# Construction of Simplified Boundary Surfaces from Serial-sectioned Metal Micrographs

Scott E. Dillard[1,2,4*] John F. Bingert[3†] Dan Thoma[4‡] Bernd Hamann[1,2§]

[1]Institute for Data Analysis and Visualization (IDAV)
University of California, Davis

[2]Department of Computer Science
University of California, Davis

[3]Materials Science and Technology Division
Los Alamos National Laboratory

[4]Materials Design Institute
Los Alamos National Laboratory

## Abstract

We present a method for extracting boundary surfaces from segmented cross-section image data. We use a constrained Potts model to interpolate an arbitrary number of region boundaries between segmented images. This produces a segmented volume from which we extract a triangulated boundary surface using well-known marching tetrahedra methods. This surface contains staircase-like artifacts and an abundance of unnecessary triangles. We describe an approach that addresses these problems with a voxel-accurate simplification algorithm that reduces surface complexity by an order of magnitude. Our boundary interpolation and simplification methods are novel contributions to the study of surface extraction from segmented cross-sections. We have applied our method to construct polycrystal grain boundary surfaces from micrographs of a sample of the metal tantalum.

**Keywords:** Surface extraction, Polygonal meshes, Visualization in Physical Sciences, Life Sciences and Engineering.

**Index Terms:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary Representations I.4.6 [Image processing and computer vision]: Segmentation—Region growing, partitioning; J.2 [Computer Applications]: Physical Sciences and Engineering

## 1 Introduction

Triangle meshes are a common representation of surface structures. They are convenient for visualizing surfaces using computer graphics hardware, and they provide a succinct and precise representation of a surface which can be used for further applications, such as simulations. In many imaging applications, a triangle mesh must be constructed from some other representation of the structure, such as a volume image (voxels) or a stack of planar curves. The problem of extracting a surface from either form of data has been studied extensively in the computer graphics and visualization fields. In this paper we make novel contributions to the study of constructing surfaces from planar boundary curves and extracting separating surfaces from segmented volumes. We describe a method for interpolating segmented region boundaries between planar sections which can track an arbitrary number of regions, and we introduce an algorithm to extract a simplified, voxel-accurate triangle mesh from the resulting segmented volume.

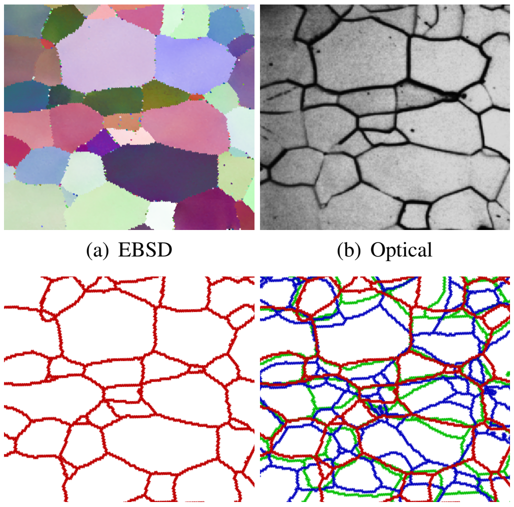
*e-mail:sedillard@ucdavis.edu
†e-mail:bingert@lanl.gov
‡e-mail:thoma@lanl.gov
§e-mail:hamann@cs.ucdavis.edu

Our method extracts crystal grain boundaries of polycrystalline materials, given serially sectioned micrographs as input. The configuration of most materials of technological importance consists of polycrystalline aggregates. The properties possessed by these polycrystals are a function of the size, morphology, phase, and spatial correlation of the constituent crystals, along with any additional constituents such as precipitates and inclusions, and defects such as voids and cracks. The spatial relationship of these features is therefore important toward understanding the relationship between structure and property that is central to the materials science discipline. The development of predictive material models also relies on accurate representations of structure. Although two-dimensional sections reveal many microstructural features that may be statistically captured through stereological approaches, for many cases a three-dimensional reconstruction is necessary to fully interpret the structure.

A sample of the metal tantalum was ground and imaged at intervals of 5 microns, and each image has sub-micron resolution. Two types of images were captured, a gray-valued optical image and an electron back-scatter diffraction (EBSD) image which measures the crystallographic lattice orientation of the metal [1]. These images can been seen in Figure 1. Because the EBSD images capture more information than the optical images, automatic segmentation of these images produces more accurate crystal grain boundaries. However, the EBSD scanner takes considerably more time to capture an image than the optical scanner. For this reason, EBSD images are taken only every fifth section, at a spacing of 25 microns. This difference in sampling densities, between the in-slice imaging directions and the through-slice sectioning direction, makes it difficult to stack the segmented images into a coherent volume. Grain boundaries vary significantly between segmented slices, as one can see in Figure 1(d).

Our problem is closely related to a common problem and practice in medical image processing: tomographic images are hand-segmented to identify features of interest, but due the time it takes to segment by hand, these segmentations are only performed on a relatively small subset of the given sections. Binary segmentations can be smoothly interpolated using distance fields, and if there are multiple segmented features, each feature can be smoothed individually and then the collection of smoothed boundaries can be patched together to form a multi-labeled volume [2]. If, however, there exist several hundred or thousand features whose boundaries need to be interpolated, it may not be practical to treat each feature individually.

We present a boundary interpolation method that can track an arbitrary number of boundaries. This interpolation is accomplished using a simple physical model, called the Potts model. Each voxel in the volume is given some segmentation label, and a local energy function is defined using the segmentation labels and optical gray-values of a neighborhood of voxels. Voxels with known segmentation labels are kept fixed while we attempt to find a labeling


(a) EBSD (b) Optical

(c) One segmented slice (d) Three consecutive slices

Figure 1: Two types of imagery: (a) electron backscatter diffraction or EBSD, and (b) optical. The high quality of EBSD images allows for robust automatic segmentation (c), but because they take significantly longer to acquire, EBSD images may be captured sparsely. In such cases, segmentation boundaries can vary dramatically between slices (d).

of unknown voxels which minimizes the total energy of the segmentation.

After interpolating the planar boundary curves, we extract a boundary surface between regions. We use the well-known marching tetrahedra method of Nielson and Franke [24]. This algorithm constructs a surface by considering a small number of segmentation configurations in a tetrahedral domain. These cases are illustrated in Figure 2. Without any additional information (such as a scalar field) to define the surface geometry, we start with the midpoint surface, halfway between differently labeled voxels. The midpoint surface exhibits staircase artifacts which must be smoothed, and it contains many more triangles than are necessary to represent the boundary surface. To mitigate both of these issues, we use a combination smooth-and-simplify algorithm. In order for materials scientists to evaluate the results of the boundary interpolation, we would like for the smooth-and-simplify algorithm to change the boundary surface as little as possible. On complex datasets, the marching tetrahedra algorithm can produce triangle meshes consisting of several million triangles, so some mesh simplification is desired to allow for interactive exploration of the results. To balance these two goals, we have developed a smooth-and-simplify algorithm which guarantees that the boundary surfaces remain confined to those tetrahedra which generate them, while reducing the number of triangles by an order of magnitude.

## 2 Related Work

The problem of creating surfaces from planar cross-sectional curves has been studied extensively. There are various solutions to the problem that rely on graph optimization [9], Delaunay triangulations [4], implicit surfaces [15, 26] and other techniques. The paper by Braude *et al.* [6] contains a brief survey of the field. We follow the approach of Weinstein's scanline-surfacing algorithm [28], solving the surface reconstruction problem in a voxelized space using a separating surface algorithm.

Separating surfaces can be extracted from a segmented volume in one of two ways, either by first decomposing the hexahedral voxel cells into five or six tetrahedral cells, or by operating directly on the hexahedral cells. Both approaches then create triangle surfaces in the interiors of cells which separate differently labeled cell vertices. Tetrahedral methods were developed independently by Müller [22] and Nielson & Franke [24]. Hexahedral methods can be considered multi-region generalizations of marching cubes [19], and thus import similar topological ambiguities. Hege solved these ambiguities with detailed case analysis [12], as did Wu and Sullivan [30], while Bischoff and Kobbelt subdivided ambiguous voxels [3]. Bischoff and Kobbelt allow the user some control over the topology inside a cell, while all other methods, ours included, arbitrarily decide the topology *a priori*, which could lead to unwanted artifacts.

If there are no additional geometric constraints, such as volume fraction information, then one has to choose arbitrarily where to place surface vertices. An obvious choice is to place them at the midpoints of edges of the underlying hexahedral voxel mesh. However this leads to aliasing artifacts. In a binary segmentation, these artifacts can be smoothed before a surface is extracted by filtering a scalar function defined by the segmentation [29, 23], or the separating surfaces can be smoothed after they are created, using some form of constrained smoothing [25]. Reitinger paid special attention to smoothing across the non-manifold edges in a multi-region separating surface [27].

In large volumes it may be desirable to simplify the smoothed surface. Numerous methods for simplifying triangle surfaces have been developed. Luebke surveyed the field with an emphasis on practical applications [20]. In scientific applications, considering the geometric error caused by simplification is important. Kalvin and Taylor provided error bounds by merging nearly-coplanar faces into so-called superfaces, and bounding the distance between a superface and its constituent triangles [16]. Guéziec descibed a method that tracks error bounds at vertices and preserves volume by carefully placing new vertices when an edge is collapsed [11]. The error-bounded simplification methods most similar to our own are the simplification envelopes of Cohen *et al.* [7] and the permission grids of Zelinka and Garland [31]. Both methods create a tolerance volume around the initial mesh and forbid any modification which moves the mesh outside of this volume. Simplification envelopes are created by an offset surface from the original mesh, and permission grids are created by rasterizing the mesh onto a voxel grid.

Bertram *et al.* considered an application that poses many of the same challenges as ours: constructing non-manifold separating surfaces from segmented, time-varying magnetic resonance images of the human heart [2]. Multiple regions (anatomical features of the heart) are manually segmented in sparsely distributed slices of the volume image. Region boundaries are interpolated between segmented slices by interpolating the signed distance fields of these boundaries. The interpolated signed distance field of every region is considered, and a voxel is assigned the segmentation label of the closest region. Separating surfaces are extracted on the dual voxel grid, placing surface vertices inside cells of the primal grid. The surface is smoothed with a Laplacian filter, subject to the constraint that each vertex remain in the voxel that generated it.

Our method offers two main advantages over some of the methods mentioned above. First, whereas the method of Bertram *et al.* calculates the distance field separately for each region boundary, the computational effort required for our boundary interpolation is independent of the number of regions. This is highly desirable in situations when the number of regions is very large, such as in our metal micrograph application. Second, our surface smoothing and simplification approach eliminates both sampling artifacts and excess triangles without introducing error. If smoothing and simplification are treated as separate steps, as done by Reitinger *et al.* [27] and Wu & Sullivan[30], there is a possibility that surface simplification will be hindered by aliasing artifacts which are not completely removed in the smoothing step. This effect is illustrated in Figure 7.

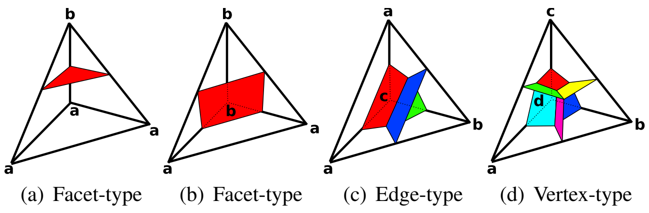


Figure 2: Tetrahedral cases.

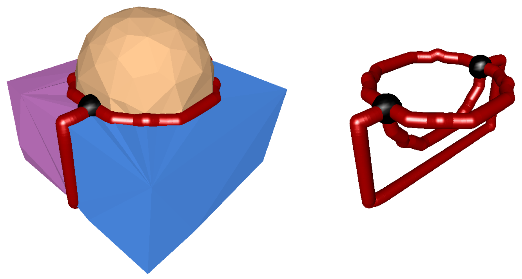


Figure 3: Boundary surface for a three-region segmentation with one sphere and two boxes. The non-manifold boundary edges are shown by red tubes and the boundary vertices are shown by black spheres. There are two vertices, four edges and six facets – one between each pair of regions, and one between each region and the surroundings.

## 3 Background

We are given a stack of segmented images which represent serial cross-sections of some volume. We call the segmented features *regions*, and each pixel of each image is given a label to associate it with a region. The boundary between two regions forms a planar boundary curve, and where three or more regions meet, multiple planar curves come together at a boundary vertex. In our application, regions are crystal grains, which are polyhedron-like structures with nearly planar facets and nearly straight edges. When a sectioning plane intersects a grain, facets appear as planar boundary curves, and grain edges appear as vertices of these curves. Grain vertices, however, are rarely captured in the sectioning plane and their locations can only be inferred.

In three-dimensional space, two adjacent regions are separated by a surface which we refer to as a *boundary facet*, although it need not be planar. Two facets between the same two regions are distinct if the facets are disjoint, i.e., the regions touch at two separate places. Three or more facets meet at a *boundary edge*, and multiple edges meet at a *boundary vertex*. We refer to the collection of boundary facets, edges and vertices as the *boundary surface*. Topologically, the boundary surface is part of a cell complex, with the region interiors being the three-dimensional cells. We assume this complex is closed so that all facets are bounded by edges if they are bounded at all, and all edges are bounded by vertices unless they form closed loops. We can ensure this property by extending the segmented volume with a layer of “void” labels.

Avoiding topological ambiguities is a concern for any boundary extraction method. We take the approach of imposing a tetrahedral domain over the rectilinear voxel grid, following Nielson and Franke [24]. Each hexahedral cell of the voxel grid is subdivided into five tetrahedra. Vertices of the tetrahedra are labeled with their region identifier. To avoid confusion between a tetrahedral vertex of the voxel grid and a triangular vertex of the separating surface, we refer to the former as a *domain vertex* and the latter as a *surface vertex*, and we use the same convention for edges and triangles.

We classify tetrahedra and triangles of the domain by counting the number of distinct labels at their vertices. Figure 2 illustrates these cases. If a domain tetrahedron has two labels among its four vertices, then it is a *facet-type* tetrahedron and some patch of a facet passes through its interior. This patch intersects three or four faces of the tetrahedron, and these faces are also referred to as facet-type faces. If a tetrahedron has three labels, then it is an *edge-type* tetrahedron and some boundary edge passes through the interior. This boundary edge enters the tetrahedron at one face, and leaves it at another face. These two faces are also edge-type, and the remaining two faces are facet-type. A tetrahedron with four distinct labels is a *vertex-type* tetrahedron. It joins four boundary edges and six boundary facets, and all four of the tetrahedral faces are edge-type. Any tetrahedron, triangle or edge of the domain with vertices that are all similarly labeled is called *interior-type*.

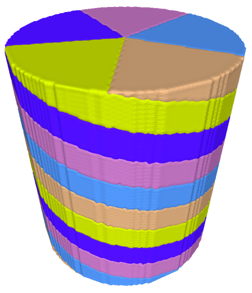
A triangle mesh is a collection of vertices, pairs of the vertices (edges) and triplets of vertices (triangles). It is called a *manifold* triangle mesh *without boundary* if for every point on the surface we can find a small enough ball, centered on this point, which contains a piece of the surface that is homeomorphic to a plane. If the local area around some point is homeomorphic to a half-plane, then the surface has a boundary. In the subsequent discussion, a manifold triangle mesh is one in which every edge is shared by exactly two triangles. If an edge is incident on only one triangle, it is a boundary edge. If every edge is incident on one or two triangles, and every vertex is incident on zero or two boundary edges, then the mesh is manifold-with-boundary. A vertex which is incident on a non-manifold edge is a non-manifold vertex. We ignore other degenerate cases, such as a singular vertex shared by the tips of two cones, because the separating surface algorithm does not produce surfaces in these configurations.

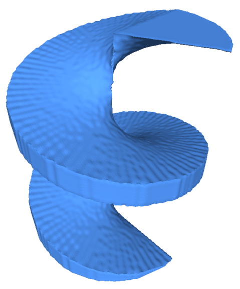
We represent the boundary surface as a non-manifold triangle mesh. It can be decomposed into manifold pieces which are the facets. These facets are represented by manifold-with-boundary triangle meshes. The boundaries of the facets meet at non-manifold boundary edges. A property of the marching tetrahedra surface extraction algorithm is that three boundary facets meet at every boundary edge, and four boundary edges will meet at every boundary vertex. (It is a coincidence that these are the same numbers of faces and edges meeting in a three-dimensional Voronoi diagram, which looks very much like our polycrystal microstructure, and indeed many cellular structures in nature.) Figure 3 illustrates a simple boundary surface example with three regions, plus a void region for the surroundings.

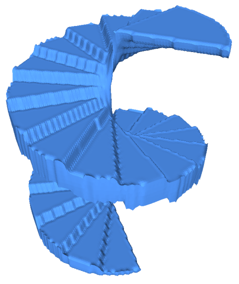
## 4 Boundary Interpolation

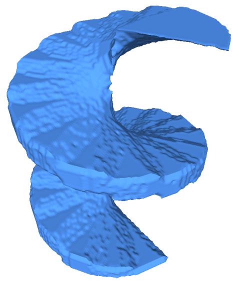
We start with a rasterized representation of the planar boundary curves for each slice, i.e., a segmented image. These images have been registered, and correspondences have been made between regions in adjacent slices, so that the same feature has the same label throughout the entire stack. At this point, a separating surface can already be extracted from the segmented volume using Weinstein’s scanline-surface algorithm [28], which we call “nearest-neighbor” boundary interpolation. If the spacing between segmented slices is adequately small, this will suffice and we can move on to extracting and simplifying the surface. However, if the spacing is large, then the segmentation may change dramatically from one slice to the next, and we want to smoothly interpolate the boundary curves between slices. We accomplish this using the Potts model.

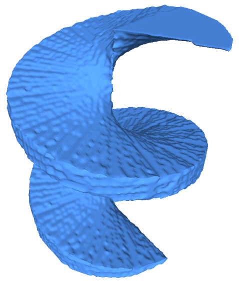
The Potts model has received attention from the computer vision community as a model for early vision, as a special case of Markov random fields [5]. It has also been studied by materials scientists as a model of the formation and motion of crystal grain boundaries in cooling metals [13]. Indeed it is this overlap between computer vision and materials science that makes the Potts model particularly attractive for our application. In the computer vision setting, the


(a) A spiral


(b) One region


(c) Nearest neighbor


(d) Potts without the optical coefficient


(e) Potts with the optical coefficient

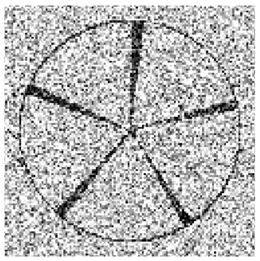
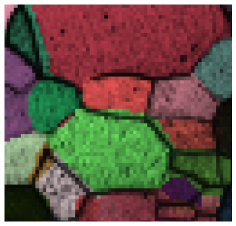

(f) The optical data

Figure 4: A synthetic segmentation on a $128^3$ voxel grid (a), with one region distinguished (b). The boundaries of every fifth slice have been interpolated using nearest-neighbor (c) and Potts interpolation without and with an optical coefficient (d,e). One slice of the optical volume dataset is shown in (f), corrupted by Gaussian noise. Notice that near the center of the spiral, where the boundaries move more slowly, the Potts interpolation can adequately smooth the surface without an optical coefficient, but on the outer edge of the spiral the boundary moves too fast and the information from the gray image is needed. In all images, the surface has been smoothed but not simplified.


(a) Without optical

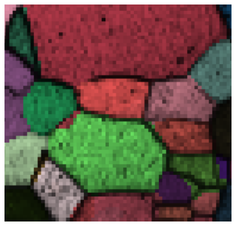

(b) With optical

Figure 5: Incorporating optical data into the Potts energy function improves fidelity of the boundary reconstruction in the regions with unknown labels. In this image, the segmentation labels are represented by color and the optical data is mapped to brightness.

Potts energy of a pixel $i$ in a labeled image $L$ is

$$E_P(i) = \sum_{j \in N_i} U(i,j) \cdot T(L_i \neq L_j). \tag{1}$$

$N$ is the neighborhood of pixels – in 3D we use a 26-voxel neighborhood. $U$ is some image-dependent function and $T(\text{true}) = 1, T(\text{false}) = 0$. Without an image-dependent weight, the Potts energy simply penalizes any neighboring pair of pixels which are labeled differently. In computer vision applications, a labeling which minimizes the Potts energy for the whole image is sought as a discontinuity-preserving restoration or segmentation of the image, so that all regions with similar gray values are compactly clustered. Boykov *et al.* showed that minimizing the Potts energy is NP-hard and provided an efficient graph-cut algorithm which approximates the minimum within a factor of two [5]. However, the complexity of their algorithm depends on the number of segmentation labels, which we consider to be unbounded. To trade guaranteed optimization results for scalability we use a Monte Carlo method, such as simulated annealing [17], the complexity of which is independent of the number of labels.

We use the Potts model to find likely region boundaries between slices in which the boundaries are known. The stack of segmented images is resampled onto a segmented volume $L$, where each image occupies a slice of the voxel grid according to its physical location within the material volume. Slices of the volume which do not have any boundary information are initialized with random segmentation labels. We then iterate the following Monte Carlo procedure:

1. Pick a voxel $i$ at random from a slice which *does not* have segmentation information. Save its label $l_0 = L(i)$. Calculate $p_0 = E_P(i)$, the Potts energy for that voxel.
2. Calculate the set $L_N = \{L(j) | j \in N(i)\}$, the set of all labels of voxels adjacent to voxel $i$.
3. Pick at random a label $l_1$ from $L_N$. Assign voxel $i$ the label $l_1$ and recalculate the new Potts energy $p_1 = E_P(i)$.
4. Pick a random number $x \in [0,1]$. If $x < \exp(-(p_1 - p_0)/t)$, or if $p_1 < p_0$, *accept* the new label $l_1$, otherwise *reject* it and restore the old label $l_0$.

Here, $t$ is the temperature parameter in simulated annealing. We restrict the procedure to those voxels whose segmentation labels are not already known, but the voxels with known grain labels participate in the energy function $E_P$. The entire system is therefore in a state that is very close to the local minimum we are searching for, and since those fixed voxels do not change their labels, the system consistently converges to the desired minimum. Because of the strong influence that fixed voxels have on the system, it is not necessary to use a complicated cooling schedule for simulated annealing. Instead we use a constant non-zero temperature which is smaller than the expected Potts energy of a random labeling. We find that, for the energy scale given by Equation 1 and a 28 voxel neighborhood, $t = 0.9$ is a good choice. Segmentation labels aggregate into smoothly varying regions which agree with the known segmentation, but borders between regions may fluctuate due to the non-zero temperature. To fix borders we "quench" the simulation by setting $t = 0$, causing the system to monotonically transition into a local minimum. We monitor the process visually in order to determine when to quench the system, but fully automatic cooling schedules can be used if automation is required. Convergence can be detected by monitoring the rate at which voxel labels change, and stopping when it falls below a threshold.

If additional information is known about the space between fixed slices, we can incorporate that into the Potts energy through the image-dependent function $U$ in Equation 1, or *optical coefficient*. If we have a scalar function $B(i) \in [0,1]$ that tells us the likelihood that a voxel $i$ is on a boundary, we can set $U(i,j) = 1 - B(j)$. This will reduce the influence that voxel $j$ has on the Potts energy of voxel $i$, so that the energy will be lower when boundaries of the segmentation coincide with boundaries of $B$. Since our optical micrographs are brighter in grain interiors and darker on grain boundaries (Figure 1(b)), we simply take the negative image as $B$, or more simply, the positive image as $U$. Figure 5 shows the effect of incorporating this optical coefficient into the Potts energy. Figure 4 shows the Potts interpolation of a synthetic segmentation. These results are discussed in Section 6.

## 5 Surface Extraction

After boundary interpolation we move on to extracting the boundary surface. If the volume is small it may be sufficient to first build the entire separating surface, smooth it and then simplify it. In large datasets with many regions and facets, we can save considerable memory by treating each facet separately. The triangle surface of a facet is first constructed, and then smoothed and simplified using the algorithm described in Section 5.1, with the constraint that the triangles adjacent to non-manifold boundary edges cannot be removed. These triangles serve as "hooks" which will be later used to glue the facets together along the edges. We keep references to these triangles in a hash table keyed on the edge-type tetrahedra that generated them. Since each facet surface is simplified before the next is built, we can reclaim the memory used for the surface vertices and edges. After all facet surfaces have been built and simplified, we glue them together at the boundary edges and then further simplify the entire mesh.

### 5.1 Smooth-and-simplify Algorithm

Rather than first smoothing the surface and then simplifying it, we perform both simultaneously by iteratively collapsing an edge and then immediately smoothing the neighborhood around the remaining vertex, while constraining the surface to remain in the domain tetrahedra that generated it. Figure 6 illustrates this for a simple 2D case. Our smooth-and-simplify algorithm is guided by the following intuition: Any surface which correctly separates voxels of different regions is "correct" with respect to the data. Among all such surfaces, we would like the one with the fewest triangles. Finding this minimal surface is NP-hard [21], so we approach the problem in a greedy manner – we try to collapse every edge, and if an edge cannot be collapsed, try to collapse it again later after the geometry of the local neighborhood as changed.

We maintain a priority queue of surface edges. There are numerous methods for defining the edge priority, and many of them favor an edge whose collapse will cause the least deviation from the starting mesh, as indicated by some measure of geometric error. We start with the midpoint surface, which is full of aliasing artifacts that we do not wish to preserve, so we avoid any effort to maintain fidelity to the starting mesh. Instead we rely on the constraining domain tetrahedra to guide the surface geometry. We choose the shortest surface edge to collapse first.

When collapsing an edge we must choose a location for the new surface vertex. If both edge vertices are manifold, we choose the new vertex by minimizing the sum of squared distances to all of the triangle planes incident on both vertices. We find this vertex using the quadric error optimization of Garland and Heckbert [10]. Note that estimates of error from the original surface are not propagated through mesh modifications, as is commonly done. Instead, the quadric is computed "from scratch" for each edge collapse. Placing new vertices in this way, rather than simply using the midpoint of the collapsed edge, prevents the surface from shrinking and keeps it farther away from the constraining domain tetrahedra. Additionally, sharp features of the surface are enhanced, as can be seen in Figure 8. If one of the edge vertices is non-manifold, we pick this as the new vertex. Manifold edges which connect two non-manifold vertices are not allowed to collapse. When collapsing a non-manifold

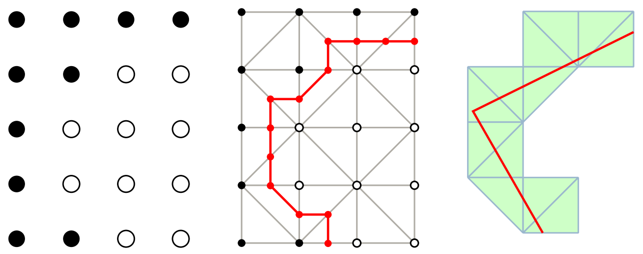


Figure 6: A simple 2D example illustrating our surface extraction approach. The separating surface from the segmentation in (a) is extracted by imposing a triangular domain over the pixels and finding the midpoint surface (b). We attempt to simplify all possible edges of the surface subject to the constraint that the surface remains in the triangles between differently labeled pixels.

edge, we again place the new vertex by minimizing the quadric error of the set of triangle planes incident on both vertices. When a non-manifold edge is incident on a boundary vertex (a vertex which connects four non-manifold edges) we pick the new vertex to be that boundary vertex. Boundary vertices are kept fixed. A simplification step follows the following procedure:

1. Remove the shortest edge from the queue.
2. If the edge cannot be collapsed, go to step 1. Section 5.2 describes the tests used here.
3. Collapse the edge, producing a new vertex $v$.
4. Smooth all vertices adjacent to $v$.
5. Update the priorities of all edges adjacent to $v$.
6. If any edge adjacent to $v$ is not in the queue, then put it in the queue.

Smoothing is performed using a bilaplacian filter, also called the recursive "umbrella operator" by Kobbelt *et al.* [18], with the added constraint of the collision test in Section 5.2. We do not smooth only to improve the visual appearance of the mesh, but to allow for more edges to be collapsed. The domain tetrahedra form a simplification envelope that constrains the surface, but this envelope is jagged and the surface will come into contact with it in many configurations that might be avoided. By applying a smoothing operator after each edge collapse, we redistribute the surface vertices so that edges which may not have been able to collapse are given another chance to do so.

### 5.2 Surface Modification Tests

The first test, called the *topology test*, ensures that we do not alter the topology of the mesh. Following Dey *et al.* [8], let the link of a vertex, Lk $v$, be the set of edges and vertices which intersect some triangle incident on $v$, but which do not intersect $v$. Let the link of an edge, Lk $uv$, be similarly defined. Then the topology test for contracting edge $ab$ is $\mathrm{Lk}\, a \cap \mathrm{Lk}\, b = \mathrm{Lk}\, ab$. Some variations of this test, such as in Hoppe *et al.* [14], additionally check for the case of a tetrahedron collapsing into a degenerate triangle. We note that this additional check is subsumed by the collision test mentioned below. The second test, *inversion test*, prevents triangles from inverting. We check the normal direction of the oriented triangle before and after the modification, and if the normal flips, the modification is not allowed. The third test, *collision test*, prevents surface triangles from leaving the volume spanned by domain tetrahedra which define the facet. This ensures that the surface correctly separates the segmented regions. There are two versions of this test, one for manifold edges and one for non-manifold edges.

For manifold edges within a facet, a simplification envelope is formed by exterior faces of the interior-type tetrahedra on either side of the facet. One such envelope is shown in Figure 7(c). It may be the case, however, that the interior of a region is made up of domain edges and vertices, but not of whole domain tetrahedra and triangles. This can happen, for instance, if the region consists of one or two voxels, or if it contains two large volumes connected by a thread-like structure. To handle these cases we also check for collision with domain edges and vertices.

Triangle-triangle and triangle-edge collisions are tested using standard static collision detection methods. However, to check for collision with a domain vertex we need to use a triangle-point dynamic collision test. When performing an edge collapse, two or three surface triangles are removed, and the remaining triangles in the neighborhood each have one of their vertices moved to a new location. We sweep out a tetrahedron formed by the old triangle and its new vertex. If this tetrahedron contains a domain vertex, the modification is not permitted. We enumerate all domain tetrahedra that might intersect a given surface triangle with a flood-fill. First we locate a domain tetrahedron that intersects the surface triangle, then flood all neighbors of this tetrahedron as long as their faces intersect the surface triangle. This might also be accomplished with more sophisticated triangle-voxel rasterization, however we find the flood-fill is simple and robust, if not optimally efficient.

For non-manifold edges, we restrict the polyline of non-manifold vertices to only intersect edge-type tetrahedra. Before a non-manifold edge is modified, we follow the line segment of the edge through the tetrahedral mesh. If this segment intersects any tetrahedron that is not edge-type or vertex-type, the modification is not permitted.

## 6 Results and Discussion

Figure 4 shows the results of Potts boundary interpolation on a synthetic dataset. A five-region spiral was rendered on to a $128^3$ voxel grid. Figure 4(b) shows one of these regions. To simulate a sparse sampling of the segmentation in one dimension, we removed all segmentation information except for every fifth slice. Figure 4(c) shows the reconstruction of the full volume using only every fifth slice. In this image the surface has been extracted from the $128^2 \times 25$ voxel volume, then stretched out in the slicing direction. Figure 4(d) shows the results of Potts interpolation without an optical coefficient. Clearly the Potts energy defined in Equation 1 penalizes boundary surface area, and therefor segmentations with minimal surface area are favored. Further, it is known that an unconstrained Potts model simulation moves boundaries by mean curvature motion [13]. However, since the Potts energy is defined over a small neighborhood of a regular lattice, we are only minimizing discrete approximations of surface area and curvature. When boundaries vary only slightly between slices, this discrete approximation suffices, but if boundaries change drastically then smooth boundaries will have an insignificant energetic advantage over staircase-like boundaries, due to discretization effects. Notice in Figure 4(d) how the boundary becomes progressively less smooth as we move from the center of the spiral outward, since the segmentation boundary is moving faster along the outer edge of the spiral. By including an optical coefficient in the Potts energy, we can coerce the boundary into regions which are *believed* to be more appropriate, by offering it an energetic advantage in these regions. This boundary belief is quantified by the optical image, which may be noisy. (Surely, if it was a high-quality image we could segment it automatically.) Figure 4(f) shows one slice of the spiral boundary image, which has been corrupted by Gaussian noise, and Figure 4(e) shows the segmentation produced by incorporating this boundary image into the Potts energy.

Figure 7 shows a comparison between two separating surfaces: one which has been smoothed with a Laplacian filter (7(a)) and then simplified using quadric error metrics (7(b)), and another produced by our smooth-and-simplify algorithm (7(e)). We compare our method to quadric error-based simplification because it is a common simplification method and it has been used before to simplify separating surfaces of segmented data [27]. Both simplified surfaces in Figure 7 have been reduced from 8912 triangles to 242. In Figure 7(a) we can see that sampling artifacts from the voxel grid have not been completely smoothed. Any simplification method that attempts to preserve mesh geometry will also preserve these artifacts, as can be seen in Figure 7(b). The inner simplification envelope is shown in Figure 7(c), and in Figure 7(d) we can see that this envelope crosses the surface produced by quadric-based simplification, and thus some domain vertices lie on the wrong side of the separating surface. It is true that we could prevent these errors by not simplifying so aggressively, but there is no way to know how many triangles can be simplified before errors occur, without using some kind of error bound. Since our smooth-and-simplify algorithm incorporates the envelope as a hard constraint, we do not need to specify a target surface mesh size, rather we let it remove all the triangles it can without causing errors.

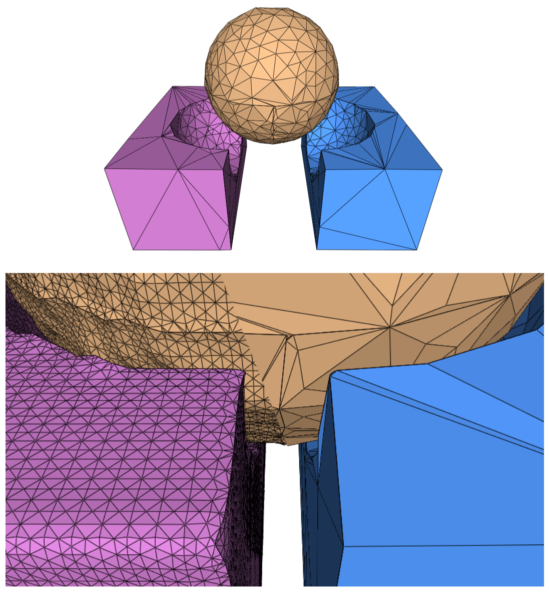


Figure 8: A synthetic $64^3$ dataset with one sphere superimposed over two abutting boxes. The midpoint surface contains over 133,000 triangles, and the simplified surface contains less than 1,000. Using quadric optimization to place new vertices enhances sharp features in the manifold facet surfaces, such as the sides of the boxes.

Figure 8 illustrates the simplification results on a simple non-manifold separating surface. The surface is simplified from 133,534 triangles down to 986. This large reduction is the result of the many co-planar triangles on the sides of the boxes, which can be aggregated into very large triangles, similar to superfaces [16]. Placing new vertices by quadric optimization enhances sharp edges of the boxes. In smooth areas of the volume, such as on the surface of the sphere, the smoothing operator tends to evenly distribute vertices and triangles have nice aspect ratios. However, near sharp features and non-manifold edges, topological and geometric constraints can cause long, thin triangles to form. We acknowledge that, in certain applications, these "bad" triangles can cause problems. We currently make no effort to avoid such triangles.

Figure 9 shows the microstructure of a sample of tantalum measuring approximately 1.6mm along the longest dimension. Figure 10 shows a closer view where grains on the sample boundary

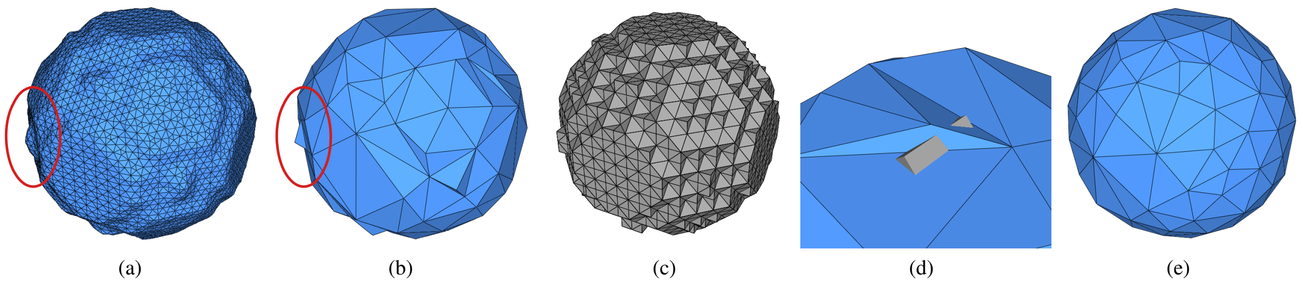


Figure 7: A sphere in a $32^3$ segmented volume. In (a) we see the smoothed midpoint surface, and in (b) it has been simplified according to quadric error metrics. In (c) we see the inner simplification envelope of the sphere, and in (d) a portion of the envelope "pokes" through the surface. Notice how the quadric error metric attempts to preserve highlighted sampling artifacts that were not completely smoothed. Our method (e) ignores these artifacts. The surface in (a) contains 8912 triangles, and the surfaces in (b) and (e) both contain 242 triangles.

| Dataset | BI | M | NM |
|---|---|---|---|
| Tantalum, (800×213×71), Fig. 9 | 10 min. | 1 hr. 5 min. | 12 min. |
| Spiral, ($128^3$), Fig. 4 | 90 sec. | | |
| Sphere and two boxes, ($64^3$), Fig. 8 | | 40 sec. | 3 sec. |

Table 1: Running times for boundary interpolation (BI), manifold facet extraction (M) and non-manifold simplification (NM)

have been removed. The Potts model boundary interpolation was performed on an $800 \times 213 \times 71$ voxel grid. The segmentation consists of 19,037 grains and 93,554 facets. The unsimplified surface contains 23,754,424 triangles, and the simplified surface contains 2,303,982 triangles. Because facets were simplified as they were extracted, using the two-stage algorithm described in Section 5, no more than 8,149,974 triangles were in memory at any point in the process. Table 1 lists the running times for the boundary interpolation and surface extraction, performed on an Intel Core Duo (single-threaded) 2.16GHz computer with 2GB RAM. Profiling reveals that the majority of time is spent in the triangle-triangle intersection collision test, which checks for collisions between surface triangles and domain triangles. We believe that using a hexhedral domain combined with one of the more complicated separating surface algorithms [12, 30, 27, 2] is likely to reduce the computational cost of these collision tests, since this could allow for fast triangle-to-voxel rasterization to be used for collision detection, similar to permission grids [31]. Further research into this area could improve performance significantly. Additionally, even though few of the other surface reconstruction methods are designed to handle the datasets we consider here, a critical comparison between our method and others is needed, using datasets with fewer regions, in order to determine where our method fits in the rich taxonomy of surface reconstruction methods.

## 7 Conclusion

We have described a new method for extracting smooth, simplified separating surfaces from segmented cross-sections. The first step of our method interpolates segmentation boundaries between cross-sections using a constrained Potts model. The second step is a voxel-accurate mesh simplification algorithm that both reduces the excessive triangle count of marching-tetrahedra and smooths artifacts that result from the underlying hexahedral mesh. Both of these goals are achieved by attempting to find the simplest triangle surface that separates segmented regions. Our boundary interpolation and surface simplification methods are novel contributions to the state of the art in three-dimensional image data processing and analysis. Our boundary interpolation method contributes to the study of surface reconstruction from planar contours, especially in the case of surfaces which partition 3D space into a large number of small regions. The results of our simplification algorithm make it clear that marching-based surface extraction for segmented volumes produces an over-abundance of triangles which are not necessary to correctly represent the separating surface.

We have applied our method to construct the crystal grain boundary surface of a serial-sectioned sample of the metal tantalum. The sample is dense with surface geometry, containing over 19,000 segmented regions. Our boundary interpolation accurately tracks grain boundaries between sparse segmented slices, and our simplification algorithm reduces the number of triangles in the initial separating surface by a factor of ten without introducing any error. Our methods can be applied to other imaging tasks which involve tracking hundreds or thousands of features using sparse segmentation information, which is the case in cellular microscopy and other biomedical imaging applications.

## Acknowledgements

The authors wish to thank Taylor Holliday, Sung Park and John C. Anderson for their help and input. This work was supported by the Materials Design Institute, a UC Davis/Los Alamos National Laboratory educational research collaboration (subcontract # 25110-002-06).

## References

[1] B. Adams, S. Wright, and K. Kunze. Orientation imaging: the emergence of a new microscopy. *Metall. Trans. A*, 24A(4):819–831, 1993.

[2] M. Bertram, G. Reis, R. H. van Lengen, S. Köhn, and H. Hagen. Non-manifold Mesh Extraction from Time-varying Segmented Volumes used for Modeling a Human Heart. In K. Brodlie, D. Duke, and K. Joy, editors, *Eurographics / IEEE VGTC Symposium on Visualization*, pages 199–206. Eurographics Association, 2005.

[3] S. Bischoff and L. Kobbelt. *Extracting Consistent and Manifold Interfaces from Multi-valued Volume Data Sets*, pages 281–285. Springer Berlin Heidelberg, 2006.

[4] J.-D. Boissonnat. Shape reconstruction from planar cross sections. *Comp. Vision Graph. Image Process.*, 44(1):1–29, 1988.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[6] I. Braude, J. Marker, K. Museth, J. Nissanov, and D. Breen. Contour-based surface reconstruction using mpu implicit models. *Graphical Models*, 69(2):139–157, 2007.

[7] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proc. SIGGRAPH*, pages 119–128. ACM Press, 1996.

[8] T. Dey, H. Edelsbrunner, H. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd)*, 66:23–45, 1999.

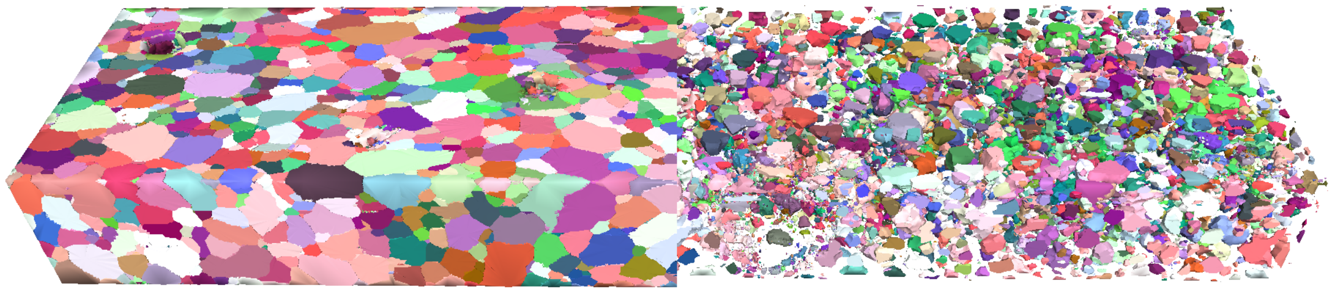


Figure 9: A sample of tantalum with 19,000 grains. On the right, each grain has been scaled down 50% about its centroid. The initial mesh contained 23,700,000 triangles, and the simplified mesh contains 2,300,000 triangles.

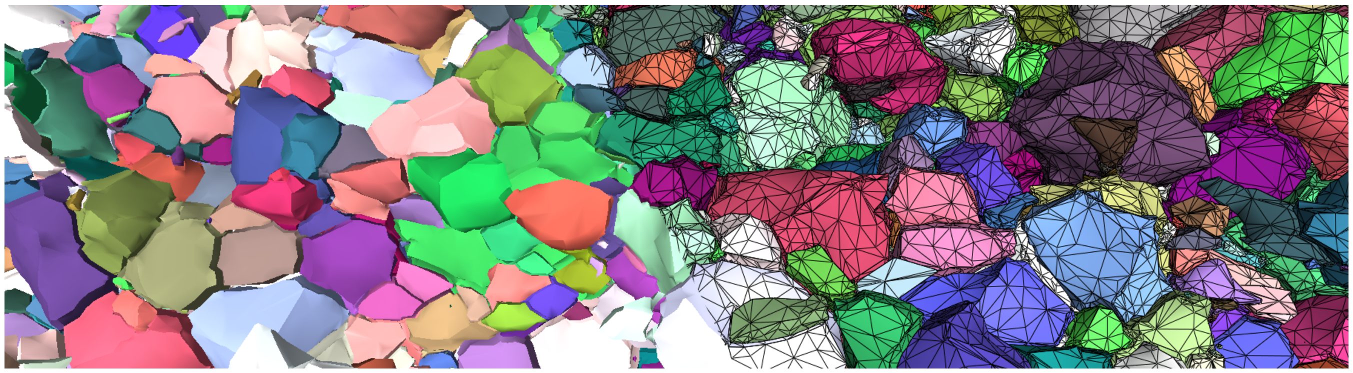


Figure 10: Shaded and wireframe views of the constructed boundary surface in the interior of the sample. Grains have been scaled down slightly to emphasize their edges and vertices.

[9] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20(10):693–702, 1977.

[10] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH*, pages 209–216. ACM Press, 1997.

[11] A. Guéziec. Locally toleranced surface simplification. *IEEE Trans. Visualization and Comp. Grap.*, 5(2):168–189, 1999.

[12] H.-C. Hege, M. Seebass, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on nonbinary classifications. Technical Report SC 97-05, Zuse Institute Berlin, 1997.

[13] E. A. Holm and C. C. Battaile. The computer simulation of microstructural evolution. *Journal of The Minerals, Metals and Materials Society (JOM)*, 53(9):20–23, 2001.

[14] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proc. SIGGRAPH*, pages 19–26. ACM Press, 1993.

[15] M. W. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):75–84, 1994.

[16] A. D. Kalvin and R. H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Comp. Graph. Appl.*, 16(3):64–77, 1996.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[18] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proc. SIGGRAPH*, pages 105–114. ACM Press, 1998.

[19] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. SIGGRAPH*, pages 163–169. ACM Press, 1987.

[20] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comp. Graph. Appl.*, 21(3):24–35, 2001.

[21] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Computational Geometry*, 5(2):95–114, 1995.

[22] H. Müller. Boundary extraction for rasterized motion planning. Technical Report 566, University of Dortmund, 1995.

[23] L. Neumann, B. Csébfalvi, I. Viola, M. Mlejnek, and E. Gröller. Feature-preserving volume filtering. In *Proc. VISSYM*, pages 105–ff. Eurographics Association, 2002.

[24] G. M. Nielson and R. Franke. Computing the separating surface for segmented data. In *Proc. Visualization*, pages 229–233. IEEE Computer Society Press, 1997.

[25] G. M. Nielson, G. Graf, R. Holmes, A. Huang, and M. Phielipp. Shrouds: optimal separating surfaces for enumerated volumes. In *Proc. VISSYM*, pages 75–84. Eurographics Association, 2003.

[26] O. Nilsson, D. Breen, and K. Museth. Surface reconstruction via contour metamorphosis: An eulerian approach with lagrangian particle tracking. In *Proc. Visualization*, pages 407–414. IEEE Computer Society Press, 2005.

[27] B. Reitinger, A. Bornik, and R. Beichel. Constructing smooth non-manifold meshes of multi-labeled volumetric datasets. In *Proc. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2005*, pages 227–234. University of West Bohemia, 2005.

[28] D. Weinstein. Scanline surfacing: building separating surfaces from planar contours. In *Proc. Visualization*, pages 283–289. IEEE Computer Society Press, 2000.

[29] R. T. Whitaker. Reducing aliasing artifacts in iso-surfaces of binary volumes. In *Proc. IEEE Symposium on Volume Visualization*, pages 23–32. ACM Press, 2000.

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

[31] S. Zelinka and M. Garland. Permission grids: practical, error-bounded simplification. *ACM Trans. Graph.*, 21(2):207–229, 2002.