Lattice Cleaving: A Multimaterial Tetrahedral Meshing Algorithm with Guarantees

Jonathan Bronson, *Student Member*, *IEEE*, Joshua A. Levine, *Member*, *IEEE*, and Ross Whitaker, *Senior Member*, *IEEE*

Abstract—We introduce a new algorithm for generating tetrahedral meshes that conform to physical boundaries in volumetric domains consisting of multiple materials. The proposed method allows for an arbitrary number of materials, produces high-quality tetrahedral meshes with upper and lower bounds on dihedral angles, and guarantees geometric fidelity. Moreover, the method is combinatoric so its implementation enables rapid mesh construction. These meshes are structured in a way that also allows grading, to reduce element counts in regions of homogeneity. Additionally, we provide proofs showing that both element quality and geometric fidelity are bounded using this approach.

Index Terms—Tetrahedral meshing, multimaterial, multilabel, biomedical, conformal meshing, watertight, mesh quality, adaptive meshing, guaranteed meshing

1 Introduction

The finite-element method (FEM) is ubiquitous in the field of scientific computing when employing partial differential equations on complicated domains. Its combination of flexibility and numerical consistency makes it the method of choice for simulations across a wide range of physical phenomena including electromagnetics, fluid dynamics, and solid mechanics. FEM relies on a decomposition of a domain into a union of discrete elements, in the form of a mesh. These elements conform to important geometries in the domain, such as the interfaces between materials or boundary conditions. While FEM allows for a wide range of grid types and topologies, in practice many implementations use tetrahedral domain decompositions because they offer a good compromise between simplicity of mesh generation, generality, ability to conform to complex geometries, and numerics.

With FEM, the solutions of PDEs are associated with a linear system induced by the operator and the boundary conditions. The approximated differential operator depends on the mesh elements, and the shapes of these elements impact the structure of this matrix—most importantly its condition number [7]. The condition number of the linear system, which is usually quite large, in turn controls the speed and/or accuracy of the numerical solution to the linear system. Thus, a second important requirement of the underlying mesh is the *quality* of the underlying elements. The dual requirements of meshes that conform to geometry and meshes that have good quality elements are often in conflict. Thus, meshing algorithms must make tradeoffs between quality and geometric fidelity.

• *The authors are with the School of Computing, University of Utah, 3893 Warnock Engineering Building, Salt Lake City, Utah 84112-9205.*

Manuscript received 27 Sept. 2012; revised 17 Apr. 2013; accepted 28 July 2013; published online 13 Aug. 2013.
Recommended for acceptance by B. Levy.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-09-0211.
Digital Object Identifier no. 10.1109/TVCG.2013.115.

These mesh requirements also interact with the specific nature of the geometric constraints and the mechanisms by which they are specified. In this work, we consider FEM simulation problems that specify materials *volumetrically*. That is, the physical materials are given by functions on the domain that evaluate to the appropriate material at a given location, and material interfaces are where these functions transition from one value to another. This volumetric specification is natural in biomedical simulations based on images [37], where material boundaries are derived from segmentations or labels, as well as simulations that rely on implicit representations of physical interfaces [14]. In this paper, we specifically address tetrahedral meshing with *multimaterial* interfaces, where the geometric constraints are nonmanifold structures with higher-order junctions of three and four materials.

Contributions. In this paper, we describe a new meshing algorithm, *lattice cleaving*, for generating tetrahedral meshes for multimaterial domains that are specified as a collection of continuous indicator functions. The output meshes conform approximately to interfaces between materials, including nonmanifold regions, where multiple materials meet. The output meshes have tetrahedral elements with provably bounded dihedral angles as well as a guaranteed fidelity of sufficiently large features. Lattice cleaving relies on a regular background lattice, with a resolution determined by the user, which is subdivided or *cleaved* to conform to material boundaries. For each cleaved background tetrahedron, it applies and modifies a stencil, used to approximate the geometry while not destroying the good quality of elements in the background lattice. Lattice cleaving requires a small, fixed number of passes through the background grid and, therefore, leads to reliably fast runtimes. Results on biomedical volumes and fluid simulations demonstrate the algorithm reliably achieves fast runtimes, geometric fidelity, and good quality elements. This paper builds on a preliminary publication of the material [8].

2 Related Work

The literature on unstructured 3D mesh generation is vast, partly as a byproduct of the wide utility of these meshes to application areas in science and engineering. Here, we divide the discussion along the two major constraints on this meshing problem: 1) producing high-quality elements and 2) conforming to complex surfaces. In addition, we review lattice-based meshing algorithms which use, in part, similar techniques to our own.

2.1 Boundary Conforming Mesh Generation

In the past decade, a significant amount of effort has gone into building high-quality surface meshes [21], particularly through Delaunay triangulations [10]. One of the most popular strategies relies on Delaunay refinement [16], [39], which iteratively inserts sample points on the domain boundary until conditions are met for sufficiently capturing both the topology and geometry of surfaces. These surface meshes are typically inputs for conformal tetrahedral meshing algorithms, with further refinements of the volumetric regions. Boissonnat and Oudot [5] and Cheng et al. [13] pioneered the first variants on provable algorithms for performing Delaunay refinement that capture the topology of smooth, surface-boundary constraints. Extending these ideas to more complex, piecewise-smooth, and nonmanifold domains followed [12]. However, these algorithms rely on various strategies for *protecting* features on the material boundaries, and the implementations of these schemes are challenging. Thus, simplifying assumptions are required in the protection scheme to make them practical [6], [18], [38].

The local, greedy strategy of Delaunay refinement schemes tend to find suboptimal configurations for vertices. Variational meshing schemes attempt to overcome this limitation by positioning vertices according to some global energy function. These strategies typically decouple, to various degrees, the vertex placement problem from the triangulation/tetrahedralization problem. Meyer et al. [33] use a variational scheme similar with repulsion between particles (points) to sample multimaterial interfaces and then connect these samples using a Delaunay triangulation. Bronson et al. [9] build on this formulation to build highly adaptive surface meshes for CAD geometries, but do not require expensive precomputations. Yan et al. [49] use an energy formulation based on centroidal Voronoi tessellations to drive particle movements. Tournois et al. [45] alternate between Delaunay refinement insertions and vertex optimizations for high-quality meshes for non-smooth shapes. These types of optimizations are nonlinear and require multiple iterations on gradient-descent-based strategies to find local minima. Thus, they are time consuming, are sensitive to initializations and parameter tuning, and do not provide typical criteria to establish guarantees on the quality of the output.

While these works represent only a taste of the most recent work in boundary-constraint meshing, they have a number of interesting shared characteristics. First, each algorithm requires at least one expensive computation, either a Delaunay triangulation in 3D, or a vertex optimization, or both (each of which necessitates an $O(n^2)$ computation), while the variational schemes may require multiple iterations on these computations. Second, these algorithms all decouple the surface constraint of boundary conforming from the volumetric quality constraint, by requiring that the algorithm *first* construct a mesh of the boundary and then next construct a volume mesh given this surface mesh as input (*surface first, then volume*). As a result, they often deal with very poor quality mesh elements, in particular *slivers* or tetrahedra with nearly cocircular vertices [11], [45]. In contrast, the proposed method follows the strategy of *volume first, then surface*, which has the benefit of being more flexible in how we manage the inevitable compromises between geometry and quality.

2.2 Tetrahedral Element Quality

While a large number of measurements and ratios are used to judge the quality of elements in meshes, in this work we focus on isotropic measures of quality applicable to linear finite elements [41]. These qualities, while somewhat generic, have the advantage of being numerically useful for the large class of elliptic operators that appear in FEM simulations of many physical phenomena, such as incompressible flows, diffusion, and electric fields. While there are many reasonable measures of tetrahedral mesh quality, we rely on the worst-case dihedral angles (both minimum and maximum) over all tetrahedra. The distance of dihedral angles from 0 and 180 degrees correlates with most other common element quality measures.

Measures of quality are typically independent of element size. Adaptivity of mesh size/resolution provides another key ingredient in the definition of element quality. Both Delaunay refinement [40] as well as variational schemes [1] have been used to improve and adapt volumetric element quality, as well as more greedy optimizations driven by local mesh improvements [20], [28]. Moreover, isotropic element quality is also indifferent to element orientations; i.e., it penalizes anisotropy in all directions equally. The proposed work provides graded meshes with smaller elements near surfaces, but does not address the problem of adaptivity and anisotropy directly.

2.3 Lattice-Based Meshing Approaches

A very common strategy for building meshes is to start with a high-quality (e.g., regular) background mesh and modify it to adhere to geometric constraints. However, the problem of making a regular lattice conform to an arbitrary surface presents some challenges, especially when tetrahedral quality is a concern. One strategy is to cut (or cleave) the cells of the input lattice to match the surface, an idea popularized by the well-known marching cubes algorithm for isosurfacing [31], and the related dual contouring method [27]. Constrained delaunay triangulation [15], [42] can then be applied to generate volume-filling tetrahedra [51]. The different configurations of surface/cell intersections are typically represented by *stencils* with the appropriate topology. Several authors propose surface reconstruction with a piecewise linear approximation of surfaces as they cut through the tetrahedra of a body-centered cubic (BCC) lattice [4], [35], with extensions to nonmanifold surfaces using a collection of indicator functions (instead of the single scalar field for isosurfacing). These algorithms examine indicator functions locally at each vertex of the mesh element. Depending on which

indicator is maximal, they next label each vertex with a material, a generalization of inside/outside for isosurfacing.

Working with lattices has advantages beyond just surfacing. For instance, an octree can be used on a regular lattice to facilitate adaptively sized elements [50]. Zhang et al. [52] use a regular lattice and encodings to achieve surface simplification while preserving topological features. This work was extended to generate tetrahedral as well as hexahedral elements [53] and within multimaterial domains [55].

Another strategy for conforming is to warp a background lattice so that primitives align with boundaries [22]. Molino et al. [34] use a BCC lattice coupled with a red-green subdivision strategy, which they then optimize to conform to the surface. That work empirically achieves good quality tetrahedra, albeit with no proof of bounds. Labelle and Shewchuk [29] propose a combination of lattice warping and stenciling, with appropriate rules that decide which combination of strategies to use, based on the input data, to ensure good quality. They describe a computer-assisted proof to compute quality bounds for their *isosurface stuffing* algorithm. Wang and Yu [47] employ a similar approach. The proposed method shares several aspects of the Labelle and Shewchuk method. Like their algorithm, we cut a BCC lattice to conform to a boundary mesh, and like their algorithm we rely on a threshold α to locally warp the lattice to remove short edges and maintain high-quality elements. However, instead of considering a single smooth isosurface, the multimaterial boundary constraints present nonsmooth and nonmanifold structures. This adds considerable complexity to the algorithm, which in the past has only been approached using additional levels of subdivision, such as in Liu et al. [30] or dual-contouring [54]. Here, we show instead that a carefully designed stencil set combined with appropriate rules for application provides quality guarantees for the resulting tetrahedra.

3 Methodology

The proposed tetrahedral meshing algorithm operates on a collection of indicator functions. We sample these functions onto a BCC lattice. Similar to many surfacing and meshing algorithms [23], [31], we rely on a set of stencils that capture local material configurations. We use the strategy of Labelle and Shewchuck [29] to construct a set of rules for each background BCC lattice tetrahedron that switch between two cleaving modes—either deforming the background BCC lattice or splitting the background tetrahedra to conform to boundary surfaces.

Within this context, the multimaterial meshing problem presents several important challenges. Unlike the isosurface case, one cannot easily restrict the size of features because feature size [1] will always go to zero, where three or more materials meet. The complexity within each lattice cell is also challenging. Considering only the material labels at vertices, the number of cases is daunting. Furthermore, even if one represents indicator functions along lattice edges as linear, the number of possible interfaces passing through a single edge grows with the number of materials, regardless of the conditions at the vertices. Therefore, geometric and topological approximations are essential.

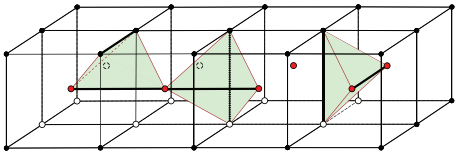


Fig. 1. The BCC lattice is composed of two grids of primal and dual vertices [29]. Each vertex is incident to 14 edges, 36 faces, and 24 tetrahedra.

3.1 Indicator Functions

There are many papers on extensions of implicit surfaces or level sets to multimaterial interfaces. Here, we represent multimaterial interfaces using a set of K-smooth, volumetric *indicator functions*, $F = \{f_i | f_i : V \mapsto \Re\}$ [32], [37]. A material label i is assigned to a point $\boldsymbol{x} \in V$ if (and only if) $f_i(\boldsymbol{x}) > f_j(\boldsymbol{x}) \; \forall \; j \neq i$. For any single material j, a continuous, inside-outside function can be constructed as $\tilde{f}_j(\boldsymbol{x}) = f_j(\boldsymbol{x}) - \min_{i \neq j}(f_i(\boldsymbol{x}))$, and the zero functions of various materials will coincide at shared boundaries.

3.2 Background Lattice and Material Interfaces

Stenciling algorithms rely on a set of regular cells. The configuration of the interfaces on these cells is used to generate an index that corresponds to some predefined tessellation. We employ a BCC lattice (Fig. 1), where each cell is composed of eight normal or *primal* cubic lattice vertices, plus a ninth *dual* vertex in the center. In addition to the 12 edges of a regular cubic cell, there are eight diagonal edges connecting each dual vertex to its cell's primal vertices, and six connecting dual to dual. Fanning out from the dual vertex are 24 lattice tetrahedra, each of which spans two lattice cells. Each lattice tetrahedron is identical in shape, as is each lattice face. The edges connecting both primal and dual vertices differ in length than the edges connecting only primal or only dual vertices. We refer to these edges as long and short, respectively.

For stencils to be applicable, decisions about the structure of each cell must be strictly local and enumerable a priori. Our strategy for mapping data onto the lattice entails several approximations. Initially, each lattice vertex represents a single material at that point, which is given by the indicator function with maximum value. If two or more functions are comaximal, the tie is settled by a very small push away from a prioritized (or random) material. Any lattice edge that contains vertices with two different labels contains a material transition, called a *cut-point*, or *cut* [29]. These edge-cuts sample a surface separating two materials.

A similar logic applies to junctions of more than two materials. A lattice face with three unique material labels on its vertices must have an associated transition point where all three materials meet. We refer to this point as a *triple-point* (*triple*). The collection of triple-points in the domain define curves where three materials meet. A lattice tetrahedron may have up to four unique material labels. The four vertices, and the four function values associated with the material labels on each vertex, define a single, isolated, material transition point. We refer to this point as a *quadruple-point* (*quadruple*).

We restrict the number of material transitions defined on a tetrahedron. Each lattice simplex may contain at most a

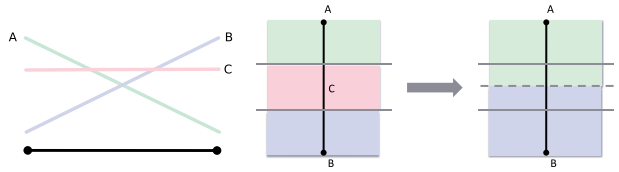


Fig. 2. An edge with materials a and b maximum on its endpoints, but with a third material c becoming maximum on the interval between.

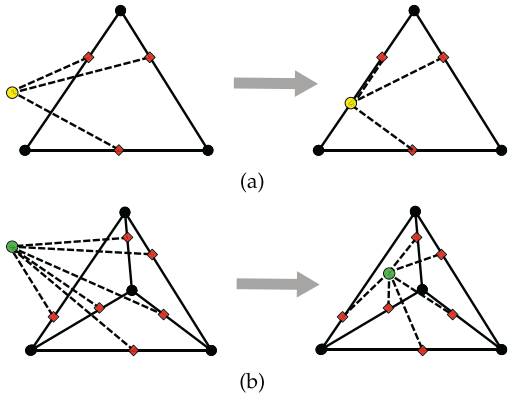


Fig. 3. (a) Triples (b) and quadruples are forced to lie within the primitive that contains the associated edge-cuts.

single transition point matching its dimensionality: an edge may have only a single cut, each face a single triple, and each tetrahedron a single quadruple. These approximations are the multimaterial generalizations of the approximation that underlies stencil-based isosurface algorithms, which ignore features that pass between vertices. Fig. 2 illustrates how such a situation might manifest on an edge. These various material interfaces are defined as the points where the values of indicator functions of the materials on the vertices are equal, and, in general, we assume these locations are given by an *oracle*. In the absence of an oracle, these points can instead be computed, for instance, by a system of linear equations.

For an edge, these transitions lie on the line segment connecting the two vertices. However, for triple or quadruple-points, they could lie outside of the corresponding triangular face or tetrahedron, respectively. In such cases, these points are projected back onto the tetrahedron, so that local stencils can apply (Fig. 3). These approximation lead to a smoothing or removal of thin features that fall below the resolution of the grid—i.e., the exceptions to the above conditions are indicative of features that fit between grid points, as proved in Section 4.1.

3.3 Stencil Keys

Selection of the appropriate stencil for any given cell is determined by a key. For the single isosurface case, this key is an ordered concatenation of labels for each vertex of the simplex being stenciled. These labels indicate whether a vertex is inside (-1), outside ($+1$), or directly on the isosurface (0). If we were to restrict vertex labels to be strictly inside and outside, this leads to 16 configurations, with only eight unique valid cases. Allowing a vertex to sit on the isosurface would lead to 81 possible configurations. In the multimaterial case, this approach to indexing stencils is problematic for several reasons. First, inside of one material is outside of another. This means a label for each material in the volume is needed. The length of the key will need to grow to accommodate data sets with larger numbers of materials. Consequently, the number of cases of stencils would need to fill the space of cases, with many duplicates for similar topologies with differing materials.

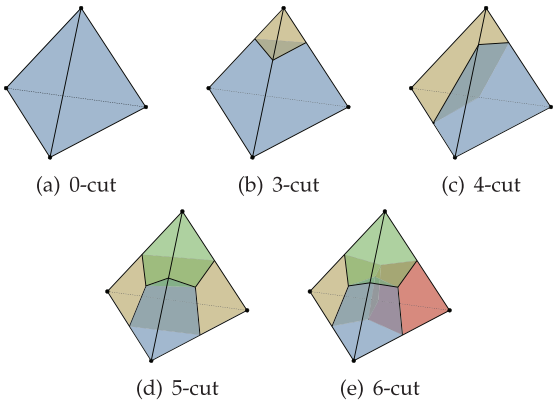


Fig. 4. The five unique interface topologies determined by the number of cut-points present on a lattice tetrahedron.

One approach to solving this issue is to instead store bits only signifying the number of materials found on a vertex. In this scenario, the case table becomes tractible, but distinguishing element materials during stenciling becomes more difficult. This task requires expensive bookkeeping and may still lead to ambiguous material regions without explicit rules to avoid them.

The solution in the proposed approach bypasses this problem by only defining stencils on lattice tetrahedra with precisely one material per vertex. Intermediate topologies, where multiple materials reside on a single vertex, are generated by various edge collapses on the original stencil. In this way, the challenges of exhaustive case tables and ambiguous regions are avoided. The key for a generic set of stencils in this multimaterial case would be a 6-bit key, each bit indicating whether a particular edge of the lattice tetrahedra contains an edge-cut. This leads to only 64 possible configurations, with many cases being invalid. In Section 3.6, we show that by generalizing all stencils to a single case, we can avoid the need for a lookup table altogether.

3.4 Quality Criteria

Within a lattice tetrahedron, we approximate the material interfaces as a set of triangular facets that connect the various cut-points with the correct topology. With no additional processing of the mesh data, there are only five unique topological cases, distinguishable by the number of edges that contain a cut: 0, 3, 4, 5, or 6. It is impossible for a lattice tetrahedron to contain only one or two edge-cuts. As illustrated in Fig. 4, these cases are composed of three types of polyhedra: tetrahedra, triangular prisms, and hexahedra.

While these polyhedra admit multiple consistent tessellations, the output tetrahedra could become arbitrarily bad (regardless of the tessellation chosen) depending on where interface points are located. Thus, we define a set of *violation conditions* that characterize the configurations of interface points that lead to bad tetrahedra. These conditions are

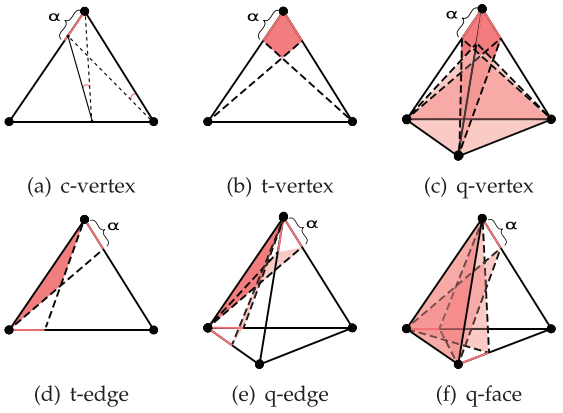


Fig. 5. An interface point violates a feature if it falls within an intersection of half-spaces defined using α. Vertices can be violated by (a) cuts, (b) triples, and (c) quads. Edges can be violated by (d) triples and (e) quadruples. Faces can only be violated by (f) quadruples.

used to decide when it is appropriate to warp the background lattice (changing topology) and when it is appropriate to leave a configuration intact. The conditions entail a threshold on the proximity between features, denoted α, and are expressed as a fraction of the edge lengths on the background lattice.

There are three different ways in which an interface might be *violating*. First, an interface point of any type may violate a lattice vertex. A cut violates a lattice vertex if it lies within a distance α to it along the shared edge. As shown in Fig. 5a, even in 2D, no matter how you choose to tessellate a face, there will always be an angle that is arbitrarily bad as the cut approaches the lattice vertex. This principle extends to interface points of higher order (i.e., triples and quadruples). Triple-points can move within the 2D space interior to a lattice face, and so their vertex violation region is a quadrilateral patch. This patch is formed by the intersection of two half-spaces. Each half-space is defined by connecting the point at distance α on one edge, to the opposite lattice vertex (Fig. 5b). Similarly, quadruple-points can be anywhere inside the lattice tetrahedron, so their vertex violation regions are formed by the intersection of three half-spaces defined by planes. (Fig. 5c).

The second group of violations pertain to edges. Degenerate tetrahedra can also arise if triple-points or quadruple-points lie too close to an edge. We define the notion of edge violation in a manner consistent with vertex violations, similarly bounding the angles. Dividing lines are formed between each vertex on the edge and the respective α position on the edge opposite that vertex (Figs. 5d and 5e). Finally, a quadruple-point has one additional violation condition, arising from its distance to adjacent faces. This violation region for faces follows the same logic as the others (Fig. 5f). Three planes are defined using the alpha parameters. Each plane contains two vertices on the face in question, and the alpha position on the edge incident to neither of these two vertices. The intersection of the half-spaces defined by these three planes forms the 3D violation space. With these violation rules in place, the next section provides a set of operations that can remedy any set of interface points that are in a violation condition.

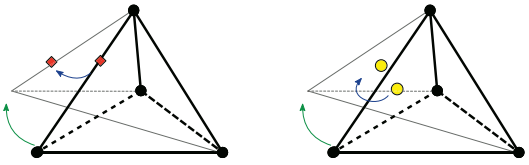


Fig. 6. When a vertex warps (green arrow), (a) cuts and (b) triples on incident faces must be updated (blue arrow) to reflect the their new locations on the surfaces.

3.5 Topological and Geometric Operations

The lattice cleaving algorithm uses two fundamental operations to ensure mesh quality. A *snap* operation merges an interface point with another point of lower order, collapsing the implicit edge between them in an output stencil. This operation is performed on interface points that are in violation, ensuring output stencil tetrahedra do not span bad angles. In conjunction, lattice vertices are *warped* spatially to conform to the interface surfaces.

The multimaterial cases introduce additional complexity into processing the snaps and warps. In the two-material case, Labelle and Shewchuk [29] warp a lattice vertex to a single violating cut and remove all adjacent cuts, effectively pulling them into the warped vertex. In the multimaterial case, this is unsatisfactory because the adjacent cuts could be interfaces to several different material sets.

If interfaces of different material sets are violating a lattice vertex, no single warp position can satisfy the surface constraints. Therefore, a number of strategies may be used to choose a suitable warp location. We use the center of mass of these violations because it is both easy to compute and distributes error proportionally among each interface. Alternative approaches might be to minimize a quadric error term for each surface, or to choose a preferred surface representative.

After snapping and warping a vertex to remove its violations, additional material interfaces may still be present on the incident edges and faces. Because these edges and faces will also be warped by the movement of the lattice vertex, the implicit surfaces may intersect them at new locations. Thus, we must update the locations of the remaining cuts and triples on incident edges appropriately, either by geometric intersection tests or via querying the oracle.

When an edge moves because one of its vertices is warped, any cut on that edge must move along the interface the cut represents. In practice, we use a linear approximation to the interface surface to perform this update. If the cut interface is of the same type as the violating cut that caused the warp, it naturally gets pulled into the snap like the 2-material case. If the new location of the cut is no longer on the edge (e.g., moves off of one of the ends), we bring it back onto the line segment at the appropriate end point. In this way, the stenciling operation remains local. We call this operation of recomputing the position of an interface on the warped lattice a *projection*, shown in Fig. 6. If the new cut position is violating, we perform a snap to the lattice vertex, and warp the vertex only if it has not already been warped previously. In this way, each lattice vertex undergoes at most one warp.

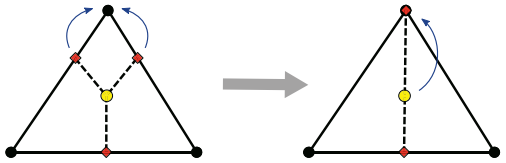


Fig. 7. Degenerate triples or quadruples are removed by subsequent snaps.

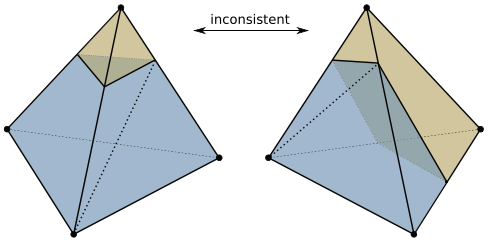


Fig. 8. Stencils for lattice tetrahedra must be consistent across faces. In this example, the blue quad patch shared between the tetrahedra is tesselated in two different ways.

If a lattice face moves because one of its vertices is warped, any triple on that face may also move. We update its location using the same strategy as with edges. If the triple leaves the face, we bring it back on, and follow-up with appropriate snaps and warps as needed. Quadruples need no projection unless a face moves in such a way that the quadruple falls outside of the new tetrahedron—in which case it will be moved onto the nearest edge/face and collocated with the corresponding cut/triple on that face.

This strict hierarchy of interface types raises another complexity unique to the multimaterial case. Snaps may cause material interfaces to degenerate such that they violate the hierarchy of interface types. For example, consider a face with a triple-point. If the cuts on two adjacent edges snap to the same lattice vertex, the triple-point is now representing only a 2-material interface, with a degenerate material region along the remaining line segment. To fix this degeneracy, the triple-point joins the two associated cuts at their warp destination, as in Fig. 7. A triple-point snap may also cause cuts to become degenerate, and a quadruple-point snap may cause cuts and triples to become degenerate. The number of these cases is quite limited, and each one is tested and corrected in a way that ensures a consistent hierarchy of features and a consistent mesh.

3.6 Generalized Stencil

After snapping and warping, the polyhedra from the topological cases described in Section 3.4 may have collapsed into intermediate topologies. Each such topology demands not only a valid tessellation stencil, but also one that does not permit degenerate tetrahedra when interface points are in nonviolating configurations. Moreover, each such stencil must be consistent both within the lattice tetrahedron, as well as across lattice faces (Fig. 8). One of the contributions of this paper is presenting a single *generalized stencil* that can be used as a master stencil for all achievable topologies. Not only does this keep the problem of stenciling local (avoiding inconsistency issues), but also it removes the requirement of implementing and storing a large stencil table that is prone to construction and transcription errors [19].

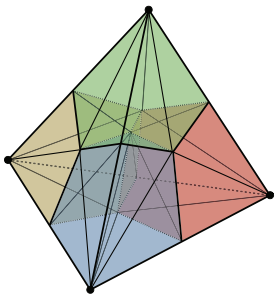


Fig. 9. The generalized stencil is constructed from the 6-cut case. Edges connect each interface point to its associated lower order features.

The generalized stencil is constructed from the most complicated topological case, the 6-cut case. An edge is formed between every pair of points that could be snapped, ensuring the snapping procedure of Section 3.5 always simplifies the topology in a manner equivalent to a series of edge collapses. On each lattice face, edges star out from the triple-point to every edge-cut and vertex on the boundary of the face. Similarly, on the interior of the lattice tetrahedron, edges star out from the quadruple-point to every triple-point, edge-cut, and vertex on the boundary of the lattice tetrahedron (Fig. 9). In the regular case, this is equivalent to barycentric subdivision of the tetrahedron. This construction tessellates the lattice tetrahedron into 24 stencil tetrahedra, each composed of a single vertex, cut, triple, and quadruple.

For every lattice tetrahedron that does not have this full complexity of material interfaces (6-cuts), we choose vertices, cuts or triples, to have *virtual material boundaries*, as if they had already snapped. This allows us a consistent way to tetrahedralize the polyhedra in the multimaterial stencils without worrying about inconsistencies or tangles across faces between stencils, keeping the stenciling operation local.

The procedure to generalize a lattice tetrahedron to the 6-cut case proceeds as follows: First, virtual edge-cuts are created for any edge missing a cut. The remaining virtual points all cascade into place from the location of these virtual cuts. For a face that has no triple-point, we label one of the three cuts as a virtual triple-point. To maintain valid topologies, we always choose a cut that lies on an edge that already contains a virtual cut. If there is no virtual cut, a predetermined cut is chosen. Finally, if a quadruple-point is not present, we must choose a triple-point location to represent the virtual quadruple-point. We pick a triple-point using the same method a triple uses to pick a cut-point. If there are multiple options, we choose the point that is collocated with the most other points, virtual or real.

Note that these rules for generalizing lattice tetrahedra operate once; they are merely the mechanism for generating a consistent set of stencils. These generalizations can even be computed offline, for all stencil cases, and then chosen from a lookup table at runtime. These rules for choosing arbitrary, but consistent, transitions from the 6-cut case to all of the others produce different (but valid and good quality) tetrahedralizations of the similar cut patterns,

depending on their orientation on the BCC lattice. This mechanism for ensuring consistency is a multimaterial alternative to the *parity* scheme used in the two-material case [29]. This procedure as described is merely one solution within a space of valid possibilities.

3.7 Grading

Like other meshing algorithms built upon the BCC lattice, we take advantage of its structure to achieve graded meshes through the use of an octree. Any cell that contains at least one edge-cut is created as leaf in this tree. We utilize the same stencil as found in Isosurface Stuffing, where bisected and quadrisected BCC tetrahedra are used to span neighboring octree cells that differ in height by one level. Since these graded tetrahedra only appear in regions of homegeneity, the introduction of multiple materials has no effect on their structure.

3.8 Algorithm

The full lattice cleaving algorithm utilizes the rules developed in this section and proceeds as follows: Using an octree structure to reduce storage and allow grading, we first sample and label each BCC lattice vertex. Alternatively, a search strategy can be employed at this point to reduce storage and time searching for interfaces. If a tie occurs, we add an epsilon adjustment to a random material, creating cuts one or more incident edges. Next, cuts, triples, and quadruples are computed for each lattice tetrahedron that has multiple unique material labels on its vertices. The octree stores leaves only for the cells containing cuts, and it is balanced such that neighboring cells never differ by more than a height of one.

Any lattice tetrahedron that is not the 6-cut case is then generalized to be so, by labeling the locations of virtual cuts, triples, and quadruples. This operation completes in a single pass over the lattice tetrahedra, either directly or through the use of a precomputed lookup table. Once all lattice tetrahedra have been generalized, three phases of snapping and warping begin.

In the first phase, all violated lattice vertices are identified and visited exactly once. Any violating interfaces on incident edges, faces, or tetrahedra are snapped to the vertex, and the vertex warps to the center of mass of the interfaces, distributing any round off equally. All adjacent nonviolating interface points are projected to remain on their respective simplices. If an oracle is available, the new interface points should be queried directly. All degeneracies are fixed with additional snaps, as described in Section 3.5. After completion of this phase, all lattice vertices will be free from violations.

In the second phase, all violated lattice edges are identified and visited exactly once. If one or more triples or quadruples violate an edge, we snap them to the cut on that edge, wherever it may be. Sometimes a triple-point violates an edge that no longer contains a cut because it has already snapped to a lattice vertex. In such cases, the triple-point would snap to that lattice vertex as well. These snaps are designed so that a lattice edge may contain singular points of transitions or be a single material across its entirety; we do not allow material interfaces to lie on the half-edge. An alternative approach would be to project these triple-points to the nearest location on the edge, and split the edge with a new cut, tessellating the output stencil tetrahedra incident to that edge. This would provide increased fidelity but also increased element count.

In the final phase, we address the problem of quadruple-points that are too close to lattice faces. Using the face violation condition, we snap any such quadruple-point to the triple-point on the face that was violated. Similar to the second phase, sometimes a quadruple-point violates a lattice face that no longer contains a triple-point. It may have snapped to an edge-cut, or to a vertex. The quadruple-point always follows the triple-point, maintaining the hierarchy of features on each edge and face. Again, a splitting procedure could be utilized to retain higher fidelity.

Finally, we output all stencil tetrahedra that contain four unique vertices, skipping over any that were collapsed during the warping and snapping process. Octree cells at higher levels are also filled with appropriate tetrahedra for grading, though there is no need to delay this step until the end.

The complexity of this algorithm is worst case $O(n)$, where n is the number of voxels in the input image. However, in practice, it is rare that a set of interface surfaces would fill the entire volume. We find the complexity is most often sublinear. Additionally, while we implemented this algorithm as an in-core solution, requiring memory for the whole volume and mesh throughout the algorithm, this need not be the case. All of the operations act locally on sets of adjacent lattice cells. Therefore a streaming solution, or moving window approach could also be employed to process lattice cleaving. The benefits of such an approach would include reduced memory footprint, and possibly better cache performance.

4 Bounded Dihedral Angles

The algorithm described in Section 3 is designed to ensure that both dihedral angles are bounded and geometric distortion of input surfaces is controlled. In what follows, we prove these properties hold true.

The violation rules defined in Section 3.4 disallow vertex positions that could lead to undesirable tetrahedra. These proofs rely on these carefully designed rules for vertex placement, a particular set of properties in the generalized stencil set, and their interaction with the background lattice.

There are multiple ways to classify types of bad tetrahedron [2], [11]. One useful partitioning groups such tetrahedra into two sets: tetrahedra whose vertices are nearly collinear, and tetrahedra whose vertices are nearly coplanar (Fig. 10). This classification includes not only tetrahedra with bad dihedral angles, but also tetrahedra with bad solid angles (i.e., the spire).

It is also useful to classify the types of bad triangular faces that can occur on these undesirable tetrahedra, namely, *daggers* and *blades*. These triangles have vertices that are nearly collinear. While a tetrahedron may still be badly shaped without their presence, (e.g., slivers), a tetrahedron that contains poor quality triangles will itself also be of poor quality.

The rules comprising our algorithm make it impossible for output tetrahedra to become badly shaped (and consequently, they have bounded dihedral angles). First,

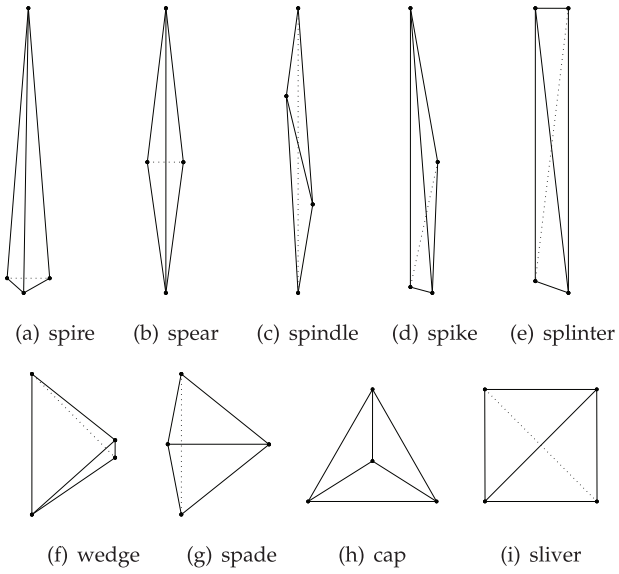


Fig. 10. Types of undesirable tetrahedra: (a-e) vertices are nearly collinear. (f-i) Vertices are nearly coplanar.

we show that background lattice tetrahedra stay of good quality. This property induces constraints on the polyhedra of our output stencils. Finally, we show that these constraints, combined with our violation conditions for warping and snapping, always lead to tetrahedra with bounded dihedral angles. Unlike the computational proof of Labelle and Shewchuk, which relies on interval arithmetic and a numerical search, this approach does not give a specific angle bound. The multimaterial problem introduces enough additional degrees of freedom to make a similar computational approach currently intractible. This direct proof does, however, provide insights into why this algorithm is successful at achieving bounded dihedral angles and gives a foundation for modifications and extensions. We begin by introducing several definitions.

Definition 1. *A dihedral angle θ is the angle between two planes.*

A tetrahedron contains six internal dihedral angles. The dihedral angle between triangular faces can be expressed as a function $\Phi : V^2 \mapsto \Re$ of the face unit normals $\hat{n}_1$ and $\hat{n}_2$:

$$\Phi = arccos(\hat{n}_1 \cdot \hat{n}_2). \tag{1}$$

Definition 2. *The aspect ratio,* $\mathrm{ar}_f = \frac{h}{l}$*, for a triangular face, f, where h is the height of the shortest altitude and l is the length of the longest edge.*

Definition 3. *The aspect ratio,* $\mathrm{ar}_t = \frac{h}{l}$*, for a tetrahedron, t, where h the height of the shortest altitude and l is the length of the longest edge.*

For triangles, the aspect ratio goes to zero as the vertices approach collinearity. For tetrahedra, aspect ratio is a measure for how close the vertices of a tetrahedron are to being either collinear or coplanar. It turns out that when tetrahedra degenerate in these ways, it must be the case that there are either dihedral angles of 0, 180 degrees, or both. All unwarped BCC lattice tetrahedra have aspect ratios $\mathrm{ar_t} = 0.866025$, and dihedral angles of 60 and 90 degrees. We next define the notion of ϵ-good.

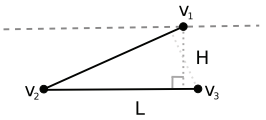


Fig. 11. The space of triangles with aspect ratio H/L. v_1 is restricted to move along the axis parallel to edge v_2v_3 such that L and H do not change. This 1D space is bounded on both sides.

Fig. 12. (Left) Three-dimensional dihedral angle (Right) projected to 2D with altitude.

Definition 4. *For $\epsilon > 0$, let $\theta_{\min}$ and $\theta_{\max}$ be the minimum and maximum dihedral angles for all possible tetrahedra with $\mathrm{ar}_t > \epsilon$. A dihedral angle, θ, is called ϵ-good if and only if $\theta_{\min} \leq \theta \leq \theta_{\max}$. Similarly, let $\phi_{\min}$ and $\phi_{\max}$ be the minimum and maximum interior angles, respectively, of all triangles with $\mathrm{ar}_f > \epsilon$. We call a planar angle, φ, ϵ-good if and only if $\phi_{\min} \leq \phi \leq \phi_{\max}$.*

Lemma 1. *For a triangle, t, with minimum and maximum interior angles $\phi_{\min}$ and $\phi_{\max}$, $\mathrm{ar}_f > 0$ iff there exists $\kappa > 0$ such that $\kappa < \phi_{\min}$ and $\phi_{\max} < 180° - \kappa$.*

Proof. Let t be a triangle composed of vertices v_1, v_2, and v_3. Let L be the length of the longest edge, joining v_2 and v_3, and H be the height of the shortest altitude, incident to v_1. v_1 can be moved along the line parallel to edge v_2v_3, without changing the height of this altitude (Fig. 11). If $H > 0$, a planar angle ϕ can only approach 0 or 180 degrees as it moves infinitely in either direction. However, as v_1 moves in either direction, either edge v_1v_2 or v_1v_3 lengthens. Eventually, this length will become longer than L, and the aspect ratio will change. Therefore, the movement of v_1 is bounded to keep ar_f fixed. The worst minimum angle possible becomes $\phi_{\min} = \arcsin(\mathrm{ar_f})$, and the maximum $\phi_{\max} = 180° - \arcsin(\mathrm{ar_f})$. □

Lemma 2. *For a tetrahedron, t, with minimum and maximum dihedral angles $\theta_{\min}$ and $\theta_{\max}$, $\mathrm{ar}_t > 0$ iff there exists $\kappa > 0$ such that $\kappa < \theta_{\min}$ and $\theta_{\max} < 180 - \kappa$.*

Proof. The dihedral angle, θ, between two incident triangular faces may be computed using the 3D face normals, as in (1). There is an equivalent planar angle, ϕ, formed by projecting these faces along the axis coinciding with their shared edge (Fig. 12).

The lengths of the two line segments that form this angle in 2D are equivalent to the lengths of the two triangle altitudes. The distance of the projection of each vertex not incident to the shared edge, to the opposite face becomes the height of an altitude in 2D projective space. This gives an equivalent equation for both the planar and dihedral angle associated with these vertices:

$$\theta = \arcsin\left(\frac{h_t}{h_f}\right), \tag{2}$$

where h_t is the height of the tetrahedron altitude in 3D (and equivalently the height of the triangle altitude

in 2D), and h_f is the height of the altitude of the incident face (and equivalently the length of the incident edge in 2D). □

Because the altitudes are orthogonal to the shared edge, by the Triangle Inequality, the lengths of the two altitudes are bounded by the edge lengths of the two triangular faces. Let L be the longest edge of the two faces and H the shorter of the two 3D altitudes between them. It must be the case that $\arcsin(\frac{H}{L}) <= \arcsin(\frac{h_i}{l_i})$ for all i. Because this relationship holds for each pair of faces of a tetrahedron, t, it also must be the case that $\arcsin(\frac{H_{max}}{L_{min}}) <= \arcsin(\frac{h_i}{l_i})$ for all i, or: $arcsin(\mathrm{ar}_t) <= \theta <= 180 - arcsin(\mathrm{ar}_t)$ for all θ in t.

Lemma 3. *There exists a set of violation parameters α_{short} and α_{long} for which all BCC lattice tetrahedra maintain ϵ-good dihedral angles after warping as described in Section 3.5.*

Proof. Let t be a BCC lattice tetrahedron and r_α be the radius of a ball around each vertex. Each ball contains the possible set of points to which its vertex may warp, given the violation parameters α_{short} and α_{long} (the violation parameters for short and long edges, respectively). If $r_\alpha = 0$, no warping takes place, and t has aspect ratio $\mathrm{ar}_t = 0.866025$. Because the worst dihedral angle of a tetrahedron can be defined as a continuous function of vertex positions, by the *intermediate value theorem*, there must exist an r_α for which $\mathrm{ar}_t = \epsilon > 0$. Thus, by Definition 4, there must exist an α_{short} and α_{long} for which the lattice tetrahedra maintain ϵ-good dihedral angles after warping. □

Definition 5. *Let p be a polyhedron subdivided into a set of polyhedra S. A polyhedral face f from the set S is considered external if it is incident to ∂p.*

Lemma 4. *All stencil polyhedra with nonviolating vertices maintain ϵ-good dihedral angles around edges incident to at least one external face.*

Proof. For every dihedral angle of a stencil polyhedron that spans an edge incident to an external face, either one face or both faces are external. If both faces are external, then the dihedral angle equals that of the enclosing background polyhedron. By Lemma 3, we know this is an ϵ-good dihedral angle. If one face is internal and the other external, there is at least one vertex, v_i, on the internal face that is not incident to the external face. The only way for the dihedral angle to lose the ϵ-good property is by moving v_i arbitrarily close to the external face, its edges, or its vertices. In each case, a violation condition from Section 3.4 would be triggered making these impossible. □

Lemma 5. *All output tetrahedra span at least two stencil polyhedron faces that meet at ϵ-good dihedral angles.*

Proof. In general, any two faces may either both be external, one external and one internal, or both internal. If both are external, by Lemma 4 we know that the dihedral angle between these faces will remain ϵ-good. If one is external and the other internal, by the violation conditions of Section 3.4 we know these faces are ϵ-good. The case of two internal faces is prevented through stencil selection. There are five regular topological stencil cases before snaps. Snapping can only simplify polyhedra, never creating additional internal faces. Thus, if no tetrahedra span two internal faces of the polyhedra in the regular topological cases, it is also the case that no tetrahedra span two internal faces after edge collapses. Among the regular topological cases, only the 5-cut and 6-cut cases have the potential for multiple internal tetrahedron faces. The generalized stencil is designed specifically to avoid any tetrahedra spanning the faces that are not guarded by violation conditions. □

Lemma 6. *All stencil triangles maintain ϵ-good planar angles after snapping and warping, as described in Section 3.5.*

Proof. All output stencils are composed from only four types of vertices: (lattice) vertices, cuts, triples, and quadruples, abbreviated v, c, t, and q, respectively. The vertices of a stencil triangle can exist in three ways. Either all three vertices lie on the same lattice face, two vertices lie on the same lattice face, or all three lie on unique lattice faces.

If all three vertices lie on the same lattice face, the violation conditions for cuts and triples guard against such an aspect ratio. There are only three sets of points that can become collinear, and our stencils specifically preclude triangles spanning them: vcv, vtc, and ctc.

If two vertices lie on the same lattice face, the triangle's aspect ratio, ar_f, can only fall below ϵ if the third vertex is in violation of the face containing the other two. This includes the third vertex violating an edge of the lattice face containing the two vertices.

Finally, if all three vertices lie on unique lattice faces, the triangle's aspect ratio, ar_f can only fall below ϵ if all three vertices are violating the vertex incident to all three lattice faces. □

Lemma 7. *All output tetrahedron have ϵ-good dihedral angles.*

Proof. Let t be an output tetrahedron. By Lemma 5, t has at least two faces joined at an ϵ-good dihedral angle along an edge e. By Lemma 6, the triangles incident to edge e are ϵ-good. The existence of one good dihedral angle with two good faces incident to it implies t must have ϵ-good dihedral angles everywhere. □

Theorem 1. *There exists a dihedral angle, $\theta^* > 0°$, such that the dihedral angles of all tetrahedra are bounded from below by θ^* and above by $180° - \theta^*$.*

Proof. By Lemmas 7 and 2. □

This proof shows that the meshes from the lattice-cleaving algorithm never degenerate, in fact Lemma 6 also ensures we produce no bad solid angles (e.g., "spires"), despite they lacking bad dihedral angles. Moreover, in practice, with proper choice of α, this bound, θ^*, is significant, and empirical results in Section 5 corroborate this fact.

4.1 Geometric Fidelity

We next make a statement about the quality of the surface approximation. Let Σ be the *interface surface*, the complex of smooth surface patches where two materials meet, as well as the associated curves where three materials are coincident, and the points where four meet. Σ is a CW-complex, and geometrically behaves as a piecewise-smooth complex [12]. It also has a well-defined medial axis M_Σ that we

define as the closure of the set of points in $\mathbb{R}^3$ that have at least two closest points in Σ. Each point in M_Σ is the center of a ball that meets Σ only tangentially. Using the medial axis, we can quantify of the scale of features at each point $p \in \Sigma$. In particular, we define local feature size, $\mathrm{lfs} : \Sigma \to \mathbb{R}$, as the distance from each surface point to the medial axis. Local feature size is well studied in smooth surfaces [1]. In our setting, local feature size approaches zero near triple junctions, which meet nonsmoothly. Consequently, we define the set of *h-regular points*, $\Sigma_h = \{p \in \Sigma \mid \mathrm{lfs}(p) > h\}$ and restrict our claims to these.

Given a tetrahedron c in the mesh, we make a claim regarding its vertices. For c, let $\Sigma|_c$ be the *restriction* of Σ to c, defined as $\Sigma \cap c$. We define an *h-regular tetrahedron* c as one, where the set of $p \in \Sigma|_c$ are regular.

Lemma 8. *Given an h-regular tetrahedron c constructed from BCC lattice edges that are no longer than h, any vertex v of c that is labeled as having two materials (surface vertex) lies on Σ.*

Proof. Because v has two materials, it is the byproduct of a warp and snap to bring it to the Σ. Prior to this, v underwent a sequence of operations that depended on the cuts of edges incident to v. As long as there was only one such cut, v only warped once, and directly to the surface as computed by our indicator function oracle. We prove that because c is h-regular, this is always the case. Assume, for sake of contradiction, that there were multiple cuts, of different materials, on edges incident to v (if the materials were identical we preferentially pick the closest cut to move to). Without loss of generality, assume there are two cuts of material type AB and BC, we call x and y. Both x and y lie in the violating zone, on an edge incident to v. They are no further apart than $2h\alpha_{\mathrm{short}} < h$. This indicates that the two surface patches defining these cuts are no further apart than h as well.

We show there must be a medial axis point within distance h of $\Sigma|c$. Consider the medial axis for any single connected material region. By definition, it is a deformation retract of this region, and in addition, it touches any point where three materials meet. Thus, within a single region the medial axis is a single connected component that connects all triple-points. If we walk along the line segment joining x and y, we must therefore cross the medial axis because otherwise it would violate the above property. As a result, there is a medial axis point at this crossing. This medial axis point must be within distance h of v. However, because v lies on $\Sigma|_c$, this violates the fact that c is h-regular, leading to a contradiction.

In generalizing this case to when more than two cuts are adjacent v, the same logic holds for any pair of them, which is sufficient to create the same contradiction above. □

All h-regular tetrahedron are well behaved; in general, they act just like the tetrahedra in the isosurface stuffing algorithm [29]. Most importantly, when they do mesh a piece of the surface, they geometrically approximate the surface. As with any pointwise probe, it is impossible to guarantee that there are no tiny features missed on account of the mesh resolution being too coarse. However, we can guarantee that for h-regular tetrahedra containing surface patches of Σ, every point on an interface triangle representing this patch lies close to Σ. This "one-sided" notion of distances mirrors [29, Theorem 2] (only the distance of every mesh vertex to Σ is bounded). When the mesh is of fine enough resolution and each surface patch of Σ sufficiently smooth, the distance bound for h-regular tetrahedra becomes two sided—the claim follows for distances from Σ to the mesh.

5 Results and Discussion

Our implementation of lattice cleaving is extremely fast, requires virtually no user interaction, and achieves bounded dihedral angles. These bounds depend on choices of the violation parameters α_{long} and α_{short}. In this section, we discuss the importance of these choices, demonstrate the lattice cleaving algorithm on various data sets, and discuss other aspects of the algorithm.

5.1 Parameter Choice

The α_{long} and α_{short} parameters determine the distances along an edge, beyond which an edge-cut is considered "safe," or nonviolating. They decide the tradeoff between snapping/warping and stencil cleaving. A user who is primarily interested in visualization might choose to turn off violations completely by using $\alpha_{\mathrm{long}} = 0$ and $\alpha_{\mathrm{short}} = 0$. This will give the most accurate geometric representation with possibly degenerate elements. As the alpha values increase, the severity of the worst possible dihedral angles decreases, up until a point. With sufficiently large alpha values, lattice tetrahedra may become flat or even inverted. If a user is interested in only one side of the interface, flat elements can be tolerated by stripping them from the surface.

Because our meshes represent volumes on both sides of each interface surface, the most appropriate parameters would seem to be those used by Labelle and Shewchuk [29] in the *double-sided* surface case. However, our multimaterial violation conditions use slightly more conservative bounds to take into account the dihedral angles near triples and quadruples. We picked the parameters $\alpha_{\mathrm{short}} = 0.357$ and $\alpha_{\mathrm{long}} = 0.203$ and found they achieved a worse case minimum angle of 2.76 degree and worst case maximum angle of 175.426 degree. In practice, after simulating many hundreds of times steps of several dozen fluid simulations, corresponding to hundreds of millions of tetrahedra, we see much better angles for the vast majority of meshes.

5.2 Aliasing

A fundamental aspect of the lattice cleaving algorithm is that it operates at a fixed resolution. Any feature that falls below grid width will not be captured in the output mesh. For 2-material interfaces, this simply results in the smoothing of a surface, or removal of small topological features. For 3-material interfaces, the behavior becomes more interesting. As the feature size around a 3-material interface always goes to zero, there will always be some degree of approximation. But beyond that, the interaction of the background lattice faces with this approximation can lead to regular patterns of topological aliasing.

Fig. 13 illustrates a scenario that can arise in 2D. In the scenario, two material interfaces come together at a sharp

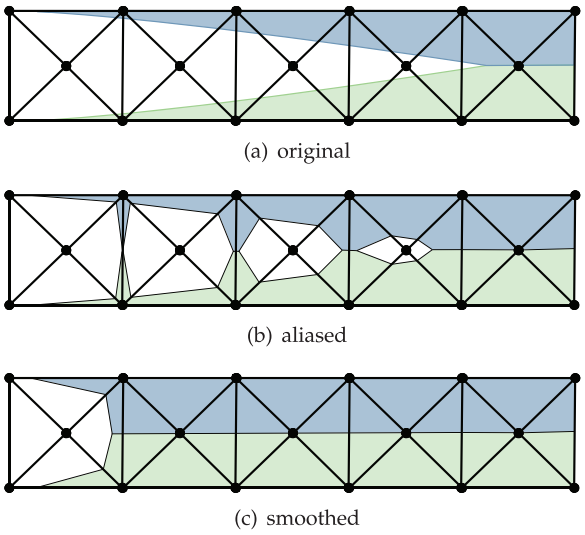


Fig. 13. (a) Three materials meet in a sharp corner feature. (b) Using material labels at each lattice vertex results in aliasing artifacts. (c) Preprocessing (e.g., smoothing) indicator functions eliminates this problem.

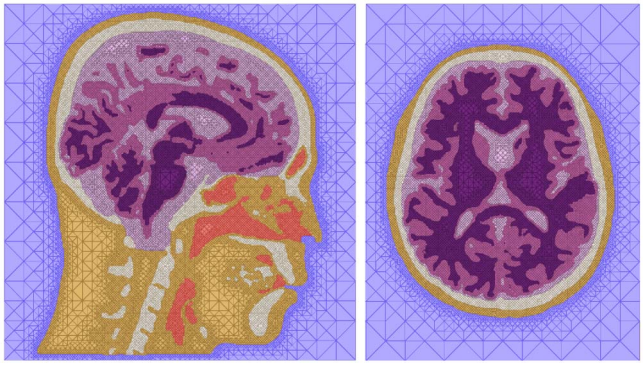


Fig. 14. Meshes generated from MRI scan of a human head. Resolution: $264 \times 264 \times 264$. Dihedral angles: $[4.33^\circ - 157.98^\circ]$. ≈ 5.3 million elements.

corner. Because the BCC lattice contain vertices from two sets of regular grids, (primal and dual), neighboring lattice faces alternate having either two or three unique material labels on their vertices. This leads to a saw-tooth-like pattern of spikes and pillars that form topologically distinct regions.

These aliasing artifacts are sometimes taken care of through the snapping and warping procedure because the sharp spikes that form are often in a violating condition anyway. However, since there is a space of cases which are not handled by snapping, explicit solutions for this problem are needed. A range of possible solutions exist, such as smoothing, tightening [48], or morphological operations such as dilation and erosion. If the input data cannot be smoothed, one could also design discrete local operators that intelligently change material labels on lattice vertices to avoid aliasing.

5.3 Examples

To illustrate some of the data sets for which lattice cleaving can be used, we provide several examples. Runtimes for these data sets were calculated on a machine with an Intel Core i7 3.2-GHz CPU and 12 GB of RAM.

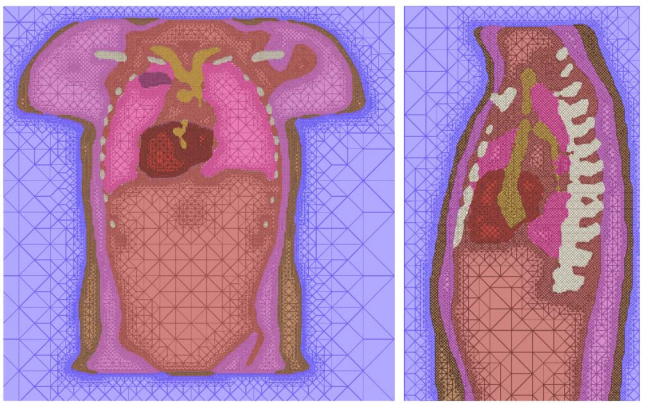


Fig. 15. Meshes generated from MRI scan of a human torso. Resolution: $208 \times 96 \times 208$. Dihedral angles: $[5.11^\circ - 159.91^\circ]$. ≈ 12.6 million elements.

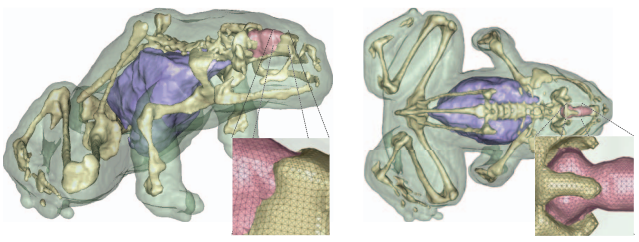


Fig. 16. Visualization using surface meshes generated from a segmented frog MRI. Resolution: $260 \times 245 \times 150$. Dihedral angles: $[6.06^\circ - 154.28^\circ]$. ≈ 14.8 million elements.

Fig. 14 shows a mesh generated from a segmented MRI scan of a human head. The algorithm completed in about 100 seconds and produced a mesh with roughly 5 million elements, all with dihedral angles between 4.33 and 157.98 degrees. Fig. 15 shows a mesh generated from a similar scan of a human torso. The algorithm completed in under a minute and produced a mesh with roughly 12 million elements, all with dihedral angles between 5.11 and 159.91 degrees.

It is often necessary to visualize multimaterial tetrahedral meshes before any simulation work is conducted. This spans from the need to qualitatively verify results are accurate, to spotting unexpected features that might influence solutions. Fig. 16 shows a visualization generated from a segmented frog MRI. The input volume was $260 \times 245 \times 150$ in size, and took just over a minute to mesh. The surface meshes can be extracted from the lattice cleaving algorithm as postprocess or generated alongside the tetrahedral mesh using the same stencil set.

This work also applies to multiphase fluid simulation and animation. To demonstrate this, we utilize the lattice cleaving algorithm in the core of a multiphase viscous fluid simulation. Fig. 17 shows a rendering and cutaway view of the underlying mesh used for physics. This simulation used a 64^3 background lattice (primal vertices), required 8 seconds to mesh, and produces, on average 1.2 million tetrahedra. Fig. 18 shows a histogram of the dihedral angles generated from 350 simulation frames. The majority of elements are of excellent quality, with small tails near the expected bounds. Counts for angles belonging to unwarped background lattice tets are scaled down.

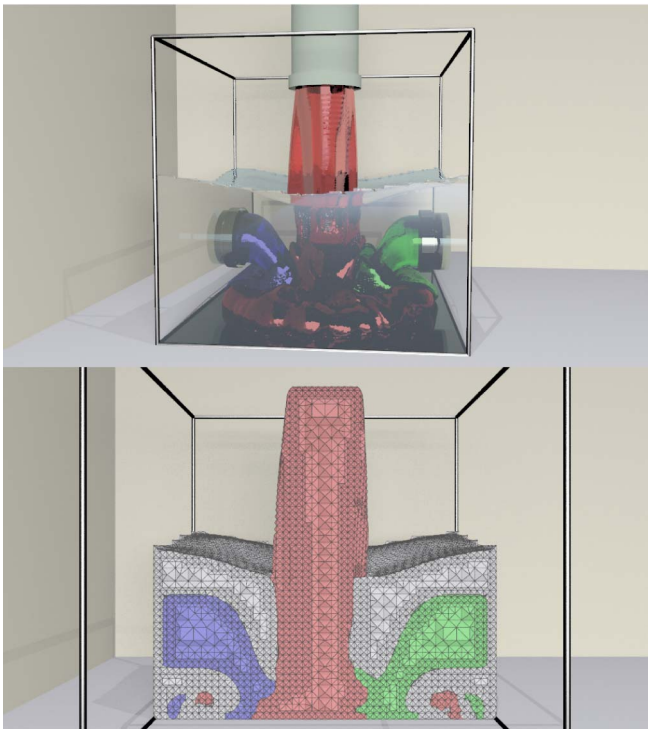


Fig. 17. A multiphase viscous fluid simulation. Each frame uses a conforming mesh to compute fluid physics.

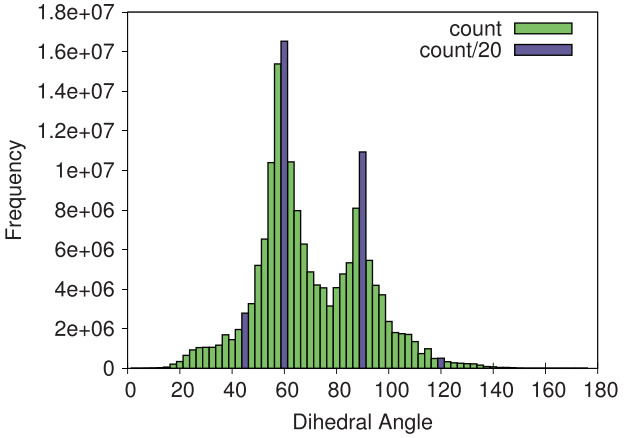


Fig. 18. A histogram of all the angles produced through the fluid simulation. Purple bars have $20\times$ the counts shown.

5.4 Algorithm Comparison

We evaluate our new meshing algorithm against two other packages commonly used in biomedical meshing: BioMesh3D v1.0 [3], [33] and CGAL v3.9 [44]. BioMesh3D uses variational optimization [33] to sample a domain and TetGen [43] to construct a Delaunay tetrahedrelization of those samples. CGAL's 3D mesh generation package contains an engine based on Delaunay refinement [16], [39], [40].

We compare these methods through their performance in three state-of-the-art simulation experiments. The first simulation is from an osseointegrated bone implant experiment, studying the effects of using direct current cathode stimulation to enhance the ability of implants to fuse into the skeletons of rabbits [26]. The second simulation is used for modeling the effects of EEG source on a human skull [17]. The third simulation used is for modeling ICD placement for defibrillation in children and adults [46].

Tables 1, 2, and 3 show the results of the simulations for these three biomedical applications, using the meshes generated by the three methods. For each of these experiments, we tuned parameters of the different system to get approximately the same number of elements (within a factor of 2). We analyzed the geometric quality of the mesh (dihedral angles), the condition numbers of the resulting stiffness matrices, and the number of iterations the solver took to converge. All of these simulations use the finite-element method with linear elements and they are solved using the SCIRun 4.5 [24] implementation of the minimal residual (MINRES) method [36] (a variant of the conjugate gradient method), combined with a Jacobi preconditioner.

We see from these simulations that the proposed meshing algorithm gives better overall bounds on dihedral angles and thereby avoids badly shaped tetrahedra. For the simulations, the proposed mesher and BioMesh perform similarly, while both mesh techniques generally outperform CGAL, requiring roughly half the number of iterations for the numerical solvers to reach convergence.

TABLE 1
Torso Simulation: 10 Materials, 208 × 96 × 208

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	12.6	7.43°	5.42e+06	553
BioMesh	22.1	0.00°	2.94e+09	399
CGAL	14.5	0.04°	2.42e+07	1008

TABLE 2
Head Simulation: Eight Materials, 264 × 264 × 264

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	11.7	6.49°	1.57e+18	493
BioMesh	7.9	0.18°	7.49e+18	525
CGAL	11.5	0.01°	1.04e+20	1026

TABLE 3
Rabbit Leg Simulation: Six Materials, 520 × 520 × 300

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	5.5	7.80°	1.22e+07	728
BioMesh	4.3	0.53°	1.12e+07	1143
CGAL	5.6	0.03°	8.94e+07	1490

Some of these differences are also seen qualitatively looking at renderings of these meshes, as in Fig. 19. Here, we see the rough, aliasing-type artifacts that result from the fact that CGAL operates only on discrete label maps—a reasonable explanation for the differences in simulation outcomes. Qualitatively, the proposed method and Biomesh are similar, but exhibit different strategies on grading/adaptivity. Besides the bounds on dihedral angles, the proposed method has significant advantages in reliability, robustness, and ease of use. All of the results for the proposed method were produced in one pass using one free parameter, which is the resolution of the finest mesh, and each took between 30 to 300 seconds. CGAL performance was competitive, each mesh taking only a few minutes to complete. BioMesh3D performs a series of variational optimizations with a set of interacting particles, and has a variety of free parameters. The BioMesh3D results required trial and error testing for parameter tuning to obtain valid mesh results. Each full mesh took over 12 hours to compute.

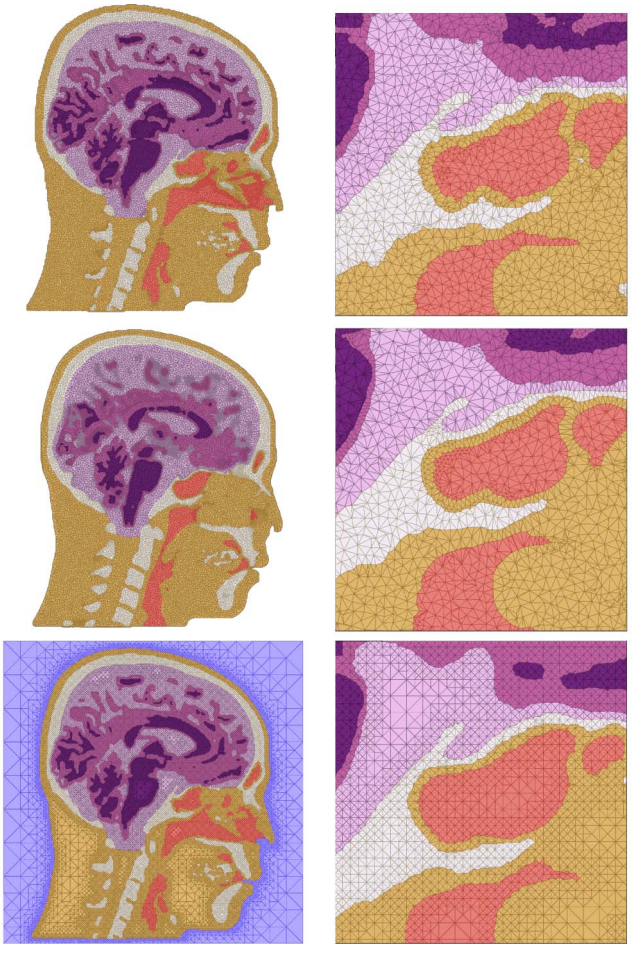


Fig. 19. Cross sections of the tetrahedral head meshes generated using (top) CGAL (middle) BioMesh3D and (bottom) Lattice Cleaving.

6 Conclusion

We have developed an extremely fast and robust conforming tetrahedral meshing algorithm for multimaterial volumetric domains. This method is guaranteed to produce meshes with bounded element quality and empirically this bound is significant, between 2.76 and 175.426 degrees. In practice, angles tend to be much better. An open-source implementation of this method, called Cleaver [25], is now available.

The method falls under the category of stenciling algorithms, operating locally on portions of a volume. As such, it is highly parallelizable and amenable to hardware acceleration. Moreover, a single generalized stencil is used for all stencil cases, removing the need for error-prone case tables while ensuring consistent meshes.

An octree structure is used to reduce element count by providing didadic grading in homogenous regions. Future work worth exploring is achieving grading on interface surfaces, which may be sufficiently smooth as to not require finest grid resolution elements. Similarly, alternative background lattices and stencils should be examined for the purpose of achieving anisotropic elements, which are invaluable for particular domains.

The simplifying restriction of at most one material transition per edge was instrumental in making this problem tractible. However, this restriction also places requirements on the smoothness of the input surfaces to avoid artifacts. Allowing for up to two transitions per edge might also relax this smoothness restriction significantly, if a suitable set of safe and compatible stencils can be constructed.

Acknowledgments

Thanks to Adam Bargteil and Darrell Swenson for the simulation and experiment data sets. This work was supported by grants from the National Center for Research Resources (5P41RR012553-14) and the National Institute of General Medical Sciences (8 P41 GM103545-14) from the National Institutes of Health.

References

[1] N. Amenta, S. Choi, T.K. Dey, and N. Leekha, "A Simple Algorithm for Homeomorphic Surface Reconstruction," *Int'l J. Computational Geometry & Applications*, vol. 12, nos. 1/2, pp. 125-141, 2002.

[2] M. Bern, P. Chew, D. Eppstein, and J. Ruppert, "Dihedral Bounds for Mesh Generation in High Dimensions," *Proc. Sixth Ann. ACM-SIAM Symp. Discrete Algorithms*, pp. 189-196, 1995.

[3] BioMesh3D, "Quality Mesh Generator for Biomedical Applications,"Scientific Computing and Imaging Inst. (SCI).

[4] J. Bloomenthal and K. Ferguson, "Polygonization of Non-Manifold Implicit Surfaces," *Proc. ACM SIGGRAPH,* pp. 309-316, 1995.

[5] J.-D. Boissonnat and S. Oudot, "Provably Good Sampling and Meshing of Surfaces," *Graphical Models,* vol. 67, no. 5, pp. 405-451, 2005.

[6] D. Boltcheva, M. Yvinec, and J.-D. Boissonnat, "Feature Preserving Delaunay Mesh Generation from 3d Multi-Material Images," *Computer Graphics Forum,* vol. 28, no. 5, pp. 1455-1464, 2009.

[7] L. Branets and G.F. Carey, "Condition Number Bounds and Mesh Quality," *Numerical Linear Algebra with Applications,* vol. 17, no. 5, pp. 855-869, 2010.

[8] J. Bronson, J. Levine, and R. Whitaker, "Lattice Cleaving: A Multimaterial Tetrahedral Meshing Algorithm with Guarantees," *Proc. 21st Int'l Meshing Roundtable (IMR),* Page to Appear, 2012.

[9] J.R. Bronson, J.A. Levine, and R.T. Whitaker, "Particle Systems for Adaptive, Isotropic Meshing of CAD Models," *Proc. 19th Int'l Meshing Roundtable (IMR),* pp. 279-296, Oct. 2010.

[10] S. Cheng, T. Dey, and J. Shewchuk, *Delaunay Mesh Generation.* CRC Press, 2012.

[11] S.-W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.-H. Teng, "Sliver Exudation," *Proc. 15th Ann. Symp. Computational Geometry,* pp. 1-13, 1999.

[12] S.-W. Cheng, T.K. Dey, and E.A. Ramos, "Delaunay Refinement for Piecewise Smooth Complexes," *Discrete & Computational Geometry,* vol. 43, no. 1, pp. 121-166, 2010.

[13] S.-W. Cheng, T.K. Dey, E.A. Ramos, and T. Ray, "Sampling and Meshing a Surface with Guaranteed Topology and Geometry," *SIAM J. Computing,* vol. 37, no. 4, pp. 1199-1227, 2007.

[14] N. Chentanez, B.E. Feldman, F. Labelle, J.F. O'Brien, and J.R. Shewchuk, "Liquid Simulation on Lattice-Based Tetrahedral Meshes," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA),* pp. 219-228, Aug. 2007.

[15] L.P. Chew, "Constrained Delaunay Triangulations," *Proc. Third Ann. Symp. Computational Geometry (SCG '87),* pp. 215-222, 1987.

[16] L.P. Chew, "Guaranteed-Quality Mesh Generation for Curved Surfaces," *Proc. Ninth Ann. Symp. Computational Geometry,* pp. 274-280, 1993.

[17] M. Dannhauer, B. Lanfer, C.H. Wolters, and T.R. Knsche, "Modeling of the Human Skull in EEG Source Analysis," *Human Brain Mapping,* vol. 32, no. 9, pp. 1383-1399, 2011.

[18] T.K. Dey, F. Janoos, and J.A. Levine, "Meshing Interfaces of Multi-Label Data with Delaunay Refinement," *Eng. with Computers,* vol. 28, no. 1, pp. 71-82, Jan. 2012.

[19] T. Etiene, L. Nonato, C. Scheidegger, J. Tienry, T. Peters, V. Pascucci, R. Kirby, and C. Silva, "Topology Verification for Isosurface Extraction," *IEEE Trans. Visualization & Computer Graphics,* vol. 18, no. 6, pp. 952-965, June 2012.

[20] L.A. Freitag and C. Ollivier-Gooch, "Tetrahedral Mesh Improvement Using Swapping and Smoothing," *Int'l J. Numerical Methods in Eng.,* vol. 40, no. 21, pp. 3979-4002, 1997.

[21] P. Frey and P. George, *Mesh Generation.* John Wiley & Sons, 2010.

[22] A. Fuchs, "Automatic Grid Generation with Almost Regular Delaunay Tetrahedra," *Proc. Seventh Int'l Meshing Roundtable (IMR),* pp. 133-147, 1998.

[23] A. Guéziec and R.A. Hummel, "Exploiting Triangulated Surface Extraction Using Tetrahedral Decomposition," *IEEE Trans. Visualization & Computer Graphics,* vol. 1, no. 4, pp. 328-342, Dec. 1995.

[24] S. Institute, "SCIRun: A Scientific Computing Problem Solving Environment," Scientific Computing and Imaging Inst. (SCI), http://www.scirun.org, 2012.

[25] S. Institute, "Cleaver: A MultiMaterial Tetrahedral Meshing Library and Application," Scientific Computing and Imaging Inst. (SCI), http://www.sci.utah.edu/software/cleaver. 2012.

[26] B.M. Isaacson, L.B. Brunker, A.A. Brown, J.P. Beck, G.L. Burns, and R.D. Bloebaum, "An Evaluation of Electrical Stimulation for Improving Periprosthetic Attachment," *J. Biomedical Materials Research Part B: Applied Biomaterials,* vol. 97B, no. 1, pp. 190-200, 2011.

[27] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual Contouring of Hermite Data," *Proc. ACM 29th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '02),* pp. 339-346, 2002.

[28] B.M. Klingner and J.R. Shewchuk, "Aggressive Tetrahedral Mesh Improvement," *Proc. 16th Int'l Meshing Roundtable (IMR),* pp. 3-23, 2007.

[29] F. Labelle and J.R. Shewchuk, "Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles," *Proc. ACM SIGGRAPH,* 2007.

[30] Y. Liu, P. Foteinos, A. Chernikov, and N. Chrisochoides, "Multi-Tissue Mesh Generation for Brain Image," *Proc. 19th Int'l Meshing Roundtable (IMR),* pp. 367-384, Oct. 2010.

[31] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH,* pp. 163-169, 1987.

[32] B. Merriman, J.K. Bence, and S.J. Osher, "Motion of Multiple Junctions: A Level Set Approach," *J. Computational Physics,* vol. 112, no. 2, pp. 334-363, 1994.

[33] M.D. Meyer, R.T. Whitaker, R.M. Kirby, C. Ledergerber, and H. Pfister, "Particle-Based Sampling and Meshing of Surfaces in Multimaterial Volumes," *IEEE Trans. Visualization & Computer Graphics,* vol. 14, no. 6, pp. 1539-1546, Nov. 2008.

[34] N. Molino, R. Bridson, J. Teran, and R. Fedkiw, "A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra," *Proc. 12th Int'l Meshing Roundtable (IMR),* pp. 103-114, 2003.

[35] G.M. Nielson and R. Franke, "Computing the Separating Surface for Segmented Data," *Proc. IEEE Visualization,* pp. 229-233, 1997.

[36] C. Paige and M. Saunders, "Solution of Sparse Indefinite Systems of Linear Equations," *SIAM J. Numerical Analysis,* vol. 12, no. 4, pp. 617-629, 1975.

[37] A.A. Pasko, V. Adzhiev, A. Sourin, and V.V. Savchenko, "Function Representation in Geometric Modeling: Concepts, Implementation and Applications," *The Visual Computer,* vol. 11, no. 8, pp. 429-446, 1995.

[38] J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven, "High-Quality Consistent Meshing of Multi-Label Data Sets," *Proc. 20th Int'l Conf. Information Processing in Medical Imaging (IPMI),* pp. 198-210, 2007.

[39] J. Ruppert, "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation," *J. Algorithms,* vol. 18, no. 3, pp. 548-585, 1995.

[40] J.R. Shewchuk, "Tetrahedral Mesh Generation by Delaunay Refinement," *Proc. 14th Ann. Symp. Computational Geometry,* pp. 86-95, 1998.

[41] J.R. Shewchuk, "What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures," *Proc. Int'l Meshing Roundtable (IMR),* pp. 115-126, 2002.

[42] J.R. Shewchuk, "General-Dimensional Constrained Delaunay and Constrained Regular Triangulations I: Combinatorial Properties," *Discrete and Computational Geometry,* vol. 39, pp. 580-637, 2005.

[43] H. Si, "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator," http://tetgen.berlios.de/, 2013.

[44] CGAL, "Computational Geometry Algorithms Library," http://www.cgal.org, 2013.

[45] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun, "Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation," *ACM Trans. Graphics,* vol. 28, no. 3, article 75, 2009.

[46] J.K. Triedman, M. Jolley, J. Stinstra, D.H. Brooks, and R. MacLeod, "Predictive Modeling of Defibrillation Using Hexahedral and Tetrahedral Finite Element Models: Recent Advances," *J. Electrocardiology,* vol. 41, no. 6, pp. 483-486, 2008.

[47] J. Wang and Z. Yu, "Feature-Sensitive Tetrahedral Mesh Generation with Guaranteed Quality," *Computer-Aided Design,* vol. 44, no. 5, pp. 400-412, 2012.

[48] J. Williams and J. Rossignac, "Tightening: Curvature-Limiting Morphological Simplification," *Proc. ACM Symp. Solid and Physical Modeling (SPM),* 2004.

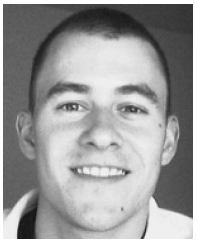
[49] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram," *Computer Graphics Forum,* vol. 28, no. 5, pp. 1445-1454, 2009.

[50] M.A. Yerry and M.S. Shephard, "Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique," *Int'l J. Numerical Methods in Eng.,* vol. 20, pp. 1965-1990, 1984.

[51] Z. Yu, M.J. Holst, and J.A. McCammon, "High-Fidelity Geometric Modeling for Biomedical Applications," *Finite Elements in Analysis and Design,* vol. 44, no. 11, pp. 715-723, 2008.

[52] N. Zhang, W. Hong, and A. Kaufman, "Dual Contouring with Topology-Preserving Simplification Using Enhanced Cell Representation," *Proc. IEEE Visualization (VIS '04),* pp. 505-512, 2004.

[53] Y. Zhang, C. Bajaj, and B.-S. Sohn, "3d Finite Element Meshing from Imaging Data," *Computer Methods in Applied Mechanics and Eng.*, vol. 194, no. 4849, pp. 5083-5106, 2005.
[54] Y. Zhang, T. Hughes, and C.L. Bajaj, "Automatic 3d Mesh Generation for a Domain with Multiple Materials," *Proc. Int'l Meshing Roundtable (IMR)*, pp. 367-386, 2007.
[55] Y. Zhang and J. Qian, "Resolving Topology Ambiguity for Multiple-Material Domains," *Computer Methods in Applied Mechanics and Eng.*, vol. 247/248, pp. 166-178, 2012.



Jonathan Bronson received the MS degree in computer science from the University of Maryland, Baltimore County, in 2008. He is currently working toward the PhD degree in computer science at the University of Utah. His research interests include computational geometry and mesh generation, scientific and information visualization, as well as image processing. He is a student member of the IEEE.

Joshua A. Levine received the PhD degree in computer science from the Ohio State University and completed a postdoc from the Scientific Computing and Imaging Institute of University of Utah. Currently, he is an assistant professor at Visual Computing division in the School of Computing, Clemson University. His research interests include geometric modeling, scientific visualization, mesh generation, medical imaging, and computational geometry. He is a member of the IEEE and ACM.

Ross Whitaker received the graduate degree summa cum laude with BS degree in electrical engineering and computer science from Princeton University in 1986. At UNC, he received the Alumni Scholarship Award, and received the PhD degree in computer science in 1994. From 1986 to 1988, he was with the Boston Consulting Group, entering the University of North Carolina at Chapel Hill in 1989. From 1994 to 1996, he was at the European Computer-Industry Research Centre in Munich Germany as a research scientist in the User Interaction and Visualization Group. From 1996 to 2000, he was an assistant professor in the Department of Electrical Engineering at the University of Tennessee and received the US National Science Foundation (NSF) Career Award. Since 2000, he has been at the University of Utah where he is a professor in the School of Computing and a faculty member of the Scientific Computing and Imaging Institute. He teaches discrete math, scientific visualization, and image processing. He has led graduate-level research group in image analysis, geometry processing, and scientific computing, with a variety of projects supported by both federal agencies and industrial contracts. He is a senior member of the IEEE.