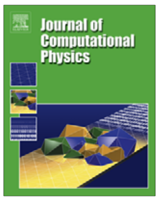
Journal of Computational Physics 228 (2008) 619–640

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



An anisotropic scale-invariant unstructured mesh generator suitable for volumetric imaging data

Andrew P. Kuprat [*], Daniel R. Einstein

Pacific Northwest National Laboratory, Biological Monitoring and Modeling, 902 Battelle Blvd., P.O. Box 999, MSIN P7-58, Richland, WA 99352, United States

ARTICLE INFO

Article history:
Received 22 June 2007
Received in revised form 3 June 2008
Accepted 18 September 2008
Available online 18 October 2008

PACS:
87.15.Aa
87.19.Uv
87.57.−s
87.61.−c
46.15.−x
02.70.−c

Keywords:
Computational fluid dynamics
Meshing biological structures
Delaunay

ABSTRACT

We present a boundary-fitted, scale-invariant unstructured tetrahedral mesh generation algorithm that enables registration of element size to local feature size. Given an input triangulated surface mesh, a feature size field is determined by casting rays normal to the surface and into the geometry and then performing gradient-limiting operations to enforce continuity of the resulting field. Surface mesh density is adjusted to be proportional to the feature size field and then a layered anisotropic volume mesh is generated. This mesh is "scale-invariant" in that roughly the same number of layers of mesh exist in mesh cross-sections, between a minimum scale size $L_{\min}$ and a maximum scale size $L_{\max}$. We illustrate how this field can be used to produce quality grids for computational fluid dynamics based simulations of challenging, topologically complex biological surfaces derived from magnetic resonance images. The algorithm is implemented in the Pacific Northwest National Laboratory (PNNL) version of the Los Alamos grid toolbox LaGriT. *Research funded by the National Heart and Blood Institute Award 1RO1HL073598-01A1.*

1. Introduction

Computational continuum physics is fast becoming an important part of biomedical research. Typically, the geometries for biomedical problems are derived from medical imaging modalities such as magnetic resonance imaging (MRI), and are geometrically complex compared to many engineering problems. Efficient unstructured mesh generation of these geometries is especially challenging because no 'true' reference geometry exists, due to the finite resolution of the imaging data. In this presentation, we propose a novel approach to this problem, based on a fast estimation of the local feature size. Due to the inherent uncertainty in the imaging-derived geometries, we formally restrict our attention to problems wherein there is a definite lower bound on the local feature size (i.e., on the order of a voxel) and to situations where, given that uncertainty, we have the latitude to alter the envelope of the prescribed surface mesh (i.e., on order a fraction of a voxel) to make the mesh suitable for computational fluid dynamics (CFD) or other numerical simulations.

* Corresponding author. Tel.: +1 509 376 6227; fax: +1 509 376 9449.
E-mail address: andrew.kuprat@pnl.gov (A.P. Kuprat).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.09.030

Excellent isotropic unstructured tetrahedral grid generation algorithms exist for creating finite-element or finite-volume grids from closed triangulated manifolds [3,17]. Isotropic algorithms, i.e., approaches that attempt to construct tetrahedral elements with nearly equal internal angles and approximately equal edge lengths, may be well suited to a large class of biomedical problems. However, for certain classes of problems such as computational fluid dynamics, isotropic elements may be neither necessary nor particularly appropriate. In boundary layer regions of the flow field, for example, derivatives normal to the wall can be much greater than they are parallel to the flow. Anisotropic grids can better resolve these features of the flow [11,22] by concentrating computational grid at the wall while keeping the overall computational cost of the problem tractable. That efficiency applies to problems beyond CFD as well. An excellent reference on anisotropic mesh generation is [9].

Achieving a pre-determined number of elements or a minimum number of elements through the smallest features of a geometry, is important for correctly resolving gradients [8] or to capture physical variations such as laminae. An example of the latter is the musculature of the heart which has three distinct layers of fiber orientations. A layered anisotropic approach – whether advancing front or Delaunay – has the advantages of creating elements that are mostly orthogonal to the wall while essentially decoupling the grid density in the normal and tangential directions. That decoupling raises the issue of what criterion to adopt for the tangential density. With surfaces derived from imaging data, the organization and density of the original surface triangles are functions of the resolution of the digital data rather than the numerical problem to be solved. Simply generating a volume grid from the original surface spacing could result in grossly under-resolving the computed field where the surface density is close to that of the local feature size or conversely over-resolving the computed field where the surface density is much finer than that of the local feature size. These observations lead to a consideration of the local feature size as an important criterion for sizing and gradation control of the surface that is complimentary to criteria that attempt to preserve surface features, topology and curvature.

In large part, previous attempts to base grid density on a concept of local feature size or scale have either been based on the definition of "throw-away" background grids onto which a local feature size field was computed [20,31,29,21], or have been based on point-insertion techniques to locally refine – but not de-refine – coarse tetrahedral grids [16], or have been based on advancing fronts [10,20]. An earlier related approach was based on Dijkstra's algorithm and element subdivision [8]. As an example of an approach based on a pre-computed feature size field, Persson [21] recently defined a local feature size field on a background grid. The feature field was defined locally as the sum of the distance of a given point to the boundary and to the medial axis. Element sizes were determined by gradient-limiting competing local element size fields based on the feature field and the curvature field. The disadvantages of this and related approaches are twofold. First, the medial axis of a triangulated manifold is inherently unstable in the sense that small perturbations in the surface triangulation can lead to large perturbations in the medial axis and thus also in the definition of local feature size. Second, the construction and computation of a feature size field on a "throw-away" background grid is computationally expensive.

In this paper, we define a feature size field on an input triangulated surface mesh without a background grid and without referencing the medial axis. Thus, determination of the feature size field is not only computationally efficient, but also robust in the sense that it is continuous and does not change unreasonably under perturbation of the surface mesh. Field values at a point on the surface are generated by shooting a ray from the point in the direction of an inwards "synthetic" normal that is well-defined even where the surface is not smooth and measuring the distance until the surface is re-intersected by the ray. We then perform a gradient-limiting operation on the field to enforce continuity. Layered, conforming, anisotropic volume mesh generation is accomplished directly with this surface field. Prior to volume mesh generation, we modify the surface mesh so that edge lengths are proportional to the feature size field, with the constraint that refinement/de-refinement preserves topology and geometry. Surface modification is iterated with a volume-conserving smoothing, with the result that surface triangles are well-shaped, well-organized and graded. Following surface modification, layers of points are strategically placed along the rays normal to the surface and connected into orthogonal layers with a Delaunay algorithm.

Because we employ Delaunay meshing, the degree of anisotropy is effectively limited by the consideration that sampling frequency on the boundary must be greater than some multiple of the local feature size in order for the topology of the input surface to be known rigorously to be recoverable from the Delaunay volume mesh [1]. We note, however, that in practice surface point densities may be lower than the rigorous result and still lead to successful boundary recovery. Point densities in [2] as well as in this work are significantly lower. These point densities translate into modest degrees of anisotropy that can still bring significant savings in terms of overall element count.

We show the results of our algorithm on three complex geometries derived from medical imaging data: a rat nose phantom (genus 1), a rat lung (genus 0) and a human heart (genus 12). The resulting meshes are deemed scale-invariant in that it is possible to obtain a desired number of layers across cross-sections in the domain, independent of scale. For each of these cases, we give aspect-ratio statistics that indicate the high quality of tetrahedra that the approach is able to produce.

2. Approach

Let S be an oriented closed unstructured surface mesh consisting of planar triangles. It is desired that we produce a volume mesh V consisting of tetrahedra that is bounded by S. Herein we focus on a surface derived from imaging data. Thus, S is a volumetric constraint or isosurface produced by the Marching Cubes algorithm [15,19], rather than a surface of known geometry. As such, it has finite resolution and a minimum resolvable feature size of a single voxel. In order to produce a mesh that is useful for numerical analysis, we perform the operations of smoothing, refinement and de-refinement to S, subject to that volumetric constraint by limiting perturbations to S to a small fraction of a voxel. In practice, this assures that the geometry is unchanged within experimental uncertainty and that the topology is respected.

We begin this remeshing of the isosurface with several iterations of volume-conserving smoothing ([13] Algorithm 4) which sweeps through nodes of the surface mesh, and for each node performs a Laplacian smoothing followed by node repositioning to exactly maintain the volume enclosed by the surface mesh. In Fig. 1, we see both the jagged features of a Marching-Cubes-generated isosurface derived from magnetic-resonance data (A,B), followed by the result of applying volume-conserving smoothing (C,D). The volume of the smoothed object has been conserved to round-off error and the motion of the surface is a fraction of a voxel.

We allow S to consist of one large enclosing closed surface S_0 and possibly any number $S_1, \ldots, S_n$ of closed surface "holes" inside of S_0. Any of the surfaces $S_0, S_1, \ldots, S_n$ may have "handles" (have nonzero genus). In the biomedical applications of interest to us, we desire each cross section of V to have a certain number M of layers of elements across (see Fig. 2). We thus desire a "scale-invariant" meshing algorithm where a cross-section with diameter d will be filled by M layers of mesh, independent of d, as long as d is within a range of feature sizes that is bracketed away from zero:

$$0 < L_{\min} \leqslant d \leqslant L_{\max}. \tag{1}$$

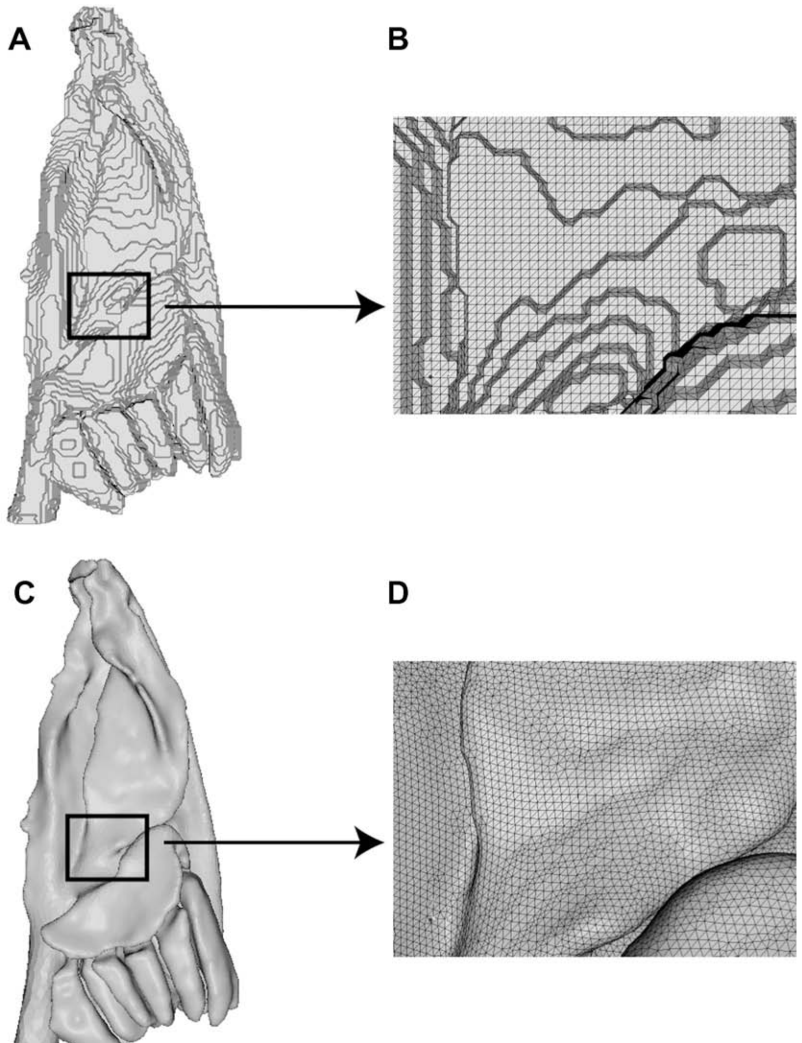


Fig. 1. Marching Cubes derived isosurface, before and after volume-conserving smoothing.

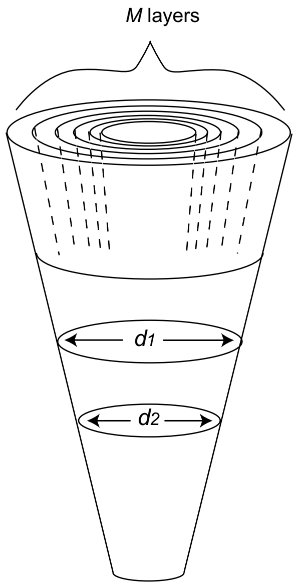


Fig. 2. *M* layers across cross-sections.

If $d < L_{\min}$, we allow a mesh cross-section to receive less than M layers; if $d > L_{\max}$, we allow a mesh cross-section to receive more than M layers (1) thus defines our "zone of invariance". We assume $L_{\min} > 0$, because the resolution of our data is one voxel; $L_{\min}$ is usually a specified fraction of a voxel width.

We now make (1) more precise, by specify how d, the "feature size" is to be defined and evaluated.

2.1. Definition of local diameter

Usually the feature size at a point $\mathbf{x}$ on a surface S is defined as the distance from $\mathbf{x}$ to the closest point of the medial axis of the volume V bounded by S [1,21]. Here the medial axis is the set of points in V defined by

$$\mathbf{y} \in \text{medial axis of } V \iff \mathbf{y} \text{ has more than one closest point } \mathbf{z} \in S. \tag{2}$$

Fig. 3 shows the pitfalls of this definition. If S is the surface of a sphere, the medial axis of V, the solid sphere bounded by S, is the single point at the center of V. The feature size at any point $\mathbf{x} \in S$ would thus be equal to the radius of the sphere. However, if the surface S is instead represented as a triangular faceted surface (e.g. make S the largest icosahedron that fits inside the original sphere), then the medial axis of the solid bounded by S consists of points on the planes that bisect the solid angles between adjacent pairs of facets. Then points $\mathbf{x}$ on S will be deemed to have feature sizes ranging from zero (if $\mathbf{x}$ is on a boundary edge of S) to r_{ins}, where r_{ins} is the largest inscribed radius of any facet in S. Thus, the mere act of discretizing the geometry by representation by piecewise linear facets has caused a violent change in the definition of "feature size".

In contrast, we wish to define a feature size field that is more robust with respect to perturbations of the geometry and which does not require construction of the medial axis. Our feature size will be roughly twice that computed using the pre-

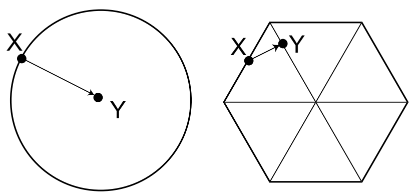


Fig. 3. Difficulty with medial-axis based definition of feature size. A small alteration in a figure changes a one-point medial axis (center of circle on left figure) to a complex medial axis (angle bisectors on right figure). This causes a great change in feature size if feature size at **x** is defined to be distance from **x** to the nearest point on the medial axis (**y**).

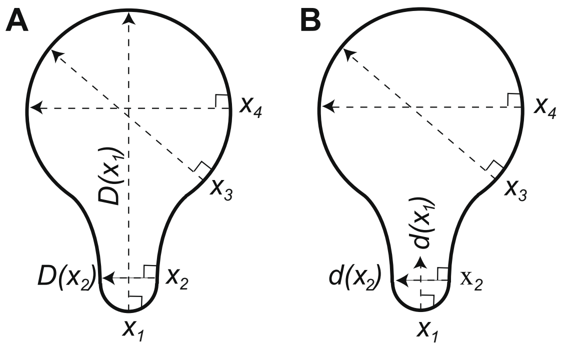


Fig. 4. (A) Definition of "local diameter" $D(\mathbf{x})$ is length to first re-intersection with surface of a ray normally cast inwards from $\mathbf{x}$. (B) Gradient-limiting of $D(\mathbf{x})$ field yields $d(\mathbf{x})$ field which removes effect of "lucky rays" that travel far further than twice the distance to the medial axis.

vious definition, in the case that the geometry is smooth and free of low amplitude undulations. In addition, it will be robust to perturbations.

For any point $\mathbf{x} \in S$, we define the "raw feature size" or "local diameter" $D(\mathbf{x})$ as the length of the line segment formed by shooting a ray from $\mathbf{x}$ in the direction of $\hat{\mathbf{n}}(\mathbf{x})$, the inward normal at $\mathbf{x}$, and truncating the ray at the first new intersection with S. That is

$$D(\mathbf{x}) \equiv \min\{\lambda > 0 \mid \mathbf{x} + \lambda\hat{\mathbf{n}}(\mathbf{x}) \in S\}. \tag{3}$$

See Fig. 4(A). In the case that S is differentiable at $\mathbf{x}$, and thus the classical normal exists, we have that points on the segment $\mathbf{x} + \lambda\hat{\mathbf{n}}(\mathbf{x})$ for very small $\lambda > 0$ have a single closest point on the boundary, namely $\mathbf{x}$. When $\lambda = \frac{D(\mathbf{x})}{2}$, $\mathbf{x} + \lambda\hat{\mathbf{n}}(\mathbf{x})$ has equal distance to $\mathbf{x}$ and the boundary point $\mathbf{x} + D(\mathbf{x})\hat{\mathbf{n}}(\mathbf{x})$. If the ray does not intersect the medial axis at a point before $\mathbf{x} + \frac{D(\mathbf{x})}{2}\hat{\mathbf{n}}(\mathbf{x})$, we would have to conclude that $\mathbf{x} + \frac{D(\mathbf{x})}{2}\hat{\mathbf{n}}(\mathbf{x})$ is on the medial axis. We thus conclude that the distance from $\mathbf{x}$ to the medial axis is bounded above by $\frac{D(\mathbf{x})}{2}$.

In the case that S is differentiable at $\mathbf{x}$, the classical normal exists and so the choice of $\hat{\mathbf{n}}$ is unambiguous. The usual case, however, is that S is piecewise linear, and that in fact $\mathbf{x}$ is a vertex where the normal is not classically defined. In this case, we define a "synthetic normal" at $\mathbf{x}$ as follows. Consider the neighborhood

$$\mathcal{N}_\epsilon(\mathbf{x}) \equiv B_\epsilon(\mathbf{x}) \cap S, \tag{4}$$

where $B_\epsilon(\mathbf{x})$ is the solid ball of radius ϵ centered at $\mathbf{x}$. We note that in the case that $\mathbf{x}$ is a vertex in the piecewise linear geometry of S, that the classical normal is defined everywhere in $\mathcal{N}_\epsilon(\mathbf{x})$ except for a set of surface measure zero. That is because $\mathcal{N}_\epsilon(\mathbf{x})$ is a piece of surface near $\mathbf{x}$, while the only points $\mathbf{y} \in S$ that the classical normal $\hat{\mathbf{n}}(\mathbf{y})$ is not defined on are those points that reside on a set of surface edges incident upon $\mathbf{x}$. With this in mind we define the "synthetic" normal as

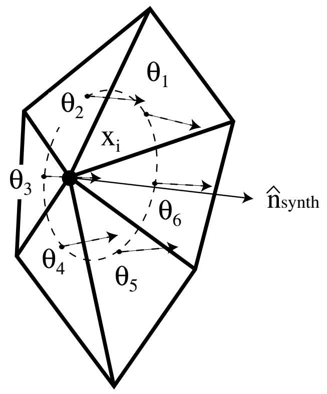


Fig. 5. The synthetic normal at a vertex i of a piecewise linear surface is the angle-weighted normal.

$$\hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}) \equiv \lim_{\epsilon\to 0} \frac{\int_{\mathbf{y}\in\mathcal{N}_\epsilon(\mathbf{x})} \hat{\mathbf{n}}(\mathbf{y})dS}{\| \int_{\mathbf{y}\in\mathcal{N}_\epsilon(\mathbf{x})} \hat{\mathbf{n}}(\mathbf{y})dS\|}. \tag{5}$$

The integral in the numerator of the right-hand side is well-defined (since the integrand is defined almost everywhere) and indeed the limit exists when the patches of surface incident upon $\mathbf{x}$ are smooth. We note trivially that if the normal is classically defined at $\mathbf{x}$, then $\hat{\mathbf{n}}_{\text{synth}}$ is equal to the classical normal and if $\mathbf{x}$ is a vertex of a piecewise linear surface, then

$$\hat{\mathbf{n}}_{\text{synth}} = \frac{\sum_i \theta_i \hat{\mathbf{n}}_i}{\|\sum_i \theta_i \hat{\mathbf{n}}_i\|}, \tag{6}$$

where the sum is over the triangles T_i sharing $\mathbf{x}$, the $\hat{\mathbf{n}}_i$ are the normals of the T_i, and θ_i are the vertex angles at $\mathbf{x}$ for each of the T_i. We can thus say that for piecewise linear surfaces, the "synthetic normal" is an "angle-weighted" normal [27] (Fig. 5). Since (5) indicates the synthetic normal is actually an intrinsic property of the surface independent of triangulation, we have that it would not change much if surface deformation due to refinement/de-refinement is sufficiently small. Although this normal is very robust [4], we doublecheck $\hat{\mathbf{n}}_{\text{synth}}$ actually points "into" the geometry by checking it makes a positive dot product with all the $\hat{\mathbf{n}}_i$. (In the highly unlikely case it does not, a value $D(\mathbf{x}) = L_{\max}$ is assigned which will be reduced to a value comparable with the feature size at neighboring nodes by the gradient-limiting operation to be described.)

So, choosing $\hat{\mathbf{n}}$ to be the synthetic normal, the ray proceeding from $\mathbf{x}$ in the direction $\hat{\mathbf{n}}_{\text{synth}}$ will again intersect S at least once (since S is assumed closed) and $D(\mathbf{x})$ is well-defined.

Note: We also perform an "outwards" interrogation of the geometry by computing another $D(\mathbf{x})$ at $\mathbf{x}$ using (3) using $\hat{\mathbf{n}} = -\hat{\mathbf{n}}_{\text{synth}}$. This outwards value (which we call $D_{\text{out}}(\mathbf{x})$) is only finite in some areas at most (e.g. where S is concave) and is used to ensure the topology of the final volume mesh agrees with the topology of the input surface mesh S in those regions. Use of $D_{\text{out}}(\mathbf{x})$ is discussed in Section 2.3.

Having defined the raw feature size by interrogation of the geometry (3), we then enforce the restriction (1) by performing 'max' and 'min' operations with $L_{\min}$ and $L_{\max}$, respectively. To be clear, we restate that $L_{\min}$ and $L_{\max}$ are user parameters, bounded below by a fraction of a voxel and bounded above by the largest dimension of the geometry. The last step in computing a local feature size is now to modify $D(\mathbf{x})$ so that we obtain a smooth function of $\mathbf{x}$ and S – so that feature size won't change much if the surface point $\mathbf{x}$ or if the surface S itself is perturbed slightly. This is accomplished by performing a gradient-limiting procedure similar to that in [21].

In [21], the author defines a feature size function over a surface mesh, extends the definition to the entire volume enclosed by the surface, and then performs gradient-limiting on the function. Given a desired bound G on the gradient of a feature size function d defined on a volume V, d is limited by the following update:

$$d(\mathbf{x}) \leftarrow \min_{\mathbf{y}}(d(\mathbf{y}) + G \cdot \text{dist}(\mathbf{x}, \mathbf{y})). \tag{7}$$

Here $\text{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ is the Euclidean distance in $\mathbb{R}^3$. The update (7) is performed repeatedly on a discrete Cartesian grid until convergence occurs. Clearly the gradient-limited d is less than or equal to the original non-gradient-limited d and it obeys

$$|d(\mathbf{x}) - d(\mathbf{y})| \leqslant G\|\mathbf{x} - \mathbf{y}\|, \tag{8}$$

which means that d is Lipschitz continuous, with Lipschitz constant G. Since we are on a discrete mesh, this essentially amounts to delivering the desired bound $\|\nabla d\| \leqslant G$.

In contrast, we perform the gradient-limiting operation (7) on $d(\mathbf{x})$ which is initialized with the values of $D(\mathbf{x})$ on the surface and is *not* extended to the volume. Instead $\text{dist}(\mathbf{x}, \mathbf{y})$ is to be interpreted as the distance between $\mathbf{x}$ and $\mathbf{y}$ measured on the surface of S– the geodesic distance. Our algorithm (Algorithm 1) is performed on the unstructured surface mesh, rather than a Cartesian volume mesh. The algorithm involves placing directed surface edges $\overrightarrow{\mathbf{x}_1\mathbf{x}_2}$ of S into a priority queue [5] ranked by

$$d(\mathbf{x}_1) - (d(\mathbf{x}_2) + G\|\mathbf{x}_1 - \mathbf{x}_2\|),$$

which is how much the gradient over the directed edge of d violates the gradient limit. (Only directed edges for which this quantity is positive are placed in the queue.) The directed edge is then relaxed to satisfy the gradient limit and then all di-

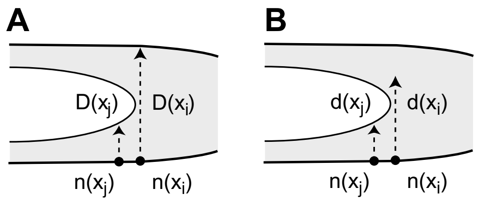


Fig. 6. Without gradient-limiting $D(\mathbf{x}_i)$ can discontinuously jump with a small perturbation in the surface S due to borderline clipping of local diameter ray. With gradient-limiting, this jump in $d(\mathbf{x}_i)$ is limited to $G\|\mathbf{x}_i - \mathbf{x}_j\|$, where $\mathbf{x}_j$ is a neighbor whose local diameter ray is robustly clipped.

rected edges in the neighborhood are added, deleted, or re-ranked in the priority queue as necessary. The process continues until the queue is empty and hence the field $d(\mathbf{x})$ is gradient-limited.

Algorithm 1: Processing of raw local diameter field

```
[Process local diameter field to be within limits and be gradient-limited so that it is an acceptable feature size field]

[Copy raw local diameter field to $d$]
For each surface node $i$
    $d(\mathbf{x}_i) \leftarrow D(\mathbf{x}_i)$

[Force $d(\mathbf{x})$ to lie between $L_{\min}$ and $L_{\max}$]
For each surface node $i$
    $d(\mathbf{x}_i) \leftarrow \min(L_{\max}, d(\mathbf{x}_i))$
    $d(\mathbf{x}_i) \leftarrow \max(L_{\min}, d(\mathbf{x}_i))$

[Reduce $d(\mathbf{x})$ in tight concave areas (See Section 2.3).]
For each surface node $i$
    $d(\mathbf{x}_i) \leftarrow \min\left(d(\mathbf{x}_i), 2\frac{M}{K}D_{\text{out}}(\mathbf{x}_i)\right)$

[Decrease $d(\mathbf{x})$ to ensure gradient of $d$ along edges is less than $G$]
For each node $i$
    For each node neighbor $j$
        $\mathbf{excess}(i,j) \leftarrow d(\mathbf{x}_i) - (d(\mathbf{x}_j) + G\|\mathbf{x}_i - \mathbf{x}_j\|)$
        If $\mathbf{excess}(i,j) > 0$ Then
    [Priority queue $P$ will contain directed edges with positive excess values]
            Place directed edge $\overrightarrow{\mathbf{x}_i\mathbf{x}_j}$ in priority queue $P$
While $P$ nonempty do
    Pop $\overrightarrow{\mathbf{x}_i\mathbf{x}_j}$ with maximal $\mathbf{excess}(i,j)$ from $P$
    [Decrease $d(\mathbf{x}_i)$ to conform to gradient limit]
    $d(\mathbf{x}_i) \leftarrow (d(\mathbf{x}_j) + G\|\mathbf{x}_i - \mathbf{x}_j\|)$
    For all directed edges $\overrightarrow{\mathbf{x}_i\mathbf{x}_k}$ emanating from $i$
    Recompute $\mathbf{excess}(i,k) \leftarrow d(\mathbf{x}_i) - (d(\mathbf{x}_k) + G\|\mathbf{x}_i - \mathbf{x}_k\|)$
    Add, delete, or rerank $\overrightarrow{\mathbf{x}_i\mathbf{x}_k}$ in $P$ as appropriate
    For all directed edges $\overrightarrow{\mathbf{x}_k\mathbf{x}_i}$ incident on $i$
    Recompute $\mathbf{excess}(k,i) \leftarrow d(\mathbf{x}_k) - (d(\mathbf{x}_i) + G\|\mathbf{x}_i - \mathbf{x}_k\|)$
    Add, delete, or rerank $\overrightarrow{\mathbf{x}_k\mathbf{x}_i}$ in $P$ as appropriate
```

One question is what is a good value for G? In fact, it can be argued that a *natural* bound on G is $\mathcal{O}(1)$. That is because our "feature size" is the length of a normal ray traversing a geometry, and for smooth geometries this is essentially double the distance from the boundary point $\mathbf{x}$ to the medial axis – and a natural bound on the gradient of this distance is 1. This latter assertion can be seen as follows. Let $d_{\text{MA}}(\mathbf{x})$ be the distance from $\mathbf{x} \in S$ to the closest point on the medial axis of V. That is

$$d_{\text{MA}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{P}_{\text{MA}}(\mathbf{x})\|, \tag{9}$$

where $\mathbf{P}\mathbb{R}(\mathbf{x})$ is the closest point on the medial axis to $\mathbf{x}$. Let Δ be an arbitrary change in $\mathbf{x}$ such that $\mathbf{x} + \Delta$ is also on S. Then

$$\begin{aligned} d_{\text{MA}}(\mathbf{x}+\Delta) &= \|\mathbf{x}+\Delta - \mathbf{P}_{\text{MA}}(\mathbf{x}+\Delta)\| \\ &\leqslant \|\mathbf{x}+\Delta - \mathbf{P}_{\text{MA}}(\mathbf{x})\| \\ &\leqslant d_{\text{MA}}(\mathbf{x}) + \|\Delta\|. \end{aligned}$$

Similarly,

$$d_{\text{MA}}(\mathbf{x}) \leqslant d_{\text{MA}}(\mathbf{x}+\Delta) + \|\Delta\|.$$

So

$$|d_{\text{MA}}(\mathbf{x}+\Delta) - d_{\text{MA}}(\mathbf{x})| \leqslant 1 \cdot \|\Delta\|.$$

So $d_{\text{MA}}(\mathbf{x})$ is Lipschitz continuous with Lipschitz constant no larger than 1. It is quite easy to construct a geometry where the Lipschitz constant of $d_{\text{MA}}(\mathbf{x})$ is arbitrarily close to 1. Thus, $d_{\text{MA}}(\mathbf{x})$ is Lipschitz continuous over all geometries with uniform Lipschitz constant 1. Since our function $d(\mathbf{x})$, the "local diameter" at $\mathbf{x}$ typically behaves like $2d_{\text{MA}}(\mathbf{x})$ (in the case of smooth geometries), we have that "2" would be a natural Lipschitz constant to enforce for $d(\mathbf{x})$. In practice, we have found increasing the amount of gradient-limiting further is desirable for producing tighter, more disciplined layers of volume mesh parallel to the surface mesh in the volume mesh generation phase. For our runs, we have used $G = 0.85$ which is a value that has worked well over all geometries.

Our resulting gradient-limited feature size or "local diameter" function $d(\mathbf{x})$ defined over surfaces is now a continuous function of $\mathbf{x}$ (Fig. 4(B)). If S is perturbed slightly, a ray shot from a vertex $\mathbf{x}_i$ on the surface might transition from being clipped to not being clipped by an interceding piece of geometry (as in Fig. 6), but the fact that $d(\mathbf{x})$ is gradient-limited will limit the jump in $d(\mathbf{x}_i)$ to being bounded by $G\|\mathbf{x}_i - \mathbf{x}_j\|$ where $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$ that already "sees" the interceding geometry. (*Note*: this argument breaks down if $\mathbf{x}_i$ possesses no neighbor $\mathbf{x}_j$ that sees the interceding geometry, but this is unlikely if the surface reasonably resolves the geometry. In the following sections we show how a good discretization of S by a surface grid is found and then a final "stable" $d(\mathbf{x})$ can be computed on this improved surface discretization if necessary.) Since the jump in $d(\mathbf{x})$ is usually bounded by a small multiple of the surface discretization edge length, it is reasonable to say that $d(\mathbf{x})$ behaves "smoothly" with respect to perturbations in the geometry S.

Although it would appear that it is a daunting task to construct $D(\mathbf{x})$ by shooting rays from all surface vertices and detecting all subsequent first intersections of these rays with S, it is in fact a rapid process. For this we employ an axis-aligned bounding box (AABB) tree [25,12] that contains at its leaf nodes bounding boxes for each of the N_{surf} triangles discretizing S. This is a balanced binary tree where each node is assigned the smallest Cartesian axis-aligned bounding box that contains the bounding boxes of its two children; this structure can be constructed in $\mathcal{O}(N_{\text{surf}} \log N_{\text{surf}})$ time. The root node is thus assigned a bounding box for the whole geometry. When we seek intersections of a ray with S, we thus intersect the ray with the root bounding box and refine the query by asking which of the child boxes intersect the ray. We then query the child boxes of only those boxes that are found to intersect the ray. (The ray-box intersection test is quickly done.) We continue on in this way until in $\mathcal{O}(\log N_{\text{surf}})$ operations we obtain the leaf boxes of all surface triangles that could possibly intersect the ray. We then simply compute the exact distance along the ray with the closest of the candidate triangles so found to intersect the ray. The operation for computing $D(\mathbf{x})$ thus has complexity $\mathcal{O}(N_{\text{surf}} \log N_{\text{surf}})$.

2.2. *Generation of a surface mesh discretization commensurate with $d(\mathbf{x})$*

Let K be the desired anisotropy of layers that we wish to generate near the boundary of S. That is, we wish the average surface triangle edge length at a point $\mathbf{x}$ on S to be K times the normal thickness of tetrahedra in layers parallel to the surface. At the same time, we wish to establish M layers across the cross-section of the geometry. The resolution of these two requirements is to make the edge length of the discretization of S at $\mathbf{x}$ be proportional to the feature size $d(\mathbf{x})$ at $\mathbf{x}$. Indeed, if we generate M layers of mesh across the "local diameter" $d(\mathbf{x})$, and the surface edge length in the neighborhood of $\mathbf{x}$ is $\frac{K}{M}d(\mathbf{x})$, then the ratio of the base triangle edge length to the altitude of tetrahedra generated at $\mathbf{x}$ will be $\frac{K}{M}d(\mathbf{x})/\frac{1}{M}d(\mathbf{x}) = K$ (assuming equal spacing of the M layers). This means that the tetrahedra generated on the surface layers will have constant anisotropy K independent of $\mathbf{x}$.

Algorithm 2: Edge bisection

```
[Algorithm refines mesh by bisecting edges exceeding a variable node-based bisection length tolerance]

Find set of edges $E = \{e = \overline{\mathbf{x}_1\mathbf{x}_2} | \text{length}(e) > \min(\texttt{bisection\_length}(\mathbf{x}_1), \texttt{bisection\_length}(\mathbf{x}_2))\}$
While $E \neq \emptyset$ do
    Sort $E$ in decreasing order of (Euclidean) length
    For each $e \in E$ do
        Refine terminal member of Rivara chain spawned by $e$
    Recompute E
```

Surface modification as a function of local diameter consists of (1) Rivara edge bisection operations to establish a maximum edge length $\frac{K}{M}d(\mathbf{x})$ over the whole surface mesh, (2) node merge operations, and (3) edge swap operations.

In regions where the edges are longer than $\frac{K}{M}d(\mathbf{x})$, we bisect edges (see Fig. 7), doubling the number of triangles incident on the refined edge. We use the principle due to Rivara [23] that restricts edge bisection to those edges that are the longest edges in their neighborhood which consists of all of the edges in all the elements that share the edge in question. If an edge is not the longest edge in its neighborhood, we consider refining the edge that is the longest edge in the neighborhood as a precondition for refining the desired edge. Continuing recursively in this fashion, a "Rivara" chain is constructed which ends in a terminal edge that is in fact the longest edge in its neighborhood and which can be bisected. Continued attempts to refine an edge will result in the refinement of a certain number of terminal edges until the original edge is eventually refined. The effect of the refinement of the additional edges is a stable refinement that tends not to degrade element quality. The edge bisection algorithm is given in Algorithm 2. `bisection_length` as stated in the algorithm is $\frac{K}{M}d(\mathbf{x})$.

Algorithm 3: Node merging

```
[Algorithm de-refines mesh by merging nodes to neighboring nodes where appropriate]

Do until nothing changes
    For each node $i$
        Order node neighbors $j$ by increasing distance from $i$
        For each neighbor $j$ of $i$
            If $\|\mathbf{x}_i - \mathbf{x}_j\| <$ merge_length Then
                If $\mathbf{damage}(\mathbf{x}_i \to \mathbf{x}_j) <$ toldamage Then
                    If $\mathbf{minimum\ inscribed\ radius}(\{T_k^{\text{new}}\}) \geqslant \frac{1}{2} \cdot \mathbf{minimum\ inscribed\ radius}(\{T_k\})$ Then
                      merge($\mathbf{x}_i \to \mathbf{x}_j$)
                      break out of inner For loop
```

The node merging algorithm is given in Algorithm 3. We sweep through the mesh, considering nodes i and whether it is advisable to merge them into their neighbors j. (See Fig. 8.) If a merge $\mathbf{x}_i \to \mathbf{x}_j$ would take place, the triangles $\{T_k\}$ containing node i would be replaced by a new set of triangles $\{T_k^{\text{new}}\}$ all containing node j. After a sufficient number of passes, the surface mesh will be de-refined where needed. Nodes are attempted to be merged if edge lengths are less than one-half the local target edge length in the neighborhood of node i. That is, merge_length is set to $\frac{K}{2M} d(\mathbf{x}_i)$. Setting merge_length equal to a fraction significantly less than the desired length prevents "thrashing" where node merge and edge bisection operations pointlessly alternate in the same region. This has the effect that observed anisotropies of elements will actually be in a range between $\frac{K}{2}$ and K. (Layer thickness $\frac{d(\mathbf{x})}{M}$ will however vary smoothly as $d(\mathbf{x})$ is a smooth function of $\mathbf{x}$.)

The "damage" of the merge $\mathbf{x}_i \to \mathbf{x}_j$ is a measure of the deformation of the surface caused by the merge. Our estimate of surface deformation (the "damage" function) is given in the Appendix. If the damage of a merge is anticipated to cause a damage exceeding 1% of the bisection length $\left(\frac{K}{100M} d(\mathbf{x}_i)\right)$, the damage is considered unacceptable and the merge operation $\mathbf{x}_i \to \mathbf{x}_j$ is rejected. For a given node i, there may be one or more neighboring nodes j that produce an acceptable damage estimate, even if some neighboring nodes j would result in unacceptable damage. The new triangulation T_k^{new} produced by an $\mathbf{x}_i \to \mathbf{x}_j$ merge depends on which neighbor j is chosen and we only accept a new triangulation if the minimum inscribed radius in $\{T_k^{\text{new}}\}$ would be at least one-half as big as the minimum inscribed radius of the original triangulation $\{T_k\}$. (If the minimum inscribed radius of the original triangulation is especially small, this requirement is tightened further.)

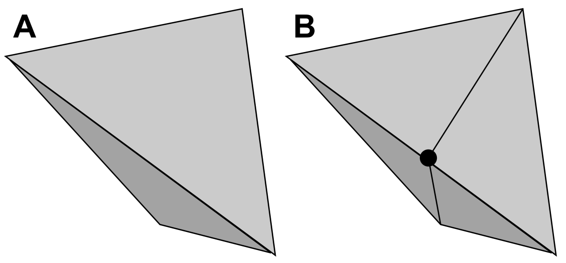


Fig. 7. Refinement of edge. (a) Two triangles that share an edge. (b) Two triangles have become four after bisection of edge.

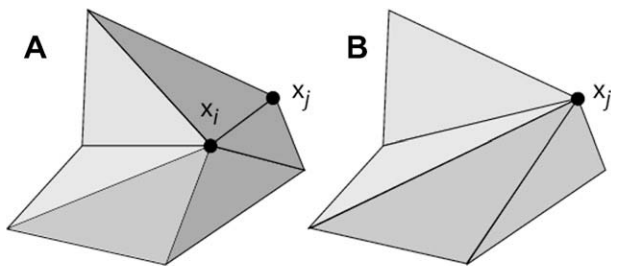


Fig. 8. Merging of node i at $\mathbf{x}_i$ to neighboring node j at $\mathbf{x}_j$.

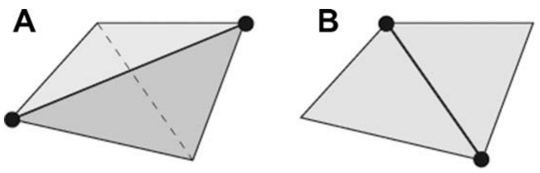


Fig. 9. Surface edge swap: swapping mutual edge of two adjacent surface triangles.

Note that our node merge operation differs from the standard procedure presented in [6]. The standard algorithm greedily contracts edges in ascending order of damage on a priority queue. Here we aspire for a target minimum edge length $\frac{K}{2M}d(\mathbf{x}_i)$ and greedily contract the shortest edges less than this length, stopping only if the damage caused by the single contraction or the resulting grid quality is unacceptable.

Edge swap operations (Fig. 9) attempt to maximize the minimum vertex angle of a candidate pair of adjacent surface triangles. We repeatedly sweep through candidates in the mesh and perform swaps if the minimum of the six vertex angles is increased by swapping their common edge, provided that the damage to the surface by such a swap is deemed acceptable. (As in the case of node merging, the damage should not exceed $(\frac{K}{100M}d(\mathbf{x}_i))$ and the damage is similarly estimated.)

Refinement and de-refinement of the surface alternates node merging with edge swapping until neither operation changes the mesh, and then ends with edge refinement. This modification of the surface is repeatedly called and alternated with volume-conserving smoothing.

After several such cycles, the surface mesh has been suitably processed so that the edge length at a node $\mathbf{x}_i$ on the surface is roughly $\frac{K}{M}d(\mathbf{x}_i)$..

Algorithm 4: Offset points

```
[Algorithm computes layers of points in volume $V$ offset from the surface points of $S$]

For each node $i \in S$
  [Surface points become 'layer 0' volume points]
  $\mathbf{x}_i^0 \leftarrow \mathbf{x}_i$
  dist$\leftarrow 0$
  [Add points along normal ray from surface out to half raw local diameter. For uniform spacing, fractional distance
  between points $\lambda_m = \frac{1}{M}$, and approximately $\frac{M}{2}$ points will be layed down]
  Do while $m = 1, 2, \ldots$
        If $m \leqslant \lfloor \frac{M}{2} \rfloor)$ then
              dist $\leftarrow$ dist $+ \lambda_m d(\mathbf{x}_i)$
        Else
              dist $\leftarrow$ dist $+ \frac{1}{M}\min(D(\mathbf{x}_i), L_{\max})$
        If dist $\leqslant \frac{D(\mathbf{x}_i)}{2}$ then
              $\mathbf{x}_i^m \leftarrow \mathbf{x}_i +$ dist$\hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)$
        Else
              Exit Do loop on $m$
```

2.3. Generation of a volume mesh from the processed surface mesh

Once a suitable surface mesh has been obtained with edge length on the surface of order $\frac{K}{M}d(\mathbf{x}_i)$ at point i, we cast m_i points in the direction $\hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)$ (Algorithm 4) forming the point set

$$\{\mathbf{x}_i^m | 1 \leqslant i \leqslant N_{\text{surf}}, 0 \leqslant m \leqslant m_i\}, \tag{10}$$

where here we have defined $\mathbf{x}_i^0 \equiv \mathbf{x}_i$ to be the surface points in S.

To explain the rationale of Algorithm 4, we provide the following argument. If we distribute $\lfloor \frac{M}{2} \rfloor$ points normally from $\mathbf{x}_i$ to $\mathbf{x}_i + \frac{1}{2}d(\mathbf{x}_i)\hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)$, spacing them out equally (that is using spacing $\frac{d(\mathbf{x}_i)}{M}$ between points on the normal ray), they will naturally form layers. Indeed, for two points $\mathbf{x}_i, \mathbf{x}_j$ on the surface that are close together and connected by an edge in S, we have $d(\mathbf{x}_i) \approx d(\mathbf{x}_j)$ due to gradient-limiting. Thus for small m, points $\mathbf{x}_i^m$ and $\mathbf{x}_j^m$ will be approximately the same distance normal from the surface and will therefore have a high probability of being connected by an edge as well. We call these points between $\mathbf{x}_i$ and $\mathbf{x}_i + \frac{1}{2}d(\mathbf{x}_i)\hat{\mathbf{n}}(\mathbf{x}_i)$ "layer-making" interior points (More generally, we can choose the distance between $\mathbf{x}_i^{m-1}$ and $\mathbf{x}_i^m$ to be some fraction $\lambda_m d(\mathbf{x}_i)$ with $\sum_{m=1}^{\frac{M}{2}} \lambda_m = \frac{1}{2}$, with λ_m monotonically increasing to create layers with closer spacing at the boundary).

However, due to gradient-limiting, $d(\mathbf{x}_i)$ may be quite reduced from the raw local diameter $D(\mathbf{x}_i)$ defined in Equation (3). Thus more points based on $D(\mathbf{x}_i)$ rather than $d(\mathbf{x}_i)$ may have to be laid down in places. These points will be termed "filler" interior points.

Consider

$$L(\mathbf{x}) = \left\{ \mathbf{y} \mid \mathbf{y} = \mathbf{x} + \lambda\hat{\mathbf{n}}(\mathbf{x}), 0 \leqslant \lambda \leqslant \frac{1}{2}D(\mathbf{x}) \right\}.$$

We call $L(\mathbf{x})$ a *semi-diameter segment associated with the surface point* $\mathbf{x}$.

Lemma 1. Coverage by Semi-diameter segments. If S is closed and $\mathcal{C}^2$ and thus possesses a classical normal at each point, then V (the volume bounded by S) is covered by all the semi-diameter segments associated with all points of S. That is

$$V = \cup_{\mathbf{x} \in S} L(\mathbf{x}).$$

Proof 1. Clearly for each $\mathbf{x}$, $L(\mathbf{x}) \subseteq V$, so $V \supseteq \cup_{\mathbf{x}\in S} L(\mathbf{x})$. Clearly $S \subseteq \cup_{\mathbf{x}\in S} L(\mathbf{x})$. Now let $\mathbf{y}$ be any point in the interior of V. Let $\mathbf{x}$ be the closest point on S to $\mathbf{y}$. That is, choose some $\mathbf{x}_c$ such that $\|\mathbf{y} - \mathbf{x}_c\| = \min_{\mathbf{x}\in S}\|\mathbf{y} - \mathbf{x}\|$. Then we claim $\mathbf{y} \in L(\mathbf{x}_c)$. First, it must be true that $(\mathbf{y} - \mathbf{x}_c)\|\hat{\mathbf{n}}(\mathbf{x}_c)$. For otherwise, one could choose a point on S in the neighborhood of $\mathbf{x}_c$ closer to $\mathbf{y}$, contradicting our assumption that $\mathbf{x}_c$ is a closest point to Y. Suppose the ray starting from $\mathbf{x}_c$ in the direction $\hat{\mathbf{n}}(\mathbf{x}_c)$ has a first re-intersection with S at $\mathbf{x}_c^{\text{opp}}$. Then $\|\mathbf{x}_c^{\text{opp}} - \mathbf{x}_c\| = D(\mathbf{x}_c)$ by (3). We have that $\mathbf{x}_c, \mathbf{y}$, and $\mathbf{x}_c^{\text{opp}}$ are collinear and since $\mathbf{x}_c$ is a closest point to $\mathbf{y}$,

$$\|\mathbf{y} - \mathbf{x}_c\| \leqslant \frac{1}{2}D(\mathbf{x}_c).$$

Hence, $\mathbf{y} \in L(\mathbf{x}_c)$. Since $\mathbf{y}$ is an arbitrary point in the interior of V, we conclude $V \subseteq \cup_{\mathbf{x}\in S} L(\mathbf{x})$. Hence $V = \cup_{\mathbf{x}\in S} L(\mathbf{x})$ □

By the lemma, we surmise that if we spread points along all the semi-diameter segments that we would adequately cover V. This is not rigorously true if the conditions of the lemma are violated. That is, if S has a sharp corner where it has no classical normal, there could be a gap in the coverage of the interior. Strictly speaking, there is no classical normal at any vertex of the piecewise linear S, but if the number of nodes on S provide sufficient angular resolution, the idea of the lemma is effectively applicable to S.

However we had previously determined that we want to distribute points between $\mathbf{x}_i$ and $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{d(\mathbf{x}_i)}{2}$ at equal intervals in order to form layers. Due to gradient-limiting, we could have $d(\mathbf{x}_i) < D(\mathbf{x}_i)$. Also we may have adjusted $d(\mathbf{x}_i)$ so that $L_{\min} \leqslant d(\mathbf{x}_i) \leqslant L_{\max}$. Thus it is quite possible that $d(\mathbf{x}_i) \neq D(\mathbf{x}_i)$. In this case, what is the best resolution for these contradictory criteria?

In a very tight space where $D(\mathbf{x}_i) < L_{\min}$, we have that the minimum spacing between nodes should be $\frac{L_{\min}}{M}$. Thus we cover the distance from $\mathbf{x}_i$ to $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{D(\mathbf{x}_i)}{2}$ with points spaced uniformly $\frac{d(\mathbf{x}_i)}{M} = \frac{L_{\min}}{M}$ starting from the boundary. Potentially, less than $\lfloor\frac{M}{2}\rfloor$ points will be laid down (making less than M layers across) but that is a necessary consequence of our limiting the feature size to a lower bound of $L_{\min}$.

If $d(\mathbf{x}_i) < D(\mathbf{x}_i)$, we must place $\lfloor\frac{M}{2}\rfloor$ points between $\mathbf{x}_i$ and $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{d(\mathbf{x}_i)}{2}$ because the gradient-limited $d(\mathbf{x})$ field varies smoothly along the surface; placing them instead from $\mathbf{x}_i$ to $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{D(\mathbf{x}_i)}{2}$ would cause significant jumps in the distances from boundary points to corresponding interior points at some adjacent boundary nodes. However to guarantee filling out of the interior, we must add additional points between $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{d(\mathbf{x}_i)}{2}$ and $\mathbf{x}_i + \hat{\mathbf{n}}_{\text{synth}}(\mathbf{x}_i)\frac{D(\mathbf{x}_i)}{2}$. In this interior region, we consider the feature size to be $\min(D(\mathbf{x}_i), L_{\max})$ and add these "filler" interior points with spacing $\frac{1}{M}\min(D(\mathbf{x}_i), L_{\max})$ to fill out the semi-diameter segment.

The points generated in Algorithm 4 are handed off to the Delaunay mesh generator described in Algorithm 5. A "big tet" is made which encloses the region to be meshed in $\mathbb{R}^3$ and the points $\{\mathbf{x}_i^m\}$ are inserted layer by layer using the incremental point insertion algorithm of Watson [28]. In the Watson algorithm, the insertion of a new point $\mathbf{x}_i^m$ leads to removal of all tetrahedra whose circumspheres contain the point. (When the tetrahedra are removed a simply connected "insertion cavity" is formed and each of the triangular bounding facets of the cavity are visible to the new point.) In the standard Watson algorithm, $\mathbf{x}_i^m$ is connected to each of the facets bounding the cavity to form a set of tetrahedra that fill the cavity and the procedure continues with insertion of the next point in the insertion list. In our version of the insertion algorithm, we abort the insertion of $\mathbf{x}_i^m$ if $m > \lfloor\frac{M}{2}\rfloor$ – that is, if $\mathbf{x}_i^m$ is a "filler" interior point greater than a distance $\frac{d(\mathbf{x}_i)}{2}$ but less than a distance $\frac{D(\mathbf{x}_i)}{2}$ from $\mathbf{x}_i$ – if it would result in the formation of a tetrahedron containing an edge that would connect $\mathbf{x}_i^m$ to a boundary point $\mathbf{x}_j^0$ on S. The reason is that filler interior points can be on "lucky" rays that travel far from the originating surface point (recall Fig. 4(A)) and in particular they might pass very close to the surface of S in a totally unexpected location and consequently degrade the quality of or even ruin the triangulation near the boundary. Indeed the purpose of filler interior points is to fill gaps in the interior and they should not connect to the boundary. After running through the list once, we run through it a couple of more times to be sure previously rejected points can't be eventually inserted.

The final step of Algorithm 5 is rejection of tetrahedra that fall *outside* of the bounding surface S. We simply delete all tetrahedra that do not have an internal point (i.e., a point $\mathbf{x}_i^m$, $m \geqslant 1$). It is however possible that a tetrahedron could be formed that connects an internal point to a "layer 0" point across a small gap where boundary points are misaligned. We prevent this by increasing the surface node density in such problem areas. This is accomplished as follows. The field D_{out} computed in Section 2.1 is computed at the same time as the inwards pointing rays are computed. The purpose of D_{out} is not to influence the size of elements, but to only prevent "connect-across" situations like in Fig. 10 occurring. We have found that in these kinds of regions where separate pieces of geometry lie close together, it is necessary to have the surface edge length less than $2D_{\text{out}}(\mathbf{x})$ in order to prevent these situations. Since we create a mesh with characteristic edge length $\frac{K}{M}d(\mathbf{x}_i)$ using the refinement, de-refinement and swap operations previously mentioned, we thus desire

$$\frac{K}{M}d(\mathbf{x}_i) \leqslant 2D_{\text{out}}(\mathbf{x}) \tag{11}$$

To force this, in Algorithm 1, we accordingly reduce $d(\mathbf{x}_i)$ where necessary so that (11) holds with equality before the final gradient-limiting operation of the feature size field d. This can in principle lead to spots where $d(\mathbf{x}_i) < L_{\min}$, but in this case it is more important to prevent incorrect grid topology at the expense of having some small cells.

We note that this increase in surface density to recover surface topology under Delaunay triangulation is an issue addressed by Amenta et al. [1,2]. In [1] a high surface point sampling density is required for a rigorous proof that correct surface topology will be recovered under 3-D Delaunay triangulation. In [2] a much lower density is shown to work in practice. Our required point density (as forced by (11)) is also much lower and we attribute this to the fact that our point sets are isotropically distributed on the surface and then stacked along the normals in the volume. This is a highly advantageous configuration for avoiding the "connect-across" phenomenon. Thus, we enjoy significant decreases in surface point density required to avoid incorrect topology. We note however that our suggested point density necessary to prevent incorrect topology for our normally-distributed point distributions is simply based on experience and that we always perform a topology check to ensure a clean boundary. We sweep through all surface nodes in the 3D mesh and check that the incident tetrahedra on each surface node form a single face-connected component. If not, the problem is re-run with a higher surface density in the problem region.

2.4. 3-D mesh improvement operations

After the 3-D mesh has been generated by Algorithm 5, adjacent layers of volume points generated from the vertices of a surface triangles will tend to form prisms (see Fig. 11). Many adjacent prisms in the same layer will have inconsistent triangulations, resulting in a thin sliver element between them (see Fig. 12). These slivers can in many cases be flipped away using a $3 \rightarrow 2$ swap operation (see Fig. 13). It has been our experience that the following heuristic volume criterion eliminates slivers so that over 99% of the elements are not slivers and the swaps do not significantly disrupt the layer structure of the mesh. We perform a $3 \rightarrow 2$ swap operation if the minimum tetrahedral volume in the new set of (2) tetrahedra yielded by the operation is greater than 3 times the minimum tetrahedral volume in the old set of (3) tetrahedra eliminated by the operation. We use minimum tetrahedral volume instead of a worst aspect ratio criterion, because worst aspect ratio might improve if perfectly good anisotropic layers are destroyed and this is not our intent. We intend to rid ourselves of slivers within an anisotropic layer while preserving the layer. Nevertheless, tetrahedra with aspect ratio much higher than K (the anisotropy of the point distribution) will generally be flipped away when the volume flipping criterion is employed (*Note*: although the vast majority of flips are $3 \rightarrow 2$, we find that a very small number of potential $2 \rightarrow 3$ flips and $4 \rightarrow 4$ flips also satisfy the volume criterion and are flipped as well. Topologically, the $2 \rightarrow 3$ flip is the opposite of the $3 \rightarrow 2$ flip, while a diagram showing a $4 \rightarrow 4$ flip is given in [9, p. 225]).

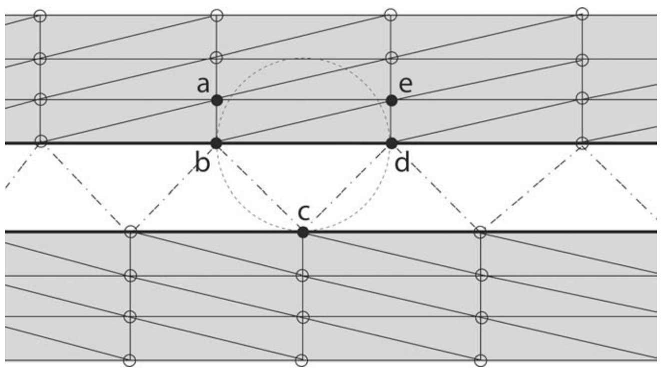


Fig. 10. *Connect-across phenomenon where separate pieces of the mesh region lie close together.* In the figure, the circumcircle of external triangle **b**, **c**, **d** formed with 'layer 0' points barely excludes the 'layer 1' points **a**, **e**. If the two meshed regions approach any closer, triangle **b**, **c**, **d** will not be part of a Delaunay mesh and undesirable connection between the meshed regions occurs. Similar phenomena occur in 3-D. This is prevented by enforcing (11).

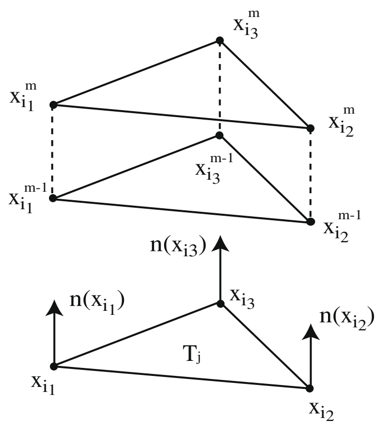


Fig. 11. Layer $m-1$ and layer m nodes generated from vertices of a surface triangle T_j form vertices of a prism.

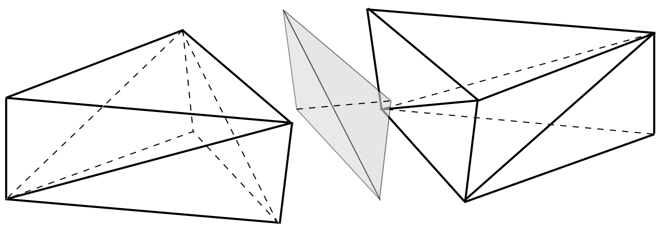


Fig. 12. Adjacent prisms with inconsistent (opposing diagonal) triangulations on their shared quad face force the Delaunay mesh to have a sliver.

Algorithm 5: Conforming Delaunay 3-D triangulation

```
[Take point set of Algorithm 4 and form layered Delaunay mesh that respects boundaries]
Create an initial mesh consisting of a single "big tetrahedron" containing all points in $\{\mathbf{x}_i^m\}$ in its interior.
[Process all points from one layer before moving onto next layer]
Do $m = 1, 2, \ldots, \frac{M}{2}, \ldots$
    For each node $i \in S$
        If $\mathbf{x}_i^m$ exists and has not been inserted into mesh Then
            Compute insertion cavity $\mathcal{C}(\mathbf{x}_i^m)$ consisting of the space occupied by all tetrahedra whose circumspheres contain $\mathbf{x}_i^m$.
            Insert $\mathbf{x}_i^m$ iff connecting $\mathbf{x}_i^m$ to triangular boundary facets of $\mathcal{C}(\mathbf{x}_i^m)$
                (i) does not create extremely small tetrahedra and
                (ii) does not connect a filler point $\mathbf{x}_i^m$ with $m > \frac{M}{2}$ to a previously inserted boundary node $\mathbf{x}_{i'}^0$
Repeat above Do Loop a couple of times to see if previously rejected $\mathbf{x}_i^m$ can be subsequently inserted.
[Delete tetrahedra exterior to $S$]
Delete all tetrahedra containing a point from the "big tet"
Delete all tetrahedra not containing an internal point $\mathbf{x}_i^m$ with $m \geqslant 1$
```

Finally, we have developed a 'tet crushing algorithm' that inserts nodes on the opposed diagonals of the slivers and then merges the nodes together, eliminating the slivers. The results from this heuristic algorithm as gauged by its performance on the examples in Section 3 are good: all tetrahedra with very small aspect ratio are eliminated. Since we are given latitude to make small finite deformations in the mesh, it is not surprising that results may exceed those of algorithms where connectivity changes are allowed but node displacements are not performed. An example of the latter is sliver exudation [7]. Our tet crushing algorithm will be presented elsewhere.

2.5. Overall tet mesh generation algorithm

The overall algorithm for producing tet meshes from an initial triangulated surface S is given in Algorithm 6. This algorithm will produce a mesh where the typical cross-section has M equally spaced layers and the tet anisotropy is generally between $\frac{K}{2}$ and K near the surface S.

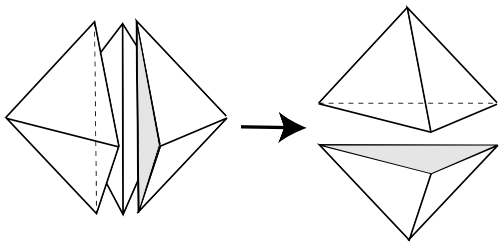


Fig. 13. Three to two flip: replacing a set of three tets with two new tets that occupy the same volume.

Clearly one can alter the parameter settings given in Algorithm 6, but these standard settings produce satisfactory meshes for a wide variety of input surfaces.

3. Examples

We give three examples of the use of the mesh generation algorithm.

Our first example geometry is the upper respiratory tract of a rat (genus 1). The geometry goes from nostrils (left hand side of Fig. 14(A)) to trachea (right hand side of Fig. 14(A)). The data is obtained from magnetic resonance imaging and the length scale is in voxel widths. The raw feature size exceeds 40 voxels at two isolated points where a few rays penetrate down rather than across an airway. The largest feature, the trachea, is correctly estimated to be in the 20–23 voxel range. Above the trachea are some folds that are in blue, meaning that they are the boundaries of smaller airways. These airways are in fact called the "turbinates" and they are intricate, convoluted airways used by the olfactory system. The smallest raw feature size is 0.2 voxels. In Panel B, we clip and gradient-limit the feature size field below at $L_{\min} = 0.5$ voxel widths (eliminating dubious subvoxel features) and above at $L_{\max} = 8$ voxel widths (eventually forcing more layers in the trachea). Panels C and D show close-up pictures of the same piece of surface mesh before and after surface refinement, respectively. Edge-bisection, node-merge, and edge-swap operations establish edge length between $\frac{K}{2M}d(\mathbf{x})$ and $\frac{K}{M}d(\mathbf{x})$, where here we have chosen meshing parameters $K = 4.2$ and $M = 7$.

Algorithm 6: Overall mesh generation algorithm

```
[Smooth initial marching cubes triangulation]
Perform Volume Conserving Smoothing (Algorithm 4 in [13])
  for 10 sweeps with underrelaxation coefficient $\omega = 1$.
[Improve surface mesh $S$]
Determine raw feature size $D(\mathbf{x})$ using (3) and (6)
Gradient-limit $D(\mathbf{x})$ to yield feature size $d(\mathbf{x})$ using Algorithm 1
[Change $S$ to have edge length between $\frac{K}{2M}d(\mathbf{x})$ and $\frac{K}{M}d(\mathbf{x})$]
For $i = 1,3$
    Do until nothing changes
        Merge nodes using Algorithm 3 with merge_length $= \frac{K}{2M}d(\mathbf{x})$ and toldamage $= \frac{K}{100M}d(\mathbf{x})$
        Swap surface edges (as in Fig. 9) with toldamage $= \frac{K}{100M}d(\mathbf{x})$
    Refine edges with Algorithm 2 with bisection_length $= \frac{K}{M}d(\mathbf{x})$
    Perform Volume Conserving Smoothing (Algorithm 4 in [13])
      for 10 sweeps with underrelaxation coefficient $\omega = 0.1$.
[Create Volume Mesh ]
Create volume points using Algorithm 4 with $\lambda_m = \frac{1}{M}$
Create volume tet mesh using Algorithm 5
Use 3 $\rightarrow$ 2 swaps and other mesh improvement operations (Section 2.4) to eliminate slivers
```

In Fig. 15 we see vertical cutaway sections through the resulting volume mesh and in Fig. 16 we see a horizontal cutaway section through the volume mesh. We note that we do indeed have 6 or 7 intact layers across airways as desired.

In Fig. 16 we see a converged computation on the rat nose mesh using the commercial STAR-CD computional fluid dynamics code [26]. Nasal inlets are extruded forward and trachea outlet is extruded backward to match the experimental setup where prescribed flow is applied at beginning of an inlet tube and zero pressure boundary conditions are prescribed at outlet of exit tube (Inlet and outlet tubes, respectively intersect extruded nostril and trachea sections). A description of our efforts at modeling airflow for purposes of particulate dosimetry estimation is given in [18].

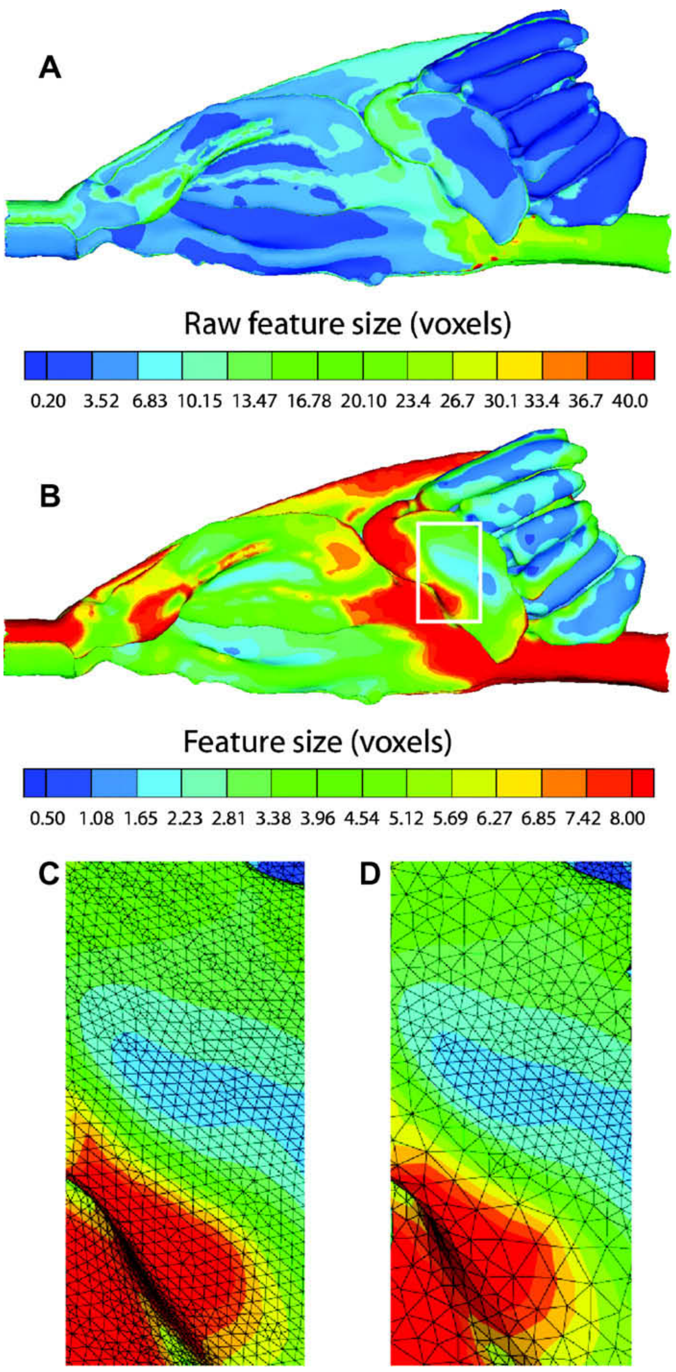


Fig. 14. Surface mesh of rat upper respiratory tract. (A) Raw feature size function depicted. (B) Clipped, gradient-limited feature size function. (C) Close-up of surface mesh at original density. (D) Close-up of surface mesh with revised density as determined by feature size function.

Our second example (Fig. 17) is a rat lung (genus 0). Meshing parameters were again set to $K = 4.2$ and $M = 7$. Magnetic imaging data was acquired using silicone casts of pulmonary airways in a 9-wk-old, male, Sprague-Dawley rat. Panel A shows the feature size field on the lung surface, as well as the locations of the cutaway sections B, C and D. Panel B shows the interior tetrahedra in a large airway, with $D(\mathbf{x})$ greater than $L_{\max}$, transitioning to a smaller airway with $L_{\min} < D(\mathbf{x}) < L_{\max}$. Panels C and D show sections through the trachea and a through a pair of airways downstream of the bifurcation, respectively. Panels E and F show both the external boundary and the tetrahedra in the neighborhood of several outlets. The outlet surfaces visible in panels E and F are not physical tissue surfaces but are cross-sectional surfaces at which

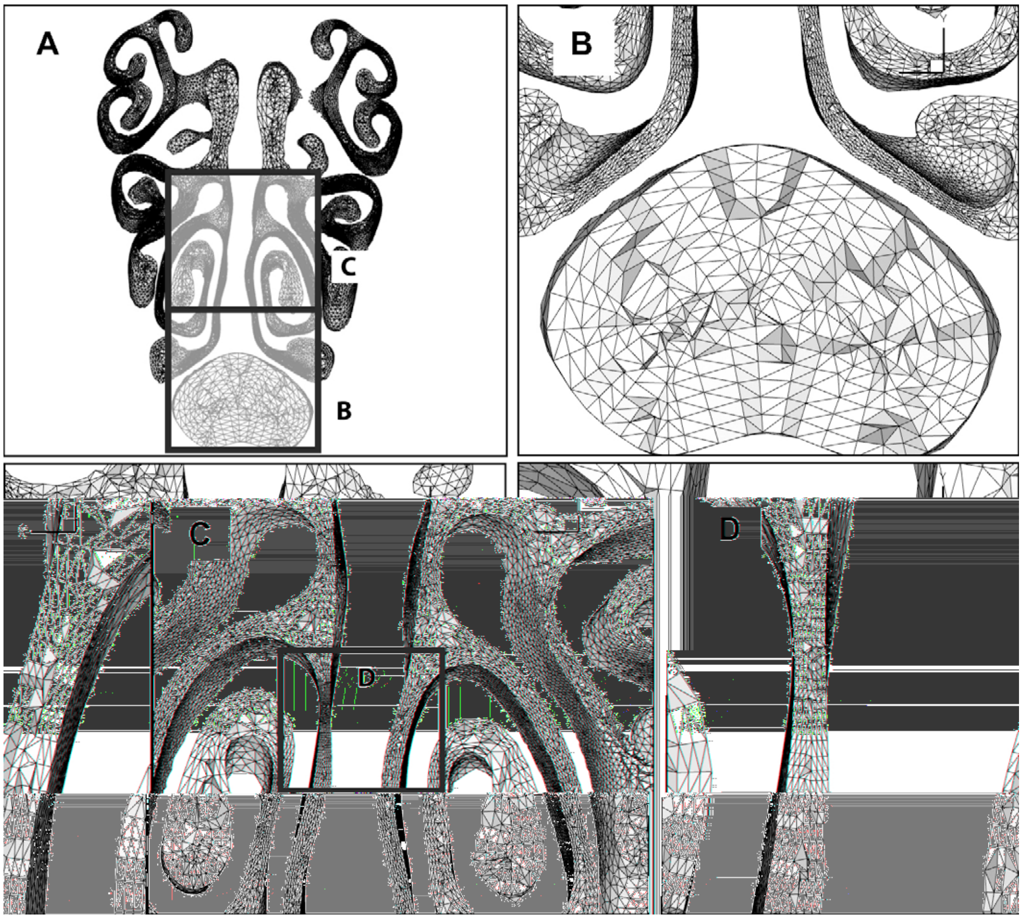


Fig. 15. Vertical slices from volume mesh of rat upper respiratory tract produced using Algorithm 6 with $K = 4.2$ and $M = 7$.

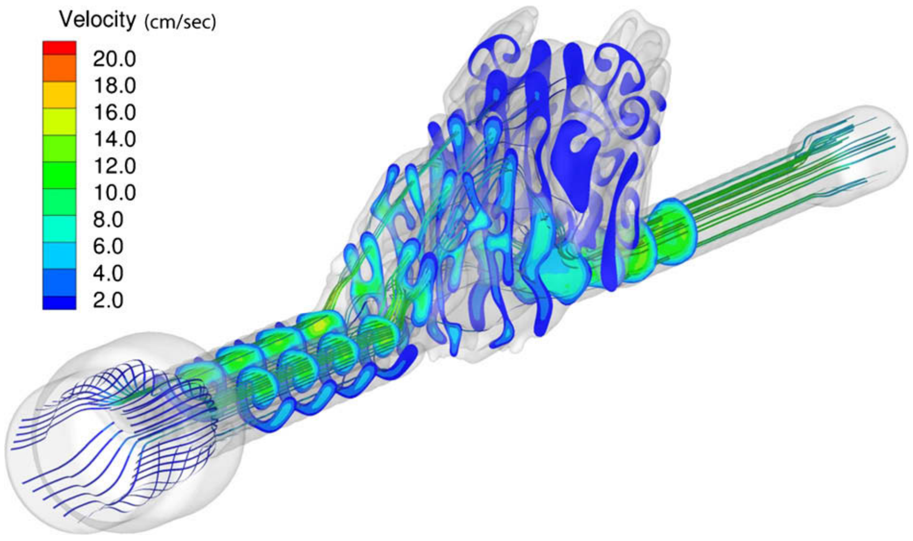


Fig. 16. Cutplanes along flow in CFD computation using volume mesh in rat upper respiratory tract. Nasal inlets are extruded forward and trachea outlet is extruded backward to match experimental setup.

outflow boundary conditions are applied. As such, M layers go *across* the outlet rather than *into* the outlet. This is accomplished by constraining the physical tissue normals adjacent to each outlet to lie in the plane of that outlet.

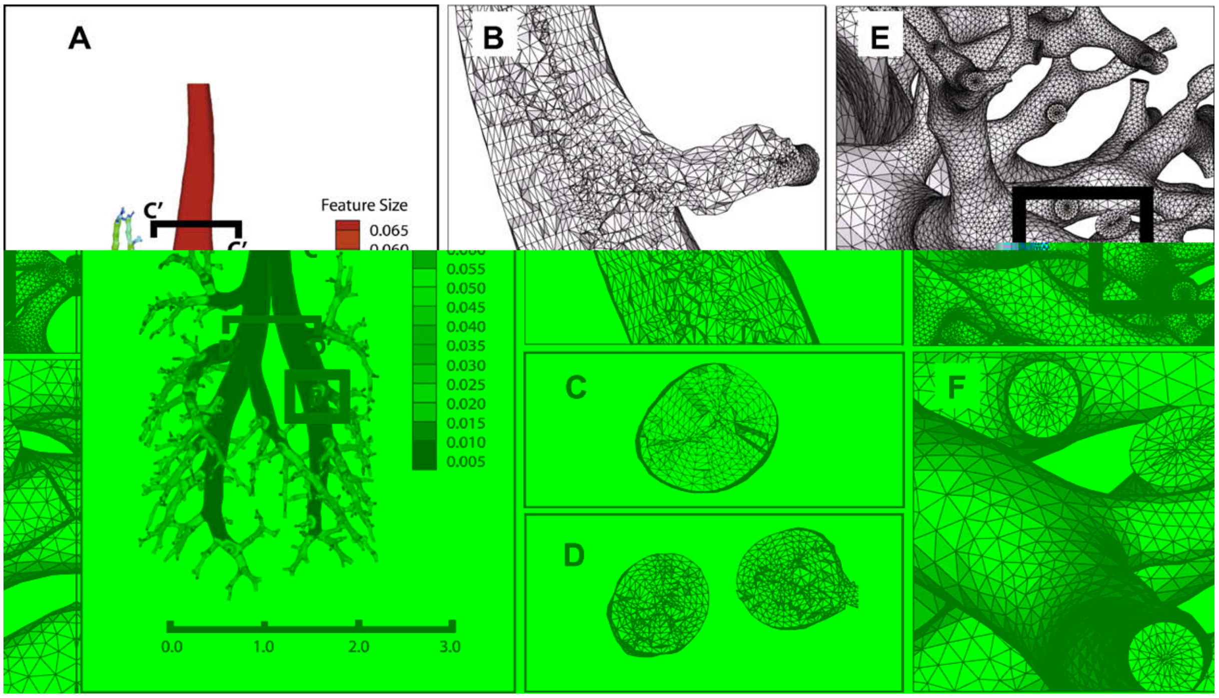


Fig. 17. Vertical slices from volume mesh of rat lung produced using Algorithm 6 with $K = 4.2$ and $M = 7$. Panel A shows the feature size field on the lung surface, as well as the locations of the cutaway sections B, C and D. Panel B shows the interior tetrahedra in a large airway, with $D(\mathbf{x})$ greater than $L_{\max}$, transitioning to a smaller airway with $L_{\min} < D(\mathbf{x}) < L_{\max}$. Panels C and D show sections through the trachea and a through a pair of airways downstream of the bifurcation, respectively. Panels E and F show both the external boundary and the tetrahedra in the neighborhood of several outlets.

Our third example (Fig. 18) is a human heart [30] of genus 12: Meshing parameters were again set to $K = 4.2$ and $M = 7$. Panel A shows the feature size field on the surface of the heart, as well as the locations of sections B and C. Panel B is a vertical cutaway through the heart, revealing the interior tetrahedra in the thick myocardium as well as the inter-atrial septum (detail in D). Panel C shows a diagonal cutaway through the heart, revealing myocardium, arterial lumen (top left) and the edge of the mitral valve. A magnified cross-section of the mitral valve is shown in Panel E. These cutaways demonstrate that we were able to achieve nearly orthogonal layering at both top and bottom scales.

Finally, we present mesh quality statistics in Fig. 19 for the meshes produced in these three examples. Here we present aspect ratio improvement, where for tetrahedra, we have defined:

$$\text{aspect ratio} \equiv 2\sqrt{6}\frac{\text{inscribed radius}}{\text{length of longest edge}},$$

which has maximum value 1 for a regular tetrahedron. The cumulative aspect ratio improvement is critical in allowing us to perform numerical computations on these challenging biomedical geometries.

4. Limitations and areas for future improvement

This algorithm deposits nodes along rays to the raw local half-diameter. For smooth geometries, this produces coverage of the interior. However when there is a jump in the surface normal due to "man-made" non-biological features, such as a machined edge, there may be incomplete coverage. In this case, at surface points where the neighborhood has a jump in the surface normal, we must cast multiple rays of interior points at various angles to the surface. For the smooth geometries shown, this is not necessary.

Highly curved concave boundaries, such as encountered when meshing the *complement* of a tubular region are likely to result in incomplete coverage of the interior region as in the case just mentioned. In this case the most straightforward approach is to let the feature size also contain a term for curvature:

$$D(\mathbf{x}) = \min(D_{\text{ray-shoot}}, 2 * R_{\text{curvature}}), \tag{12}$$

where $R_{\text{curvature}}$ is the principal radius of curvature of smallest magnitude. This result is then gradient-limited as before. A numerically stable algorithm for computing principal curvatures is given in [24]. Our experience with airway geometries has been that this is not required, but we have found that meshing of the tissue space around the coronary arteries of the heart necessitates this term.

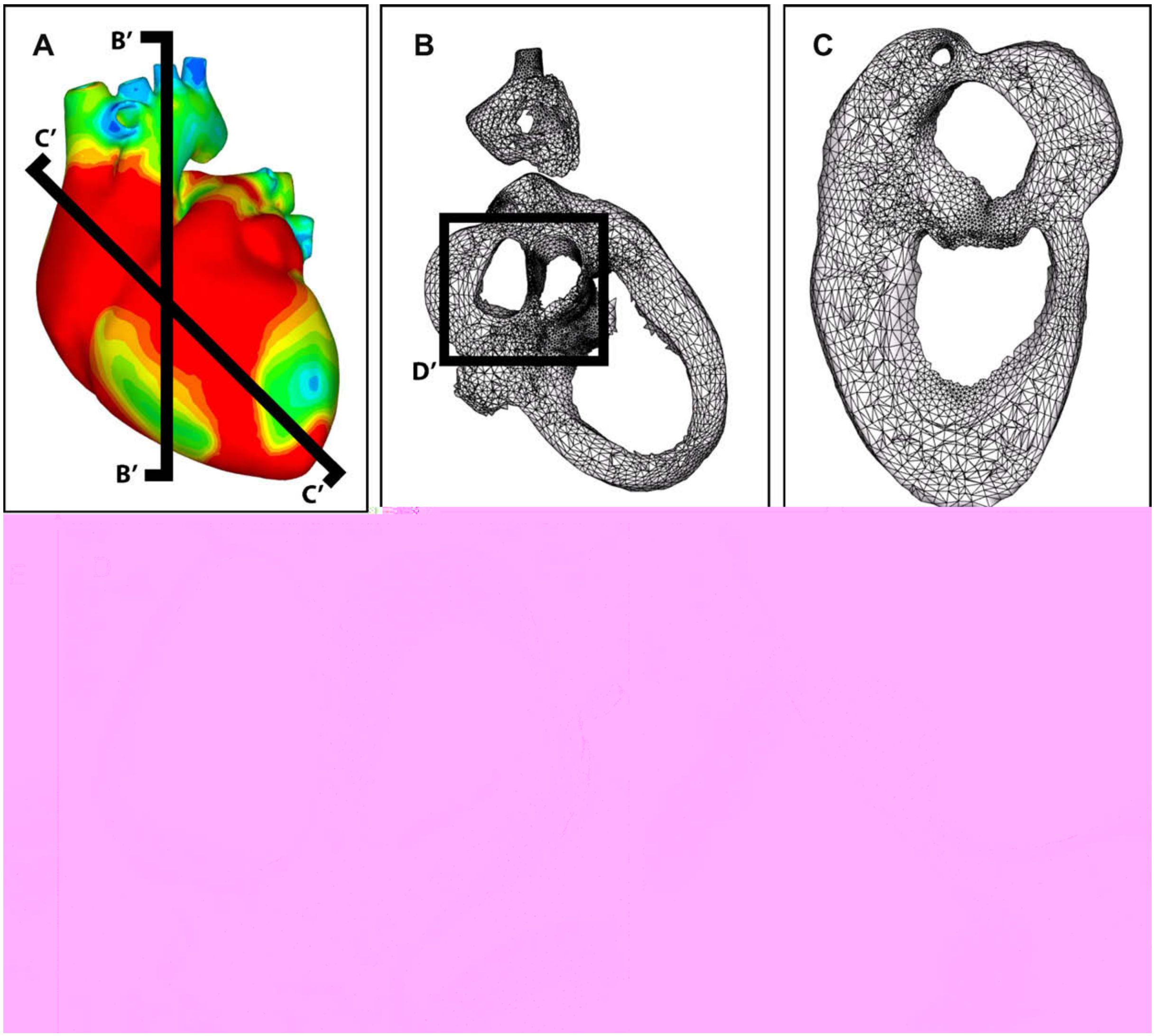
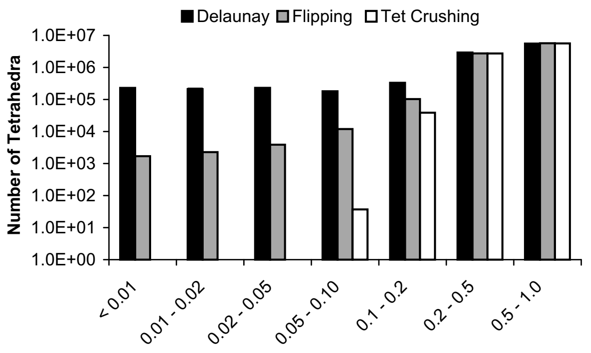


Fig. 18. Vertical slices from volume mesh of human heart produced using Algorithm 6 with $K = 4.2$ and $M = 7$. Panel A shows the feature size field on the surface of the heart, as well as the locations of sections B and C. Panel B is a vertical cutaway through the heart, revealing the interior tetrahedra in the thick myocardium as well as the inter-atrial septum (detail in D). Panel C shows a diagonal cutaway through the heart, revealing myocardium, arterial lumen (top left) and the edge of the mitral valve. A magnified cross-section of the mitral valve is shown in Panel E.

The scheme of depositing points along rays to half the local diameter can also cause significant overlap with some kinds of geometries. Clearly, for a perfect sphere or torus there is no overlap, but for e.g. a (rounded) brick, there is significant overlap. The biological geometries we deal with here are tubular or crevasse-like and there is usually little overlap. Extra points in the interior region are usually acceptable. It is advisable to filter the set of interior points before Delaunay connection so that if two interior points (generated from two different surface points) are by chance a small fraction of a voxel apart, the one which is further from its surface generator point is deleted. (This filtration is rapidly done with a AABB tree.)

Our algorithm makes an automatic choice of when small-scale features should be ignored. In Fig. 20 in the left-side illustration, the surface undulations are so gentle they do not figure into the feature size computed by the algorithm. Indeed rays cast from points A and B miss neighboring features. The result is that the medial axis (dotted gray lines) is effectively ignored and the feature size reflects only the larger scale size of the geometry. On the right-side illustration, the amplitude of the undulations is large enough to be detected by the algorithm. Here, a ray cast normally from the inflection point between a peak and a trough (point B') intersects a neighboring feature, causing the feature size to be lessened to the scale of the undulations (The feature size at the point A' is also reduced as indicated, due to gradient-limiting of the feature size field). Hence, the medial axis (dotted lines) is effectively recognized. We have found this automatic behavior to be desirable for meshing our biological geometries. However, the user may demand some control over what is or is not an "ignorable" undu-

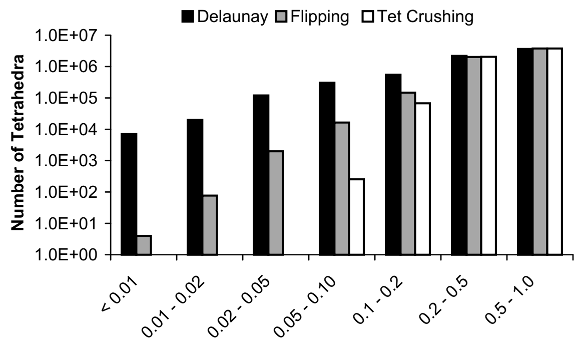
Binned Aspect Ratio



Rat URT

Aspect Ratio

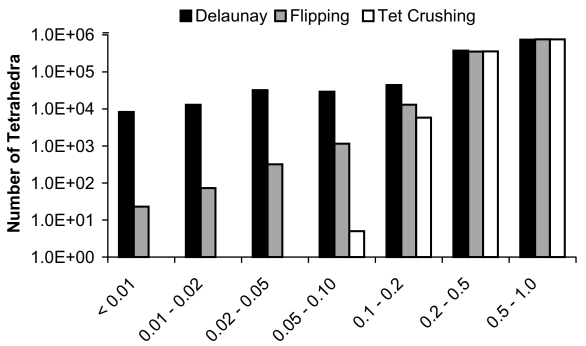
	Delaunay	Flipping	Tet Crushing
Minimum	2.92E-06	3.62E-04	5.89E-02
< 0.01	229620	1698	0
0.01 - 0.02	173384	2127	0
0.02 - 0.05	230592	3877	0
0.05 - 0.10	180810	11930	37
0.10 - 0.20	331993	103012	38661
0.20 - 0.50	2923158	2741282	2738166
0.50 - 1.00	5571754	5697029	5641834



Rat lung

Aspect Ratio

	Delaunay	Flipping	Tet Crushing
Minimum	2.52E-04	7.05E-03	5.99E-02
< 0.01	7013	4	0
0.01 - 0.02	19960	77	0
0.02 - 0.05	119913	1983	0
0.05 - 0.10	306458	16398	254
0.10 - 0.20	551290	147502	67417
0.20 - 0.50	2189099	2017952	2045995
0.50 - 1.00	3646786	3775105	3772451



Human Heart

Aspect Ratio

	Delaunay	Flipping	Tet Crushing
Minimum	9.62E-05	1.81E-03	8.32E-02
< 0.01	8309	23	0
0.01 - 0.02	12955	73	0
0.02 - 0.05	32181	320	0
0.05 - 0.10	29245	1154	5
0.10 - 0.20	44404	13013	5837
0.20 - 0.50	376128	354326	358242
0.50 - 1.00	744118	758154	755690

Fig. 19. Mesh quality improvement by flipping and then tet crushing for one human and two rat geometries.

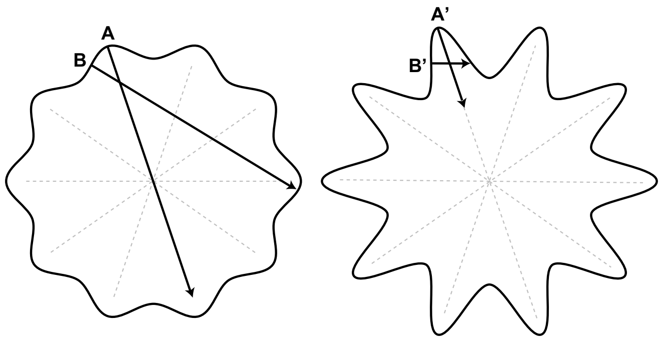


Fig. 20. Perturbation from circular shape in left-hand figure is too small to affect feature size. Larger perturbations in right-hand figure are not ignored. Smaller feature size detected at B' forces smaller feature size at A' due to gradient-limiting.

lation on the surface. One solution that would cause the algorithm to recognize smaller undulations would be adding a curvature correction to the feature size computation as in (12).

5. Conclusions

We have implemented in the PNNL version of the LaGriT grid management toolbox [14] an algorithm for the generation of anisotropic meshes that are scale-invariant over a prescribed range. The algorithm relies on a concept of feature size that is obtained by considering the lengths of rays that are shot normally from the surface into the volume and reintersect the surface. This approach is especially suited to rapid, quality meshing of biological structures where CFD calculations require a definite number of intact mesh layers in regions of great geometric complexity.

Acknowledgement

Research funded by the National Heart and Blood Institute Award 1RO1HL073598-01A1. The authors take this opportunity to acknowledge the contributions of Drs. Kevin Minard and Rick Jacob (MRI data acquisition), Senthil Kabilan and Dr. James Carson (MRI data segmentation and editing), and Dr. Rick Corley (program support). We also gratefully acknowledge the helpful critical comments of the reviewers which significantly improved the manuscript.

Appendix A. Damage estimate for node merging

If we merge node i into merge j in the neighborhood Ω_i of $\mathbf{x}_i$, we create a new patch Ω_{new} of piecewise linear triangles as in Fig. A.1. Here, the triangles T_k in Ω_i that contain the edge $\overline{\mathbf{x}_i\mathbf{x}_j}$ are eliminated (since the edge is contracted to a point) and all other triangles T_k yield a new triangle T_k^{new} by replacement of node i with node j and keeping the other two nodes the same.

Thus the action of merging node i to node j deforms the neighborhood Ω_i into Ω_{new}; the thickness of the volume between these two surface patches is thus a good measure of the damage of the merge operation. Note that in fact the volume between Ω_i and Ω_{new} is tetrahedralized by the set of tetrahedra $\{\mathcal{T}_k\}$ formed by adjoining the point $\mathbf{x}_i$ to each of the triangles in Ω_{new}. The altitude of $\mathcal{T}_k$ above its base T_k^{new} is $|(\mathbf{x}_i - \mathbf{x}_j) \cdot \hat{\mathbf{n}}_k^{\text{new}}|$, where $\hat{\mathbf{n}}_k^{\text{new}}$ is the unit vector normal to T_k^{new}. So the volume swept out by merging i to j is formed by the union of tetrahedra sitting atop Ω_{new}, none of which has altitude exceeding

$$\text{height bound} \equiv \max_k |(\mathbf{x}_i - \mathbf{x}_j) \cdot \hat{\mathbf{n}}_k^{\text{new}}|. \tag{A.1}$$

We thus naturally define $\mathbf{damage}(\mathbf{x}_i \to \mathbf{x}_j)$ to be this height bound.

Note that this damage estimate discriminates among possible merge neighbors in neighborhoods with creases, such as depicted in Fig. A.2. Here the merge $i \to j$ has a damage estimate of zero, while, e.g. the merge $i \to k$ has a positive damage estimate. For a sufficiently small damage tolerance, the merge $i \to k$ will not be performed.

We also perform an orientation check to make sure that merge $i \to j$ does not introduce a fold into the mesh. We define a "patch normal" $\mathbf{x}_{\Omega_i}$, similar to (6) but instead each triangle is weighted by triangle area:

$$\hat{\mathbf{n}}_{\Omega_i} \equiv \frac{\sum_k A_k \hat{\mathbf{n}}_k}{\|\sum_k A_k \hat{\mathbf{n}}_k\|}, \tag{A.2}$$

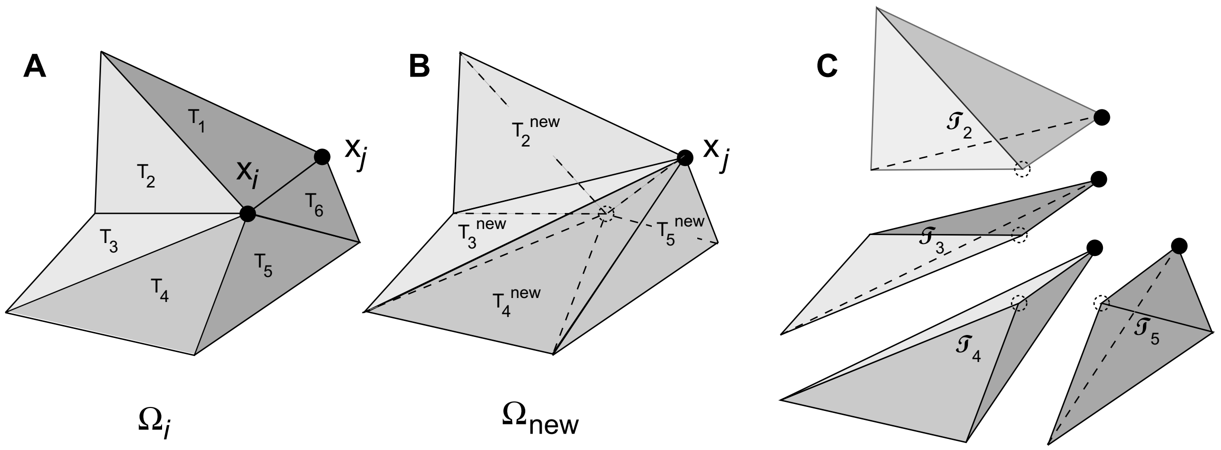


Fig. A.1. Merging of node i at $\mathbf{x}_i$ to neighboring node j at $\mathbf{x}_j$. (A) Appearance before merge where neighborhood Ω_i of $\mathbf{x}_i$ consists of N surface triangles T_k. (B) Appearance after merge where replacement neighborhood Ω_{new} consists of $N-2$ new surface triangles T_k^{new}. (C) Deformation of surface by this merge sweeps out a volume that can be decomposed into $N-2$ tetrahedra $\mathcal{T}_k$.

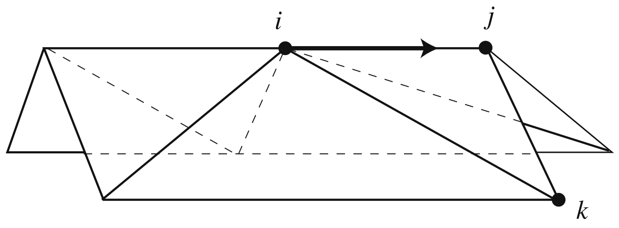


Fig. A.2. Merging along crease ($\mathbf{x}_i \to \mathbf{x}_j$) incurs far less damage than merging down the crease ($\mathbf{x}_i \to \mathbf{x}_k$).

where the sum is over the triangles T_k incident on node i and then A_k and $\hat{\mathbf{n}}_k$ are the areas and normals of the triangles. This normal is more representative than $\hat{\mathbf{n}}_{\text{synth}}$ of the average normal direction of the whole patch Ω_i and in fact it is easy to prove that it is independent of the position $\mathbf{x}_i$ and is only determined by the geometry of the line segments comprising the patch boundary $\partial\Omega_i$.

To check that the merge $i \to j$ does not introduce a fold into the mesh, we check that if $\hat{\mathbf{n}}_k^{\text{new}}$ are the normals for the new triangles in Ω_{new}, that

$$\hat{\mathbf{n}}_{\Omega_i} \cdot \hat{\mathbf{n}}_k^{\text{new}} > 0 \quad \forall k. \tag{A.3}$$

If this condition is not obeyed, we surmise that merge $i \to j$ will introduce a fold into surface and we do not perform the operation.

Appendix B. Damage estimate for surface edge swapping

The swap of the mutual diagonal of adjacent triangles (Fig. 9) also introduces damage. However, this operation can be viewed as the merging of a hypothetical node $\mathbf{x}_i$ on the midpoint of the initial diagonal to one of the vertices $\mathbf{x}_j$ not on the diagonal. Accordingly, the damage estimate **damage**($\mathbf{x}_i \to \mathbf{x}_j$) from the previous section is employed. Also, to prevent introduction of creases into the surface mesh, we use the orientation check (A.3) from the previous section.

References

[1] N. Amenta, M. Bern, Surface reconstruction by voronoi filtering, Discr. Comput. Geom. 22 (1999) 481–504.

[2] N. Amenta, M. Bern, M. Kamvysselis, A new voronoi-based surface reconstruction algorithm, Siggraph (1998) 415–421.

[3] T.J. Baker, Automatic mesh generation for complex three-dimensional regions using a constrained delaunay triangulation, Engineering with Computers, Springer-Verlag, 1989. Num 5, pp. 161–175.

[4] J.A. Bærentzen, H. Aanæs, Signed distance computation using the angle weighted pseudonormal, IEEE Trans. Vis. Comput. Graphics 11 (3) (2005) 243–253.

[5] Introduction to Algorithms, 2nd ed., Cormen, Leiserson, Rivest, and Stein, MIT Press, 2007.

[6] H. Edelsbrunner, Geometry and Topology for Mesh Generation, Cambridge University Press, 2001.

[7] H. Edelsbrunner, D. Guoy, An experimental study of sliver exudation, in: Proc. 10th Int. Meshing Roundtable, 2001, pp. 307– 316.

[8] R. Garimella, M.S. Shepard, Tetrahedral mesh generation with multiple elements through the thickness, Eng. Comput. 15 (2) (1999) 181–197.

[9] Delaunay Triangulation and Meshing, George and Borouchaki, Hermes, Paris, 1998.

[10] Y. Ito, A.M. Shih, B.K. Soni, Reliable isotropic tetrahedral mesh generation based on an advancing front method, in: Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA, Sandia National Laboratories, SAND #2004-3765C, September 19–22, 2004, pp. 95–106.

[11] K.E. Jansen, M.S. Shephard, M.W. Beall, On Anisotropic mesh generation and quality control in complex flow problems, in: Tenth International Meshing Roundtable, 2001.

[12] A. Khamayseh, G. Hansen, Use of the spatial kD-tree in computational physics applications, Commun. Comput. Phys. 2 (2007) 545–576.

[13] A. Kuprat, A. Khamayseh, D. George, L. Larkey, Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines, J. Comput. Phys. 172 (2001) 99–118. http://dx.doi.org/10.1006/jcph.2001.681.

[14] LaGriT – Los Alamos Grid Toolbox. <http://lagrit.lanl.gov>. For PNNL version, contact corresponding author.

[15] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Comput. Graph. 21 (4) (1987) 163–169.

[16] D.L. Marcum, Efficient generation of high-quality unstructured surface and volume grids, Eng. Comput. 17 (2001) 211–233.

[17] D.L. Marcum, N.P. Weatherill, Unstructured grid generation using iterative point insertion and local reconnection, AIAA J. 33 (9) (1995) 1619–1625. 0001–1452.

[18] K.R. Minard, D.R. Einstein, R.E. Jacob, S. Kabilan, A.P. Kuprat, C.A. Timchalk, L.L. Trease, R.A. Corley, Application of magnetic resonance (MR) imaging for the development and validation of computational fluid dynamic (CFD) models of the rat respiratory system, Inhal. Toxicol. 18 (2006) 787–794.

[19] T.S. Newman, H. Yi, A survey of the marching cubes algorithm, Comput. Graph. 30 (5) (2006) 854–879.

[20] S.J. Owen, S. Saigal, Surface mesh sizing control, Int. J. Numer. Meth. Eng. 47 (2000) 497–511.

[21] P.O. Persson, Mesh size functions for implicit geometries and PDE-based gradient limiting, Eng. Comput. 22 (2006) 95.

[22] J.F. Remacle, X. Li, M.S. Shephard, J.E. Flaherty, Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods, Int. J. Numer. Methods Eng. 62 (2005) 899–923.

[23] M. Rivara, New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations, Int. J. Num. Meth. Eng. 40 (1997) 3313–3324.

[24] S. Rusinkiewicz, Estimating curvatures and their derivatives on triangle meshes, in: Symposium on 3D Data Processing, Visualization, and Transmission, 2004.

[25] B. Smits, Efficiency issues for ray tracing, ACM SIGGRAPH 2005 Courses, Session: Introduction to real-time ray tracing, Article No. 6, <http://doi.acm.org/10.1145/1198555.1198745>, 2005.

[26] STAR-CD, computational fluid dynamics (commercial) code, <http://www.cd-adapco.com/products/STAR-CD/index.html>.
[27] G. Thürmer, C. Wüthrich, Computing vertex normals from polygonal facets, J. Graph. Tools 3 (1) (1998) 43–46.
[28] D.F. Watson, Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes, Comput. J. 24 (2) (1981) 167–172.
[29] S. Yamakawa, C. Shaw, K. Shimada, Layered tetrahedral meshing of thin-walled solids for plastic injection molding FEM, in: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, Cambridge, Massachusetts, 2005, pp. 245–255.
[30] Y. Zhang, C. Bajaj, B.S. Sohn, 3D finite element meshing from imaging data, Comput. Methods Appl. Mech. Eng. 194 (48–49) (2004) 5083–5106.
[31] J. Zhu, T. Blacker, R. Smith, Background overlay grid size functions, in: Proceedings of the 11th International Meshing Roundtable, Sandia National Laboratories, September 15–18, 2002, pp. 65–74.