontrivial problems. Isovolume is very smooth, but it achieves this with a loss in volume fraction accuracy. The results from our new algorithm highlight the importance of the clipping variant: when element types are limited to tetrahedral cells, it still exhibits surface artifacts (though less prominent than PLIC). Allowed the full complement of primitive types, it achieves results subjectively as artifact-free as Isovolume as well as a more accurate (slightly larger) volume.
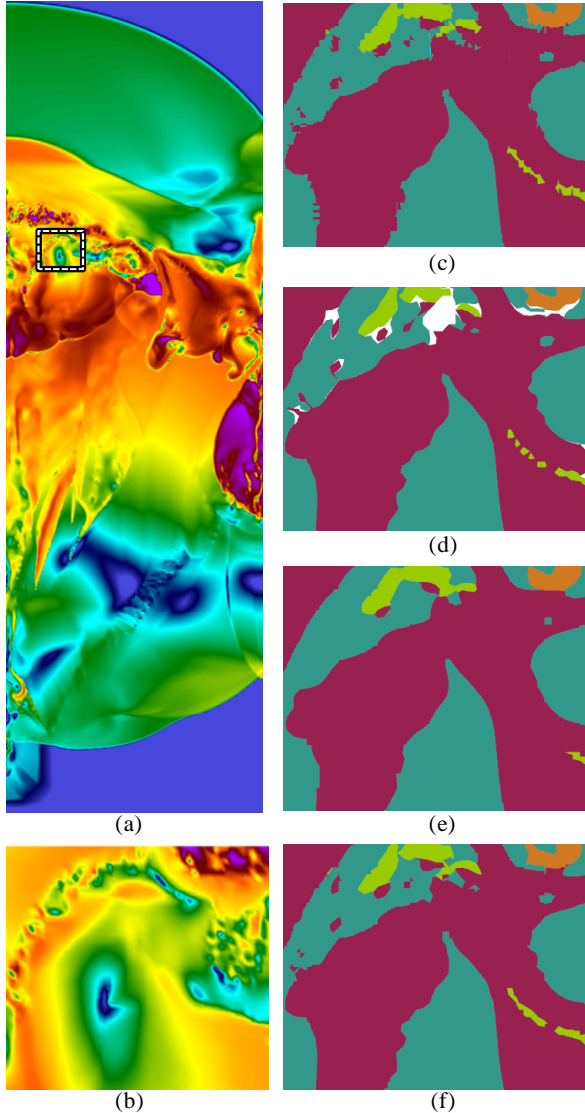
Figure 9: *(a) Unstructured dataset from ALE3D simulation showing material velocities. (b) Close-up of zoom region showing velocities. Reconstructions shown in zoom region are (c) PLIC, (d) Isovolume, (e) Equi-Z, (f) Equi-Z/i30.*

Figure 9 shows a hyper velocity impact of a large particle upon several layers of shielding from the ALE3D unstructured grid simulation code [ALE3D], decomposed over 64 parallel computational domains. At this late time step, the materials have become twisted and stretched under high pressure, like turbulent fluids mixing. In panels (c) through (f) we see a close zoom of the highlighted region to examine fine scale details of the reconstruction algorithms. Of course, the "correct" reconstruction is ill-defined, although the PLIC algorithm is guaranteed to faithfully represent each material in a given cell. However, the PLIC algorithm introduces many jagged edges, creating sharp features not implied by the data. Further, the Isovolume algorithm creates large holes near the multi-material areas. Our new algorithm produces very smooth results, though it can miss many small pieces of materials unless iteration is enabled.
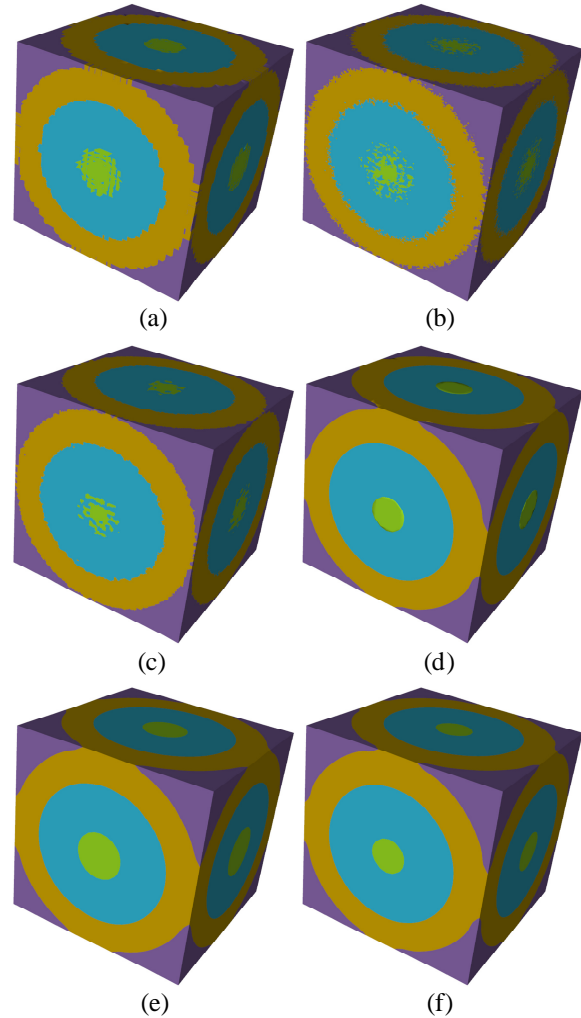

Figure 10: *"Rect3D" Regular grid data set of nested spheres reconstructed with (a) PLIC (b) Discrete/10sec, (c) Discrete/60sec, (d) Isovolume, (e) Equi-Z, (f) Equi-Z/i5.*

Figure 10 shows "Rect3D", a three dimensional regular grid data set comprising eight nested sphere materials, four of which are visible on the outside faces as shown. We expect perfect circular boundaries on these four material spheres where they intersect the planar faces of the cube. The PLIC algorithm again shows discontinuities at cell boundaries. The Discrete algorithm results contain random noise, even after long periods of iteration, and its realism at material boundaries is limited by the quantization factor. Isovolume generates smooth results but has some minor lighting artifacts near material boundaries due to its poor inter-material connectivity. Our new algorithm (when not restricted to tetrahedral cells) generates smooth results and cohesive external surfaces, both without and with iteration.

4.5 Volume fraction accuracy

One metric is most often selected as foremost in importance in interface reconstruction algorithms: their ability to consistently reproduce the input volume fractions as the correct volume in the output geometry for each cell.

While this capability often comes at the cost of others, the degree to which the algorithms honor volume fractions is nonetheless a useful metric, and so we examine it here.

For an initial study of volume fraction accuracy, we utilize a straightforward 2D "Linear" test data sequence, a two-material data set where a single-slope continuous linear interface divides the entire problem, intersecting the central cell of a small grid over a full range of slopes and volume fractions (35 data sets in total). The results over this full sequence of data are shown in Table 2. The PLIC algorithm is obviously very strong, with error on the order of single-precision floating point limits. The Discrete approach has fixed error bounds depending on the chosen quantization factor (15x15 here). As expected for a two material mixed cell case, the Isovolume approach and our new algorithm perform identically on this metric, with maximum errors of 2.3% and a median error of 0.13%. Even with only two iterations, our new algorithm improves even more, with error dropping by about half, to a maximum of 1.0% and a median of 0.072%. When our algorithm is restricted to triangular output (with no iteration), it performs the worst, with much larger errors (1.3% median and 3.4% max), showing the importance of the approach to clipping.

	PLIC	Discrete	Isovol	Equi-T	Equi-Z	Equi-Z/i2
Median	7.1e-7	2.4e-3	1.3e-3	1.3e-2	1.3e-3	7.2e-4
Maximum	1.6e-6	4.3e-3	2.3e-2	3.4e-2	2.3e-2	1.0e-2

Table 2: *Volume fraction reconstruction error in the central cell of the "Linear" data set sequence.*
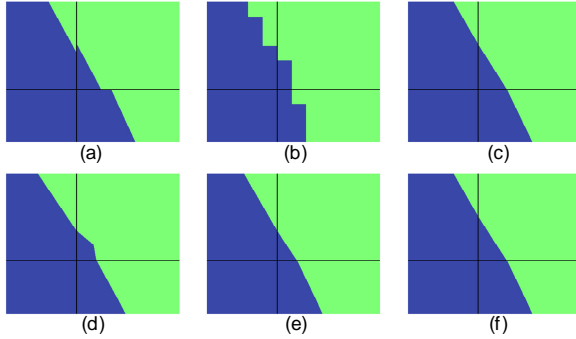

(a) (b) (c)

(d) (e) (f)

Figure 11: *Close zoom of a blue circle against a green background material reconstructed using (a) PLIC (b) Discrete (c) Isovolume (d) Equi-T (e) Equi-Z (f) Equi-Z/i10.*

We also use a "Circle" data sequence to examine the behavior on ensemble data with nonlinear boundaries. This is a two-material data set, where a unit circle of one material is overlaid on a background material. This is a sequence of data sets of varying resolution, ranging from a 10x10 to a 50x50 structured grid, and can be used to study the convergence properties of the algorithms under increasing resolution. Figure 11 shows a close-up of the reconstruction of this dataset for each algorithm.

As the data represents a unit circle, it has a total area over all cells of π at any resolution. The area achieved by each algorithm is shown in Figure 12. As expected, PLIC is accurate with this metric, the Discrete algorithm has fixed error bounds, and Isovolume and our new method without iteration perform similarly as this is a two-material case. Note that *all* algorithms do exhibit asymptotic convergence

to the correct volume fraction answer under increasing resolution. The key feature to note here is the pronounced effect our iteration scheme has on the volume fraction accuracy; errors on the ensemble are noticeably improved compared to what was achieved without iteration.
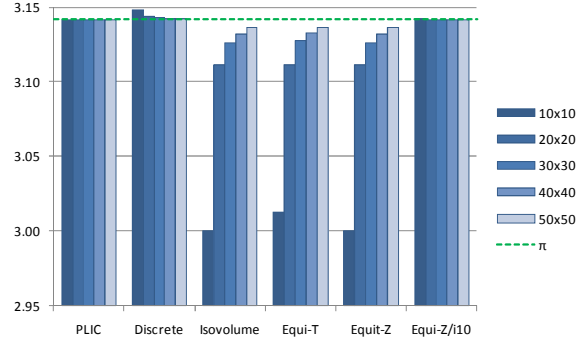

Figure 12: *Area of a unit circle material sampled onto a grid as reconstructed by each algorithm for a sequence of increasing resolutions. The correct answer at all resolutions is π.*

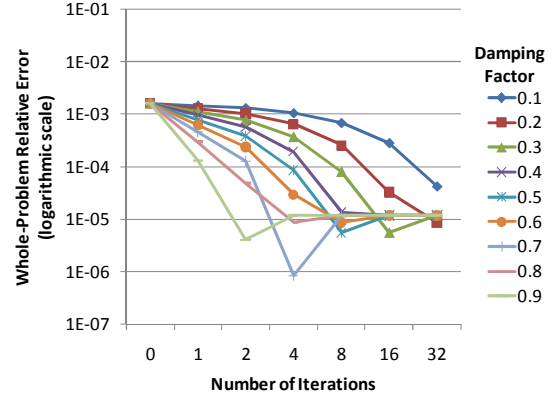### 4.5.1 Volume fraction convergence

Figure 13: *Relative error of material volume on the "50x50 Circle" test data set for various convergence parameters in the Equi-Z/iterative algorithm.*
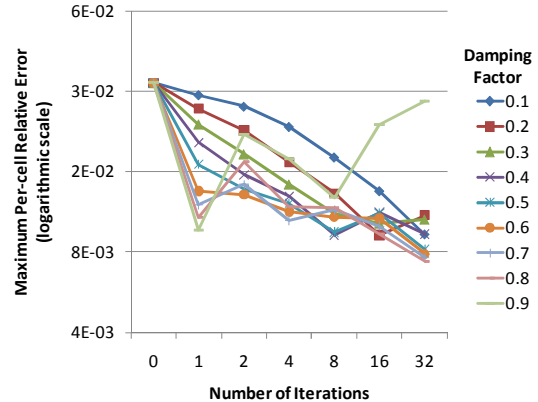

Figure 14: *Maximum relative error for any single cell in the "Linear" test data sequence for various convergence parameters in the Equi-Z/iterative algorithm.*

In Figure 13 we investigate the convergence properties of the Equi-Z algorithm under iteration on the Circle data set. This test examines the summed error contributions over all reconstructed cells. We see the combined relative error is on the order of 10^{-3}, and with iteration generally improves by two orders of magnitude to 10^{-5}. All damping factors show similar total improvement here, with larger damping factors converging as quickly as one order of magnitude per iteration. Figure 14 shows the results from single-cell reconstructions on the Linear data sequence. These results show the maximum error for any linear interface intersecting the cell of interest, generally converging to less than 1%. Here we see a penalty for using a damping factor that is too large: the 0.9 factor results in oscillations which effectively negate the benefits of iteration.

4.6 Surface accuracy

As mentioned, honoring volume fractions is often considered the paramount metric in interface reconstruction algorithms. However, preserving volume fractions does not ensure a good reconstruction from an analysis perspective. The role of material interface reconstruction algorithms in analysis informs examples: How closely does our material approximate a sphere? How compact is our material? Where is its centroid? What is the moment of inertia? All of these are dependent on a correct reconstruction of the *shape* of the object. The surface area of our reconstructed geometry, directly or indirectly, plays a role in the calculation for each of these examples, and so we study it here to provide a quantitative basis for this metric.
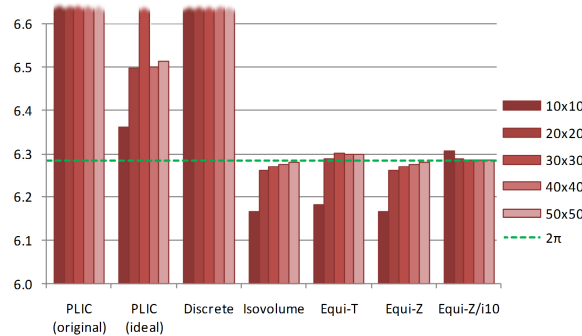

Figure 15: *Perimeter of a unit circle material sampled onto a grid as reconstructed by each algorithm for a sequence of increasing resolutions. The correct answer at all resolutions is 2π.*

We use our "Circle" test again to explore this metric, since it has well known properties, and we can examine the nature of surface area accuracy while being cognizant of the volume fraction accuracy. Being a unit circle, it has a known correct answer for surface area (perimeter in 2D) as well: 2π. Figure 15 shows the surface area measurements on this sequence of increasing resolution data sets. The first result, the unmodified PLIC algorithm, is off the chart – this is due to the connectivity problem discussed in section 4.2, where adjacent cells do not have correct connectivity, leading to duplicate geometry and a meaningless measurement of surface area. We applied an algorithm to remove the duplicate portions of this geometry, leading to values which PLIC could theoretically achieve in an ideal

case, yet the results are still much higher than the correct answer. This is due to interface mismatches at cell boundaries, resulting in a jagged surface where small discontinuities add up over the whole perimeter of the circle. (Figure 11(a) shows this phenomenon using PLIC on this data set.) Furthermore, the result does not appear to be converging to the correct answer at higher resolutions; even with a "perfect" choice of interface orientation, more cells can simply lead to more wobbles, even if the wobbles are smaller.

The Discrete algorithm has a similar problem; its interface is defined by small axis-aligned stair-stepped line segments, and thus is at a disadvantage in this metric, generating a surface perimeter far too large, also off the chart.

The other algorithms, due in part to their inherent connectivity, perform better. Our new algorithm, even without iteration, shows asymptotic convergence to the correct answer, but only the Equi-Z variant. In contrast, Equi-T, by being restricted to triangular primitives, introduces small wobbles of its own, though they are less pronounced than PLIC. When iteration is added, our new algorithm outperforms all others at every resolution.

Figure 16 shows this from a different perspective by quantifying the surface roughness on this Circle data sequence. We know the sum of the exterior angles of a closed polygon is 2π; however, a rougher surface will have more negative angles where it is concave. As the data being reconstructed represents a convex shape (i.e. a unit circle), we expect no negative angles in a smooth reconstruction. By summing the *absolute value* of these angles, we can measure surface roughness as the amount by which this summation is greater than 2π. Note that Equi-Z and Isovolume achieve close to or exactly 2π, while Equi-T and PLIC are significantly higher (note the logarithmic scale), indicating a surface with more jagged and undulating edges.
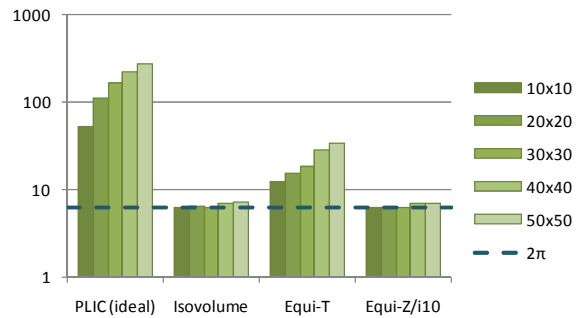

Figure 16: *Sum of absolute value of exterior angles in Circle boundary at increasing resolutions. A convex polygon has value 2π; each jagged edge results in a negative angle, in turn resulting in an increase in this sum.*

4.7 Memory complexity

Complex output geometry has consequences, from the extra runtime in later analysis, to memory usage both during and after reconstruction, to memory imbalances which can affect scaling in a parallel setting. Here we measure this output geometry complexity directly.

	PLIC (orig)	PLIC (ideal)	Isovol	Equi-T	Equi-Z	Equi-Z/i5
Polygons	2116	2116	2421	2245	2267	2116
Lines	592	383	332	429	350	367

Table 3: *Average number of 2D polygons and lines created in each domain for material 3 in the 2D ALE3D data set.*

Table 3 shows the amount of geometry created during the reconstruction on the 2D ALE3D data set. We select only one material in order to enable fair comparisons with the PLIC and Isovolume algorithms, which cannot generate good connectivity between materials and would otherwise be many times higher. By using this method we can note that the PLIC algorithm does generate a slightly lower number of cells due to its use of arbitrary polygons, but the number of lines lying along the interface surface is generally larger, even after applying a post-processing filter to remove duplicate geometry. The extra subdivision of the Equi-T algorithm, due to its restriction to triangular output shapes, increases its number of surface lines beyond Equi-Z, and the Isovolume algorithm is lowest at the cost of inaccurate volume fractions in its reconstruction.

	PLIC (orig)	PLIC (ideal)	Discrete 10sec	Discrete 60sec	Isovol	Equi-T	Equi-Z	Equi-Z/i5
Cells	13k	13k	1,029k	1,029k	38k	67k	64k	64k
Faces	75k	65k	373k	265k	20k	44k	22k	23k

Table 4: *Number of volumetric cells and surface faces created for material 3 in the 3D Rect3D data set. Note the cells for PLIC are complex arbitrary polyhedra.*

Table 4 shows the 3D equivalent of the same information for the Rect3D data set, again selecting only one material to enable fair comparisons for poor-connectivity algorithms. The fixed subdivision in the Discrete approach causes an explosion in the number of output cells and surface geometry. We see PLIC has a benefit in terms of volumetric cells due to its use of arbitrary polyhedra, but this does not mean the geometry is strictly simpler; this can seen in the number of surface faces being significantly larger than any algorithm but Discrete, even with duplicate geometry removed. We see again that Isovolume performs very well on this metric, particularly in terms of number of faces. Our new algorithm performs nearly as well, though we note that the Equi-Z variants, both with and without iteration, are improved over Equi-T, as the latter results in extra subdivision from its use of only tetrahedral elements.

4.8 Reconstruction performance

PLIC	Isovol	Equi-T	Equi-Z	Equi-Z/i5
37.26 ms	67.57 ms	0.73 ms	1.39 ms	18.27 ms

Table 5: *Median reconstruction runtime per domain on the 2D ALE3D simulation data set. The Discrete algorithm was not tested as it only operates on structured meshes.*

We test performance using the two largest test data sets: the "Rect3D" nested spheres 3D regular grid, and the 2D unstructured data set from an ALE3D simulation, both with eight total materials. See Section 4.4 for images and more a detailed description of these data sets. These results were generated on a 2.5GHz Intel Harpertown system with 8GB RAM and GCC 4.2 with –O2 optimization.

In Table 5 we see the time taken during the reconstruction phase on the 2D ALE3D data set for each algorithm supporting unstructured grids. Note that Isovolume is the slowest, as it requires multiple passes to support many materials. The variants of our new algorithm are the fastest, though we note that the Equi-T version, being restricted to triangular shapes, is simpler and faster. There is additional cost associated with each iteration pass, but even with iteration the absolute runtime of our algorithm is still a significant improvement compared to the alternatives.

	PLIC	Discrete 10sec	Isovol	Equi-T	Equi-Z	Equi-Z/i5
Volumetric reconstruction	7.31 s	9.95 s	2.08 s	0.16 s	0.13 s	0.93 s
Surface extraction	6.02 s	2.39 s	0.09 s	0.16 s	0.12 s	0.13 s

Table 6: *Runtime on Rect3D for both full 3D reconstruction and extraction of surface geometry.*

Table 6 shows the runtime on the Rect3D data set. Again, even with multiple iterations, our new algorithm is the fastest for reconstruction. Note that the tetrahedral 3D Equi-T variant is more complex than its triangular 2D equivalent, and also generates more cells, so here we see a benefit by using the Equi-Z variant. Also, note that we measured the performance for the Discrete algorithm, with its dial-an-accuracy approach, at ten seconds, while we noted in Figure 10 that even with a sixty second iteration time there were still visible artifacts. The second row in Table 6 shows the time to extract the surface geometry. This depends on the complexity and number of output cells, and so this measure can be indicative of performance for other geometric visualization and analysis tasks occurring after reconstruction. For this metric, the large number of cells in the Discrete algorithm and the complexity of the arbitrary polyhedra generated in the PLIC algorithm are major reasons for their reduced performance. Though the Isovolume approach is faster in this extraction phase, Equi-T and Equi-Z are close behind. The combined reconstruction time is thus significantly faster in our proposed algorithm.

5 Conclusions and Future Work

In this paper, we present a new approach to material interface reconstruction which improves upon existing methods in several ways. First and foremost, it has correct connectivity and supports multiple materials – features which are critical for analysis and visualization. While making no guarantees for reconstructed accuracy in volume fraction, we have described an iteration technique which largely addresses this problem. Additionally, we contend that volume accuracy is only one of several accuracy metrics critical for analysis; when measuring surface area, for instance, other algorithms have severe accuracy weaknesses in comparison with our technique. And finally, our method has good performance on real data sets while exhibiting good subjective realism and aesthetically pleasing reconstructions due to its inherent smoothness and material placement. Combined, this evaluation shows that

our approach to material interface reconstruction strikes a favorable balance in the visualization and analysis-space.

We expect future work to focus on improvements to the iteration scheme, such as material ordering, damping factors, and potentially even local iteration modifications which can provide accuracy guarantees. In addition, we expect to more fully study the scalability on large data sets.

Acknowledgements

References

[AS07] AHN, H.T. & SHASHKOV, M. Multi-material interface reconstruction on generalized polyhedral meshes. *Journal of Computational Physics 226*, 2096-2132 (2007).

[ALE3D] ALE3D TEAM. ALE3D Home Page. At https://wci.llnl.gov/codes/ale3d/

[Ams66] AMSDEN, A.A.. Particle-in-cell method for the calculation of the dynamics of compressible fluids. *LA-3466, Los Alamos Scientific Lab., Univ. of California, N. Mex.* (1966)

[AGD*08] ANDERSON, J.C., GARTH, C., DUCHAINEAU, M.A. & JOY, K. Discrete Multi-Material Interface Reconstruction for Volume Fraction Data. *Computer Graphics Forum (Proc. of Eurographics/IEEE-VGTC Symposium on Visualization 2008) 27*, (2008).

[BL03] BANKS, D.C. & LINTON, S. Counting cases in marching cubes: Toward a generic algorithm for producing substitopes. *Proceedings of the 14th IEEE Visualization 2003* (VIS'03) 8 (2003).

[BJH*00] BONNELL, K.S., JOY, K.I., HAMANN, B., SCHIKORE, D.R. & DUCHAINEAU, M. Constructing material interfaces from data sets with volume-fraction information. *Proceedings of the conference on Visualization'00,* 367-372 (2000).

[CBB*05] CHILDS, H., BONNELL, K., BRUGGER, E., MEREDITH, J., MILLER, M., WHITLOCK, B., MAX, N. A contract based system for large data visualization. *Proceedings of the conference on Visualization*, 2005. VIS 05. IEEE 191-198 (2005).

[HSS*97] HEGE, H.C., SEEBASS, M., STALLING, D. & ZÖCKLER, M. A generalized marching cubes algorithm based on non-binary classifications. *ZIB Preprint SC 97–05* (1997).

[HN81] HIRT, C.W. & NICHOLS, B.D. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, 201-225 (1981).

[LC87] LORENSEN, W.E. & CLINE, H.E. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 169 (1987).

[Mer04] MEREDITH, J. Material Interface Reconstruction in VisIt, *Nuclear Explosives Code Developers Conference, '04.* NECDC. (2004).

[NW76] NOH, W.F. & WOODWARD, P. SLIC/simple line interface calculation. *Lecture Notes in Physics 59*, 330-340 (1976).

[OS88] OSHER, S. & SETHIAN, J.A. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 12-49 (1988).

[PY92] PARKER, B. & YOUNGS, D. Two and three dimensional Eulerian simulation of fluid flow with material interfaces. *Third Zababakhin Scientific Talks.* (1992).

[RBW99] ROBERTS, L., BRUGGER, E.S. & WOOKEY, S.G. MeshTV: scientific visualization and graphical analysis software. *University of California Research Lab Report, Lawrence Livermore National Laboratory, UCRL-JC-118600* (1999).

[WS03] WU, Z. & SULLIVAN, J.M. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering 58*, 189-207 (2003).