Lattice Cleaving: Conforming Tetrahedral Meshes of Multimaterial Domains with Bounded Quality

Jonathan R. Bronson, Joshua A. Levine, and Ross T. Whitaker

Scientific Computing and Imaging Institute, Salt Lake City, UT, U.S.A.
{bronson,jlevine,whitaker}@sci.utah.edu

Summary. 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, in order to reduce element counts in regions of homogeneity.

Key words: tetrahedral meshing, multimaterial, multi-label, biomedical, conformal meshing, watertight, mesh quality, bounded quality, adaptive 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 make 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 [5]. 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.

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 [25], where material boundaries are derived from segmentations or labels, as well as simulations that rely on implicit representations of physical interfaces [10]. In this paper we specifically address tetrahedral meshing with *multimaterial* interfaces, where the geometric constraints are non-manifold 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 non-manifold 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 tetrahedra, 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 run times. Results on biomedical volumes and fluid simulations demonstrate the algorithm reliably achieves fast run times, geometric fidelity, and good quality elements.

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 using Delaunay triangulations. One of the most popular strategies relies on Delaunay refinement [11, 27], 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 [4] and Cheng et al. [9] 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 non-manifold domains followed [8]. However, these algorithm 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 [12, 26].

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 [6, 22, 30]. These strategies typically decouple, to various degrees, the vertex placement problem from the triangulation/tetrahedralization problem. The 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.

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 [29]. 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 $180°$ 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 [28] 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 [14, 17]. 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 [20]. 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 [3, 24], with extensions to non-manifold 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. An octree on the regular lattice can be used to give adaptively sized elements [31]. Another strategy for conforming is to warp a background lattice so that primitives align with boundaries [15]. Molino et al. [23] 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 [18] propose a combination of lattice warping and stenciling, with appropriate rules that decide which combination of strategies to use, based on the input data, in order to ensure good quality. They describe a computer-assisted proof to compute quality bounds for their *isosurface stuffing* algorithm. The proposed method shares several aspects of the Labelle and Shewchuk approach. 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 non-manifold 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.* [19] or dual-contouring [32]. 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 body-centered cubic (BCC) lattice. Similar to many surfacing and meshing algorithms [16, 20], we rely on a set of stencils that capture local material configurations. We use the strategy of Labelle and Shewchuck [18] 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 in order 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.

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*, $F = \{f_i | f_i : V \mapsto \Re\}$ [21, 25]. 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

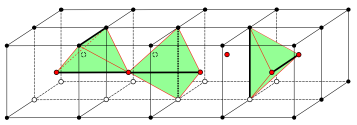


Fig. 1: The BCC lattice is composed of two grids of primal and dual vertices. Each vertex is incident to 14 edges, 36 faces, and 24 tetrahedra.

Stenciling algorithms rely on a set of regular cells. The configuration of the interfaces on these cells are used to generate an index that corresponds to some predefined tessellation. We employ a BCC lattice (Fig. 1), where each cell is composed of 8 normal or *primal* cubic lattice vertices, plus a 9th *dual* vertex in the center. In addition to the 12 edges of a regular cubic cell, there are 8 diagonal edges connecting each dual vertex to its cell's primal vertices, and 6 connecting dual to dual. Fanning out from the dual vertex are 24 lattice tetrahedra, each of which spans two lattice cells.

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. Each lattice vertex represents a single material at that point, which is given by the indicator function with maximum value. Ties are 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* [18]. These edge-cuts sample a surface separating two materials.

A similar logic applies to junctions of more than two materials. A lattice face with 3 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 3 materials meet. A lattice tetrahedron may have up to four unique material labels. The 4 vertices, and the 4 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 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*.

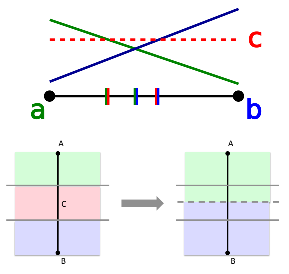


Fig. 2: An edge with materials a and b maximum on its endpoints, but with a third material c becoming maximum on the interval between.

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 4.2.

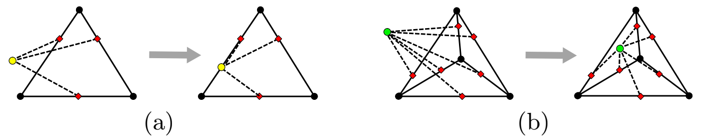
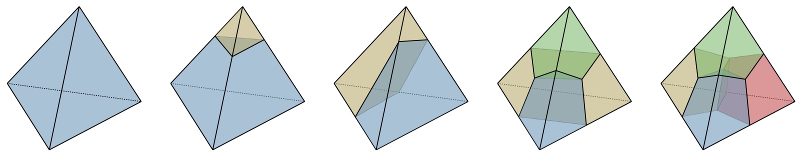


Fig. 3: Triples (a) and quadruples (b) are forced to lie within the primitive that contains the associated edge-cuts.

3.3 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 5 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 1 or 2 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)

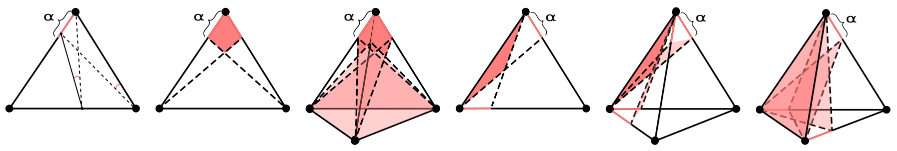


(a) 0-cut (b) 3-cut (c) 4-cut (d) 5-cut (e) 6-cut

Fig. 4: The 5 unique interface topologies determined by the number of cut-points present on a lattice tetrahedron.

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 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. 5(a), 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 halfspace is defined by connecting the point at distance α on one edge, to the opposite lattice vertex (Fig. 5(b)). Similarly, quadruple-points can be anywhere inside the lattice tetrahedron, so their vertex violation regions are formed by the intersection of 3 half-spaces defined by planes. (Fig. 5(c)).



(a) c-vertex (b) t-vertex (c) q-vertex (d) t-edge (e) q-edge (f) q-face

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.

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. 5(d,e)). 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. 5(f)).

3.4 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 conjuction, vertices are *warped* spatially in order to conform to the interface surfaces.

The multimaterial case introduces some extra complexities in dealing with the consequences of snaps and warps. The first challenge arises when a violated lattice vertex is incident to multiple lattice edges that have cuts. In the two-material case, Labelle and Shewchuk [18] perform a single warp to a single cut and remove the other cuts, essentially pulling them into the warped vertex. In the multimaterial case, this is unsatisfactory, because the adjacent cuts could be interfaces to several different materials. After warping, these additional materials may still be present on the remaining edges. Thus, we must ensure that all of the cuts and triples on incident edges and faces are updated appropriately.

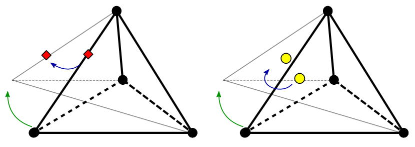

(a) cut-project (b) triple-project

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.

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.

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 followup 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 colocated with the corresponding cut/triple on that face.

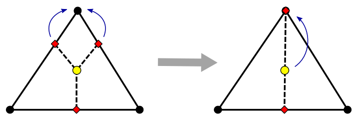


Fig. 7: Degenerate triples or quadruples are removed by subsequent snaps.

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.

3.5 Generalized Stencil

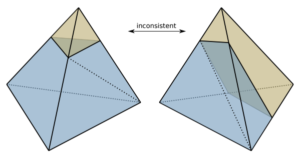


Fig. 8: Stencils for lattice tetrahedra must be consistent across faces.

After snapping and warping, the polyhedra from the topological cases described in Section 3.3 may have collapsed into intermediate topologies. Each such topology demands not only a valid tessellation stencil, but 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 it also removes the requirement of implementing and storing a large stencil table that is prone to construction and transcription errors [13].

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 3.4 always simplifies the topology in a manner equivalent to a series of edge coallapses. 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.

To maintain consistency between adjacent tetrahedra, virtual cuts are chosen first. The remaining virtual points all cascade into place from this decision. 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 cutpoint. If there are multiple options, we choose the point that is collocated with the most other points, virtual or real.

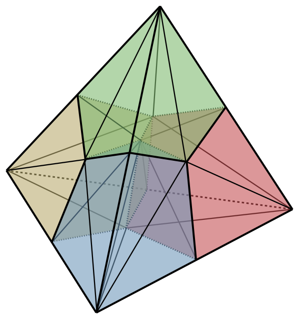


Fig. 9: The generalized stencil is constructed from the 6-cut case. Edges connect each interface point to its associated lower order features.

Note that these rules for generalizing lattice tetrahedra operate once; they are merely the mechanism for generating a consistent set of stencils. Also, 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 [18].

3.6 Algorithm

The full lattice cleaving algorithm proceeds as follows. Using an octree structure to reduce storage and allow grading, we first sample and label each BCC lattice vertex. Cuts, triples and quadruples are computed for each lattice tetrahedron that has multiple materials. Any lattice tetrahedron that is not the 6-cut case is generalized by labeling the locations of virtual cuts, triples, and quadruples. This is followed by three phases of snapping and warping.

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. All degeneracies are fixed with additional snaps.

In the second phase, all violated lattice edges are identified and visited exactly once. If one or more triples or quadruples violates an edge, we snap them to that edge. These snaps are implemented so that a lattice edge or face may contain singular points of transitions or be a single material across its entirety; we do not allow materials to lie on the half-edge, or half-face.

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 on the face that was violated. Sometimes a quadruple-point violates a face that no longer contains a triple. 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. To finish, we output all stencil tetrahedra that contain 4 unique vertices, skipping over any that were removed during the warping and snapping process.

4 Dihedral Angles and Geometric Fidelity

The algorithm described in Section 3.6 is designed to ensure that both dihedral angles are bounded and geometric distortion of input surfaces is controlled. In what follows, we sketch proofs of these properties.

4.1 Bounded Dihedral Angles

The violation rules defined in Section 3.3 disallow vertex positions that could lead to undesirable tetrahedra. The following proof relies 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 several ways to classify types of bad tetrahedron [2, 7]. One useful partitioning groups such tetrahedra into two sets: tetrahedra whose vertices are nearly collinear, and tetrahedra whose vertices are nearly coplanar. 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 triangular faces that can occur these undesirable tetrahedra. 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 certainly be of bad quality.

The rules comprising our algorithm make it impossible for output tetrahedra to become badly shaped (and consequently, they have bounded dihedral angles). First, we show that background lattice tetrahedra stay of good quality. This properties induce 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. This direct proof does, however, provide insights into why this algorithm is successful at achieving bounded dihedral angles and gives a foundation for modifications or extensions.

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

A tetrahedron contains 6 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 $\hat{n}_2$:

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

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

Definition 3. *The* aspect ratio, $\mathrm{ar}_t = a/l$*, for a tetrahedron, t, where a the length 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 degerenate in these ways, it must be the case that there are either dihedral angles of $0°$, $180°$, or both. We next define a notion of ϵ-good.

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 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 $\theta_{\max} < 180° - \kappa$. Similarly, for a triangle face f, 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 $\phi_{\max} < 180° - \kappa$.*

We omit the proof for brevity. Lemma 1 provides a crucial link between the bound on aspect ratio and then bound on the minimum and maximum dihedral angles. The mechanisms of our algorithm are such that they specifically ensure the property of ϵ-good for tetrahedra, and in doing so, coupled with Lemma 1. it must be the case that they have bounded dihedral angles.

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 2. *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 3.4.*

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. □

Lemma 3. *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 2, 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 3.3 would be triggered making these impossible. □

Lemma 4. *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 3 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 3.3 we know these faces are ϵ-good The case of two internal faces is explicitly prevented, for brevity we summarize why. There are 5 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 5. *All stencil triangles maintain ϵ-good planar angles after snapping and warping, as described in Section 3.4.*

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 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 6. *All output tetrahedron have ϵ-good dihedral angles.*

Proof. Let t be an output tetrahedron. By Lemma 4, t has at least two faces joined at an ϵ-good dihedral angle along edge e. By Lemma 5 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, since it positions all four vertices. □

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 6 and 1. □

This proof shows that the meshes from the lattice-cleaving algorithm never degenerate, in fact Lemma 5 also ensures we produce no bad solid angles (e.g. "spires"), despite them 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.

4.2 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 [8]. 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, $\mathrm{lfs}\colon \Sigma \to \mathbb{R}$, 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 7. *Given an h-regular tetrahedron c constructed from BCC lattice edges which 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 $2h\alpha_{\text{short}} < h$. 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 which 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.

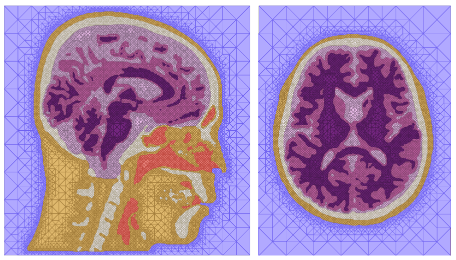
In generalizing this case to when more than two cuts are adjacent v, the same logic holds for any pair of them, which is sufficient to create the same contradiction above. □

All h-regular tetrahedron are well-behaved; in general, they act just like the tetrahedra in the isosurface stuffing algorithm [18]. 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 Theorem 2 of [18] (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

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}. 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 [18] 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 $\alpha_{\text{short}} = 0.357$ and $\alpha_{\text{long}} = 0.203$ and found they achieved a worse case minimum angle of $2.76°$ and worst case maximum angle of $175.426°$. 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.

To illustrate some of the datasets for which lattice cleaving can be used, we provide several examples. Fig. 10(a) 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 $157.98°$. Fig. 10(b) 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 $159.91°$.


(a) head

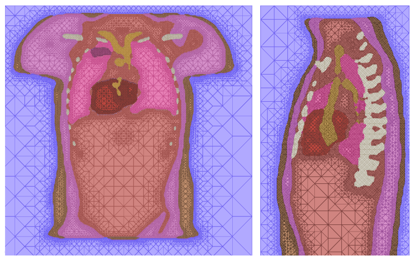
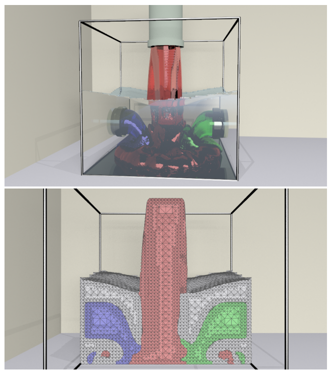
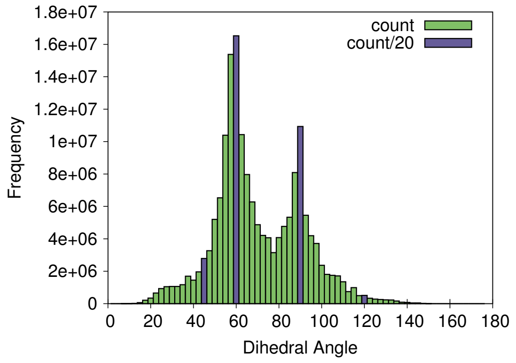

(b) torso

Fig. 10: Meshes generated from medical data. (a) MRI scan of a human head. Resolution: 264x264x264. Dihedral angles: $[4.33° - 157.98°]$. $\approx$ 5.3 million elements. (b) MRI scan of a human torso. Resolution: 208x96x208. Dihedral angles: $[5.11° - 159.91°]$. $\approx$ 12.6 million elements

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. 11(a) 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. 11(b) 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.



(a)



(b)

Fig. 11: A multiphase viscous fluid simulation. (a) Each frame uses a conforming mesh to perform fluid physics. (b) A histogram of all the angles produced throughout the simulation.

Acknowledgments:

This work was supported in part by grants from the US National Institutes of Health/National Center for Research Resources Center for Integrative Biomedical Computing, 2P41 RR01125523-12.

References

1. N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Comput. Geometry Appl.*, 12(1-2), 2002.
2. M. Bern, P. Chew, D. Eppstein, and J. Ruppert. Dihedral bounds for mesh generation in high dimensions. In *SODA*, pages 189–196, 1995.
3. J. Bloomenthal and K. Ferguson. Polygonization of non-manifold implicit surfaces. In *SIGGRAPH*, pages 309–316, 1995.
4. J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.
5. L. Branets and G. F. Carey. Condition number bounds and mesh quality. *Numerical Linear Algebra with Applications*, 17(5):855–869, 2010.
6. J. R. Bronson, J. A. Levine, and R. T. Whitaker. Particle systems for adaptive, isotropic meshing of CAD models. In *IMR*, pages 279–296, Oct. 2010.
7. S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver exudation. In *Symp. on Comp. Geom.*, pages 1–13, 1999.
8. S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. *Discrete & Computational Geometry*, 43(1):121–166, 2010.
9. S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SIAM J. Comput.*, 37(4):1199–1227, 2007.
10. N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *SCA*, pages 219–228, Aug. 2007.
11. L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Symp. on Comp. Geom.*, pages 274–280, 1993.

12. T. K. Dey, F. Janoos, and J. A. Levine. Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers*, 28(1):71–82, Jan. 2012.
13. T. Etiene, L. Nonato, C. Scheidegger, J. Tienry, T. Peters, V. Pascucci, R. Kirby, and C. Silva. Topology verification for isosurface extraction. *IEEE TVCG*, 18(6):952–965, 2012.
14. L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *Int. J. Num. Meth. Eng.*, 40(21):3979–4002, 1997.
15. A. Fuchs. Automatic grid generation with almost regular Delaunay tetrahedra. In *IMR*, pages 133–147, 1998.
16. A. Guéziec and R. A. Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE TVCG*, 1(4):328–342, 1995.
17. B. M. Klingner and J. R. Shewchuk. Aggressive tetrahedral mesh improvement. In *IMR*, pages 3–23, 2007.
18. F. Labelle and J. R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *SIGGRAPH*, 2007.
19. Y. Liu, P. Foteinos, A. Chernikov, and N. Chrisochoides. Multi-tissue mesh generation for brain image. In *19th IMR*, pages 367–384, October 2010.
20. W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169. ACM, 1987.
21. B. Merriman, J. K. Bence, and S. J. Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112(2):334 – 363, 1994.
22. 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. Vis. Comput. Graph.*, 14(6):1539–1546, 2008.
23. N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *IMR*, pages 103–114, 2003.
24. G. M. Nielson and R. Franke. Computing the separating surface for segmented data. In *IEEE Visualization*, pages 229–233, 1997.
25. A. A. Pasko, V. Adzhiev, A. Sourin, and V. V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.
26. J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven. High-quality consistent meshing of multi-label datasets. In *IPMI*, pages 198–210, 2007.
27. J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.
28. J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Symp. on Comp. Geom.*, pages 86–95, 1998.
29. J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *IMR*, pages 115–126, 2002.
30. J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.*, 28(3), 2009.
31. M. A. Yerry and M. S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *Int. J. Num. Meth. Eng.*, 20:1965–1990, 1984.
32. Y. Zhang, T. Hughes, and C. L. Bajaj. Automatic 3d mesh generation for a domain with multiple materials. In *IMR*, pages 367–386, 2007.