Tetrahedral Image-To-Mesh Conversion For Biomedical Applications*

Andrey N. Chernikov
Department of Computer Science
Old Dominion University
Norfolk, VA
achernik@cs.odu.edu

Nikos P. Chrisochoides
Department of Computer Science
Old Dominion University
Norfolk, VA
nikos@cs.odu.edu

ABSTRACT

The modeling of physical processes in biomedical image analysis requires a discretization of the image space into simple shapes like triangles in two dimensions and tetrahedra in three dimensions. These discretizations are known as meshes, and the construction of the meshes as image-to-mesh conversion. There are a number of requirements on image-to-mesh conversion, the most critical of them being the shape of mesh elements in terms of the absence of small angles, the faithful geometrical representation of the tissues by the mesh elements, small number of elements for real-time Finite Element and Finite Volume analysis, and rapid execution times. We present a novel algorithm for triangular and tetrahedral image-to-mesh conversion which allows for guaranteed bounds on the smallest dihedral angle and on the distance between the boundaries of the mesh and the boundaries of the tissues. The algorithm produces a small number of mesh elements that comply with these bounds. We also describe and evaluate our implementation of the proposed algorithm on two publicly available three-dimensional medical atlases. The implementation is faster than a state-of-the art Delaunay code, and in addition solves the small dihedral angle problem.

Categories and Subject Descriptors

I.3.5 [**Computing Methodologies**]: Computer Graphics—*Computational Geometry and Object Modeling*

General Terms

Algorithms, Performance, Design, Theory

Keywords

Image-to-mesh conversion

*Provisional patent pending.

ACM-BCB '11, August 1-3, Chicago, IL, USA

1. INTRODUCTION

The ability to tessellate a medical image of a tissue into tetrahedra will enable quantitative analysis on patient-specific images using Finite Element (FE) and Finite Volume (FV) methods. This has significant implications in a vast range of areas, such as imaged-guided therapy, development of advanced patient-specific blood flow mathematical models for the prevention and treatment of stroke, patient-specific interactive surgery simulation for training young clinicians, and study of biomechanical properties of collagen nano-straws of patients with chest wall deformities, to name just a few. Below we briefly describe the above mentioned applications which are the most familiar to us due to the ongoing collaborative research efforts.

Non-Rigid Registration for Image Guided Neurosurgery. The Brain Tumor Society estimates that each year more than 200,000 people in the USA are diagnosed with a primary or metastatic brain tumor. A greater percentage of resection and a smaller volume of postoperative residual tumor are associated with an improved prognosis for the patient [7, 16]. The majority of malignant gliomas recur within 2 cm of the enhancing edge of the original tumor, providing impetus for improved surgical resection. Precise delineation of resection margins is very difficult because tumors can closely resemble brain. Maximal resection is also complicated because tumors infiltrate and can be immediately adjacent to critical functioning brain tissue. Moreover, deformation of brain structures occurring intra-operatively renders pre-operative images inaccurate. Non-rigid registration techniques are capable of utilizing transformations that model local deformations. The registration problem is formulated as an iterative estimation of the deformation using an approximation method. A tetrahedral finite element mesh has a dual role in the formulation. First, it is used to find the mechanical energy of the system and model deformation of the brain as a physical body based on FEM. In addition, the mesh is also used to regularize, or smooth, the displacements estimated with block matching [6, 11].

Blood Flow Simulation. Cerebrovascular disease (CVD), or stroke, is one of the leading natural causes of death in the US, accounting for 1 in 17 deaths in 2005, and associated with debilitating morbidity among survivors. Interactions of blood flow in the human brain occur between different scales, determined by flow features in the large arteries, the smaller arteries and arterioles, and the capillaries all being coupled to cellular and sub-cellular biological processes. While many biological aspects have been studied systematically, surprisingly little effort has been put into studying

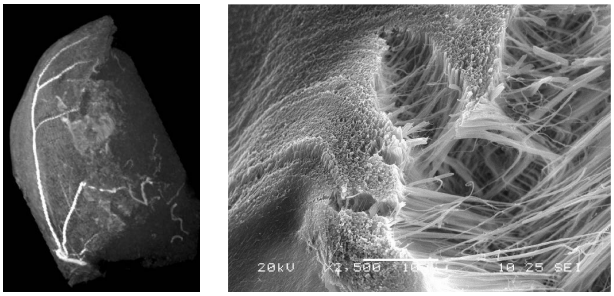

Figure 1: Left: MRA, courtesy of Dr. Wu of Neurosurgical Department of Huashan Hospital Shanghai Medical College, Fudan University, Shanghai, China. Right: Image Analysis of Patient Cartilage, courtesy of Dr. Stacey of Old Dominion University.

blood flow patterns within the brain and no studies exist that couple brain-shift and multi-scale blood flow in the context of neurosurgery or simulators for training young neurosurgeons. Recent 3D imaging of the human brain provides statistical information for constructing realistic topological finite element models on which future brain simulations will be based. Moreover, imaging technology and real-time image-guided neurosurgery systems in advanced operating suites can be used for better understanding for treatment and prevention of CVD. The missing piece is the capability to perform finite element simulations at a level-of-detail that can provide insight on the what, why, and how in addressing stroke-related problems to improve the long-term prospects of patients for good quality of life while reducing health care costs. Again, Image-to-Mesh (I2M) conversion is a critical building block in these simulations.

Medical Simulators. The purpose of interactive surgery simulation is to train young surgeons in conjunction with a haptic and visual computer interface, and it has been shown in some studies to have a measurable impact on surgical skill and patient outcome. Finite elements are a method for numerically estimating unknown displacements, stresses, forces and possibly other variables, by expressing an equation for mechanical equilibrium conditions. This approach requires that a tissue volume be decomposed into simple geometrical components such as tetrahedra or hexahedra. In this paper we focus on tetrahedral meshes only. There are ongoing efforts [25] that focus on hexahedral elements.

Image Analysis Of Patient Cartilage For Predicting Surgical Responses. Collagen fibers (nano-straws) are responsible for tensile strength of cartilage, see Figure 1. The cartilage of patients with chest wall deformities has been described as weak; therefore, abnormalities of collagen fibers observed in the cartilage of these patients may underlie these disorders. Until recently, it has not been possible to accurately and reproducibly measure biomechanical properties of collagen fibers. A state-of-the-art instrument, four-probe MultiView4000TM AFM (Atomic Force Microscopy), and physics-based nonrigid registration and I2M technologies for 2D and 3D images from our group have made such an undertaking possible.

1.1 Requirements on Image-To-Mesh Conversion

The problem of unstructured I2M conversion is the following. Given an image as a collection of voxels, such that each voxel is assigned a label of a single tissue or of the background, construct a tetrahedral mesh that overlays the tissues and conforms to their boundaries. In this paper we present an algorithm for constructing meshes that are suitable for real-time finite element analysis, i.e., they satisfy the following requirements:

1. Elements do not have arbitrarily small angles which lead to poor conditioning of the stiffness matrix in FE and FV Analysis for biomechanics applications. In particular, we guarantee that all dihedral angles are above a user-specified lower bound which can be set to any value up to 35.26°. In contrast, guaranteed quality Delaunay methods only satisfy a bound on circumradius-to-shortest edge ratio which in 3D does not imply a bound on dihedral angles.
2. The mesh offers a reasonably close representation (fidelity) of the underlying tissues. Since the image is already an approximation (up to a pixel granularity) of a continuous physical object, even a strict matching of the mesh to individual pixel's boundaries will not lead to a mesh which is completely faithful to the boundaries of the object. Moreover, this approach will produce a large number of elements that will slow down the solver. Instead, our solution is to expose parameters that allow for a trade-off between the fidelity and the final number of elements with the goal of improving the end-to-end execution time of the FE analysis codes.
3. The number of tetrahedra in the mesh is as small as possible provided the two requirements above are satisfied. We achieve this goal by developing a specialized mesh decimation procedure.
4. The mesh can be constructed in a reasonably short period of time, typically within a few minutes.

Below we describe our efficient implementation that meets all of these requirements. Our proposed method is designed to be suitable for parallelization which will further improve its execution time and allow for greater problem sizes.

1.2 Approach and Related Work

There is a large body of work on constructing guaranteed quality meshes for Computer Aided Design (CAD) models. The specificity of CAD-oriented approaches is that the meshes have to match exactly to the boundaries of the models. The most widely used guaranteed-quality CAD-oriented approach is based on Delaunay refinement [13]. However, the problem with Delaunay refinement in 3D is that it allows only for a bound on circumradius-to-shortest edge ratio of tetrahedra, which does not help to improve the dihedral angles. As a result, almost flat tetrahedra called slivers can survive. There are a number of post-processing techniques to eliminate slivers [1–3, 19, 23, 32]. While some of them have been shown to produce very good dihedral angles in practice, we are not aware of an implementation that can *guarantee* significant (1° and above) dihedral angle bounds.

Labelle and Shewchuk [18] described an Isosurface Stuffing method for guaranteed quality tetrahedral meshing for domains defined by general surfaces. They offer a one-sided fidelity guarantee (from the mesh to the model) in terms of the Hausdorff distance, and, provided the surface is sufficiently smooth, also the guarantee in the other direction (from the model to the mesh). Their algorithm first constructs an octree that covers the model, then fills the octree leaves with high quality template elements, and finally warps the mesh

vertices onto the model surface, or inserts vertices on the surface, and locally modifies the mesh. Using interval arithmetic, they prove that new elements have dihedral angles above a certain threshold. However, images are not smooth surfaces, and to the best of our knowledge, this technique has not been extended to mesh images. One approach could be to interpolate or approximate the boundary pixels by a smooth surface, but it would be complicated by the need to control the maximum approximation (interpolation) error. On the other hand, an I2M solution can benefit from the fact that images provide more information on their structure than general surfaces. For example, in our proposed I2M algorithm we do not have to struggle with the problem of quadruple-zero tetrahedra, which complicates the Isosurface Stuffing method. Quadruple-zero tetrahedra are those that have all four vertices on the surface, and it is not clear if they should be classified as interior or exterior.

There are also heuristic solutions to the I2M problem, some of them developed in our group [10, 20], that fall into two categories: (1) first coarsen the boundary of the image, and then apply CAD-based algorithms to construct the final mesh, (2) construct the mesh which covers the image, and then warp some of the mesh vertices onto the image surface. The first approach tries to address the fidelity and then the quality requirements, while the second approach does it in reverse order. Unfortunately, neither of these approaches can guarantee the quality of elements in terms of dihedral angles. Both of them face the same underlying difficulty which consists in separating the steps that attempt to satisfy the quality and the fidelity requirements. As a result, the output of one step does not produce an optimal input for the other step. An approach based on filling in brick elements with quality tetrahedra was developed by Hartmann and Kruggel [15], however, it keeps an over-refined mesh near the boundaries. Another method by Dogan et al. [9] produces a mesh as a by-product of an iterative segmentation procedure, by an application of a CAD-oriented mesh generator Triangle [27] to the segmented boundaries.

The solution we propose in this paper is to simultaneously satisfy the quality and the fidelity requirements. We achieve this goal by constructing an initial fine mesh with very high quality and fidelity. The construction of this mesh is feasible due to the specific structure of the input, which is a collection of cubic blocks corresponding to the voxels of the image. This initial mesh, however, has a large number of elements due to the fact that it is a one-fits-all solution with respect to the angle and fidelity parameters, for a given image, since it satisfies the highest dihedral angle and fidelity bounds. Therefore, we implement a post-processing decimation step that coarsens the mesh to a much lower number of elements while at all times maintaining the required fidelity and quality bounds.

Mesh coarsening using vertex removal operation, which we use in our algorithm, has been employed previously in various formulations in a large number of published results (some of which mentioned below) for a variety of optimization problems. However, we are not aware of these or other publications to propose a mesh simplification algorithm that simultaneously bounds the quality and the fidelity of the mesh in the sequence of vertex removal (also known as edge collapse) operations. For example, Cohen et al. [8] proposed *simplification envelopes* for producing hierarchies of level-of-detail approximations of polygonal models. They guarantee a bound on the distance between the object and its approximation, however do not account for the dihedral angles. Garland and Heckbert [12] presented a surface simplification algorithm which can preserve surface shape and associated properties like color and texture, but also does not consider the dihedral angles in the volume. Trotts and Joy [33] bound the deviation error in the linear spline approximations of the scalar fields. Cignoni et al. [5] measure only the distance between the original and the simplified surfaces. Chopra and Meyer [4] describe a tetrahedral mesh simplification framework where the error metric uses a *normal stretch ratio* which accounts for the length change in the base normal of an affected tetrahedron. Hoppe et al. [17] represent a mesh as a system of springs, and accept an edge collapse only if it reduces the total energy. Yan et al. [34] prioritize edge collapse operations based on the importance of the corresponding vertices in the hierarchical structure. Schroeder et al. [26] use the decimation criterion based on the distance of the vertex to the average plane in its neighborhood. Pajarola and Rossignac [22] prioritize edges in order of increased surface approximation error for progressive mesh compression.

Zhang et al. [35] described an algorithm to construct adaptive and quality 3D meshes from imaging data. Similar to our approach, they create an initial octree-based mesh, and then improve its quality using iterative edge contraction. Specifically, their approach removes tetrahedra with the worst ratio of the longest to shortest edge length by contracting their shortest edges; however, when it is detected that a requested ratio threshold cannot be reached the strategy is reversed to point insertion through longest edge bisection. The conceptual difference with the algorithm proposed in this paper is that the approach by Zhang et al. uses mesh decimation to improve both element shape and mesh size simultaneously which is not always guaranteed to reach a desired threshold, while we trade element shape for mesh size and thus can stop shape deterioration at any bound below the high starting value. Another approach proposed by Reid et al. [24] and Goksel et al. [14] is to iteratively deform an initial mesh by vertex movement and other operations to conform to the boundaries in the image.

Our proposed approach may appear to require excessive amounts of computational time and storage. However, we demonstrate that with a carefully optimized implementation it can be used to mesh three-dimensional images of practically significant sizes even on a regular desktop workstation. Furthermore, our time measurements show that for two complex medical atlas images (brain and abdominal) it is 28% to 42% faster than a state-of-the art Delaunay software.

The rest of the paper is organized as follows. In Section 2 we describe the proposed algorithm in detail. In Section 3 we present the implementation details along with the experimental evaluation. Section 4 concludes the paper.

2. ALGORITHM

The proposed Lattice Decimation (LD) algorithm works both for 2D and for 3D images. For explanation purposes, in Figure 2 we show a simple 2D example of an image being converted into a triangular mesh. The size of this image is 50×50 voxels. It defines two circular objects, which could represent tissues or materials, one within another, shown with different colors (cyan and magenta) against white background.

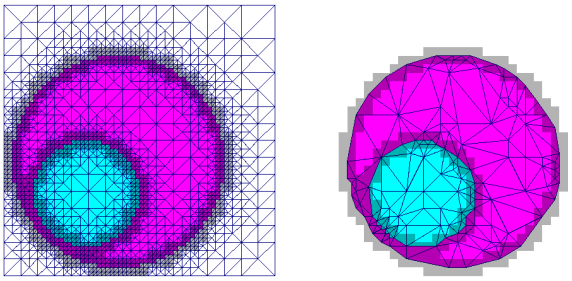

Figure 2: An illustration of the main steps performed by our LD I2M algorithm. The angle bound is set to 20°, and the fidelity bounds are both set to two voxels. Left: The initial fine mesh which fills in the quadtree, 2076 triangles inside the circles, 3534 triangles total. Right: The final decimated mesh, 263 triangles inside the circles, the outside triangles are removed. The inter-tissue boundaries are within the marked leaves, and therefore within the requested fidelity tolerance.

The mesh has to provide a faithful representation of the underlying tissues, i.e., each element needs to be marked with the physical properties of a unique type of tissue. To measure the distance between the boundaries of the two regions (the image of a tissue and the corresponding submesh), we use the Hausdorff distance. It can be specified as either a two-sided distance, or a one-sided distance. For tissue boundary I and mesh boundary M, the one-sided distance from I to M is given by

$$H(I \to M) = \max_{i \in I} \min_{m \in M} d(i, m),$$

where $d(\cdot, \cdot)$ is the regular Euclidean distance. The one-sided distance from M to I is given similarly by

$$H(M \to I) = \max_{m \in M} \min_{i \in I} d(m, i).$$

Note that $H(I \to M)$ is generally not equal to $H(M \to I)$. The two-sided distance is symmetric:

$$H(I \leftrightarrow M) = \max\{H(I \to M), H(M \to I)\}.$$

2.1 Input

The input to our algorithm is a 2D or a 3D bitmap. Each voxel of the bitmap corresponds to a separate material or tissue, as indicated by a single label (color) assigned to this voxel. The user also supplies the desired angle lower bound and fidelity bounds. We will use starred letters θ^* and H^* to denote the bounds on the angle and the Hausdorff distance, respectively.

2.2 Construction of the Octree

We construct an octree (in 3D) or a quadtree (in 2D) that satisfies the following properties (see Figure 2):

1. The octree (equivalently, its root node) completely encloses all the tissues from the image, except possibly for the background voxels that can be ignored.

2. There is extra space, equal to or greater than the maximum of the fidelity parameters, between the tissues and the exterior boundaries of the octree.

3. The boundaries between the leaves correspond exactly to the boundaries between the voxels. This is possible by using integer coordinates corresponding to voxel indices.

4. No leaf contains voxels from multiple tissues. The nodes of the tree are split recursively until all of the leaves satisfy this condition.

5. The sizes of the octree leaves respect the 2-to-1 rule, i.e., two adjacent leaves must differ in depth by no more than one level.

2.3 Computation of the Distance Transform

A distance transform of an image is an assignment to every voxel of a distance to the nearest feature of the image. In our case, the features are the boundaries between the tissues, and the distance is measured in the usual Euclidean metric. We implemented the Euclidean Distance Transform (EDT) algorithm described by Maurer [21]. We chose this algorithm for two reasons: (1) its linear time complexity with respect to the number of voxels, and (2) it is formulated to work in an arbitrary dimension. We run the EDT computation on the extended image, i.e., the image is padded with imaginary background voxels (or truncated of the extra background voxels) to the size of the octree root node.

2.4 Labeling of Octree Leaves

For each leaf of the octree, we find the maximum distance to the inter-tissue boundaries, using the EDT values of the voxels enclosed by this leaf. In Figure 2 we marked the leaves that are within the tolerance (2 voxels in this example) with transparent gray filling.

2.5 Filling in the Octree

We process the leaves in the order of their size, starting with the smallest, in order to ensure the conformity of the mesh along the boundaries, see Figure 2 for a 2D example. The procedure is recursive on dimension: to *triangulate* an n-dimensional *face* of the leaf, first *triangulate* all of its $(n-1)$-dimensional sub-*faces*. If at least one of the $(n-1)$-dimensional sub-*faces* is split by a mid-point, introduce the mid-point of the n-dimensional *face* and connect to the elements of the $(n-1)$-dimensional *triangulation* of the sub-*face* to construct the n-dimensional *triangulation* of the *face*. If none of the sub-*faces* was split, use the diagonals of the *face*.

This procedure is equivalent to using a finite number of predefined canonic leaf triangulations, with the extra benefits of reducing manual programming labor and being applicable in an arbitrary dimension. For all possible resulting leaf triangulations we obtain a minimum dihedral angle of 35.26° in 3D or a minimum planar angle of 45° in 2D. Hence, these are the bounds that the algorithm can guarantee.

Once all octree leaves are filled with tetrahedra, we finish the construction of the mesh data structure by identifying face-adjacent tetrahedra, in order to facilitate the decimation procedure.

2.6 Mesh Decimation

We say vertex $\mathbf{u}$ is *merged* to vertex $\mathbf{v}$ if vertex $\mathbf{u}$ and edge $\mathbf{uv}$ are removed from the mesh, such that all tetrahedra (triangles) incident upon edge $\mathbf{uv}$ are also removed from the mesh and the remaining edges that were incident upon $\mathbf{u}$ now become incident upon $\mathbf{v}$. See Figure 3 for an illustration.

Our decimation algorithm is shown in Figure 4. We main-

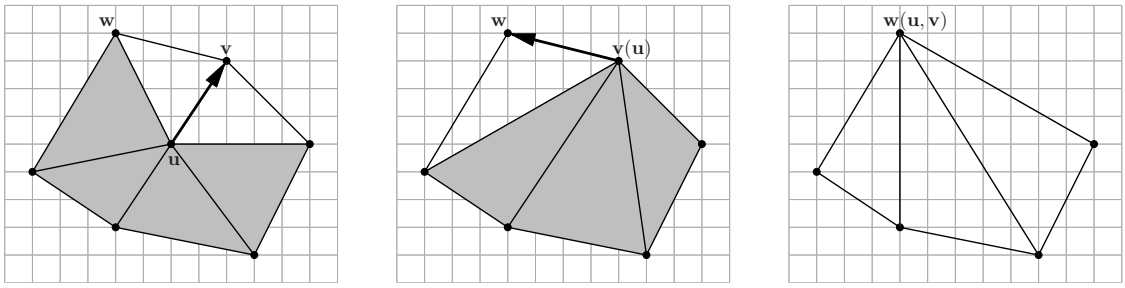

Figure 3: An illustration of the vertex merge operation. Left: Vertex u is evaluated for merging to vertex v. The shaded triangles need to be checked for the effect of changing their shape. Center: Vertex u is merged to vertex v. The list of vertices in brackets shows merge history. Vertex v is evaluated for merging to vertex w. Right: Vertex v is merged to vertex w.

tain a queue Q of mesh vertices that are candidates for merging. The algorithm removes from and adds vertices to Q until Q becomes empty. Note that after the initialization a vertex can be added on the queue only as a result of a merge of an adjacent vertex. Therefore, when none of the vertices in Q passes the check for a merge, Q will become empty and the decimation procedure will terminate. Suppose n is the total number of vertices in the original mesh. Every vertex is added to Q once in the beginning. Afterwards, a vertex is added to Q only if one of its vertex neighbors was merged. If m is the total number of merges performed (obviously $m < n$), then the total number of evaluations is bounded from above by $n + cm$, where c is the maximum number of vertex neighbors for each vertex. Assuming c is constant, the feasibility of performing a vertex merge is evaluated a linear number of times with respect to the original number of vertices in the mesh.

2.6.1 Maintaining Element Quality

The function CHECK4QUALITY(T, θ^*) returns *true* if and only if all elements on the list T are not inverted and have all angles (planar in 2D or dihedral in 3D) above the bound θ^*. Therefore, the merge is not accepted if at least one newly created angle is smaller than θ^*.

2.6.2 Maintaining Fidelity to Boundaries

This check, represented by the function CHECK4FIDELITY(T, $\mathcal{O}$, $H^*(I \to M)$, $H^*(M \to I)$) consists of two parts, for each of the one-sided Hausdorff distances. To evaluate the distance from the boundary of the sub-mesh to the boundary of the corresponding tissue, for each of the boundary faces (edges in 2D or triangles in 3D) of elements in T, we recursively check for the intersection with the octree nodes. If at least one of the faces intersects at least one of the nodes marked as outside the fidelity tolerance, the merge is discarded. To evaluate the distance from the boundary of each tissue to the boundary of the corresponding sub-mesh, for each vertex we maintain a cumulative list of the boundary vertices that were merged to it. If at least one of the boundary vertices, as a result of a sequence of merges, is further away from its original location than the corresponding fidelity tolerance, the merge is discarded.

2.6.3 Maintaining Tissue Connectivity

The geometric constructions used in our algorithm are assigned colors based on their location with respect to the tissues on the bitmap:

1. Each leaf of the octree (quadtree) derives the color from the block of voxels that it encloses; remember that the nodes are split recursively until they enclose voxels of a single color, in the limit case a leaf encloses a single voxel.

2. Each tetrahedron in 3D (or triangle in 2D) derives its color from the octree (quadtree) leaf that it is used to tetrahedralize; it keeps the original color even after it changes shape due to vertex merge. As a result, all tetrahedra (triangles) are always correctly classified with respect to the underlying tissues, including the quadruple-zero (triple-zero) ones.

3. Each mesh vertex derives its color from the block of incident voxels (eight in 3D or four in 2D); if the block of voxels has multiple colors, the vertex is considered *boundary*.

The following rules help us maintain the original structure of the inter-tissue boundaries: (1) boundary vertices cannot merge to non-boundary vertices, (2) a vertex cannot merge to a non-boundary vertex of a different color, and (3) a boundary vertex can merge to another boundary vertex only along a boundary edge—this helps to prevent the case when a vertex from one boundary merges to another boundary along a non-boundary edge, and thus the merge connects the parts of the boundaries that were not originally connected.

3. IMPLEMENTATION AND EVALUATION

We implemented the proposed Lattice Decimation (LD) algorithm in C++, in both two and three dimensions. The following implementation decisions have significantly improved the performance:

1. Most of the computation is performed in integer arithmetic. This is possible due to the fact that vertex coordinates are integers; they are indices with respect to the matrix of voxels. The only floating point computation is involved in the comparison of cosines of angles since long integer arithmetic could overflow. In addition, the lengths of the integer variables correspond to the range of values of each specific arithmetic operation, such that very long integers are used only when necessary to avoid overflow. For example, if variable x is represented with b bits, then x^2 requires $2b$ bits, while x^4 requires $4b$ bits; using $4b$ bits for x^2 would be excessive.

2. All expensive mathematical functions, such as trigonometric, square root, etc., including floating point division, are avoided in the computationally critical parts. Instead,

DECIMATION($\mathcal{M}$, $\mathcal{O}$, θ^*, $H^*(I \to M)$, $H^*(M \to I)$)
Input: $\mathcal{M}$ is the initial mesh
$\mathcal{O}$ is the octree
θ^* is the lower bound on the minimum angle bound
$H^*(I \to M)$ and $H^*(M \to I)$ are the upper bounds on one-sided Hausdorff distances
Output: Decimated mesh $\mathcal{M}$ that respects angle and fidelity bounds

```
1:  Initialize Q to the set of all vertices in M
2:  while Q ≠ ∅
3:    Pick v_i ∈ Q
4:    Q ⟵ Q \ {v_i}
5:    Find A = {v_j} the set of vertices adjacent to v_i
6:    for each v_j ∈ A
7:      Find T = {t_k} the set of tetrahedra incident
          upon v_i and not incident upon v_j
8:      for each t_k ∈ T
9:        Replace v_i with v_j in t_k
10:     endfor
11:     if (Check4Quality(T, θ*) ∧
          Check4Fidelity(T, O, H*(I → M), H*(M → I)) ∧
          Check4Connectivity(T, M))
12:       Merge v_i to v_j, update M
13:       Q ⟵ Q ∪ A
14:       break
15:     endif
16:     for each t_k ∈ T
17:       Replace v_j with v_i in t_k
18:     endfor
19:   endfor
20: endwhile
21: return M
```

Figure 4: A high level description of the decimation algorithm. The actual implementation is slightly different and more elaborate to support efficient data structures and to minimize computation, for more details see Section 3.

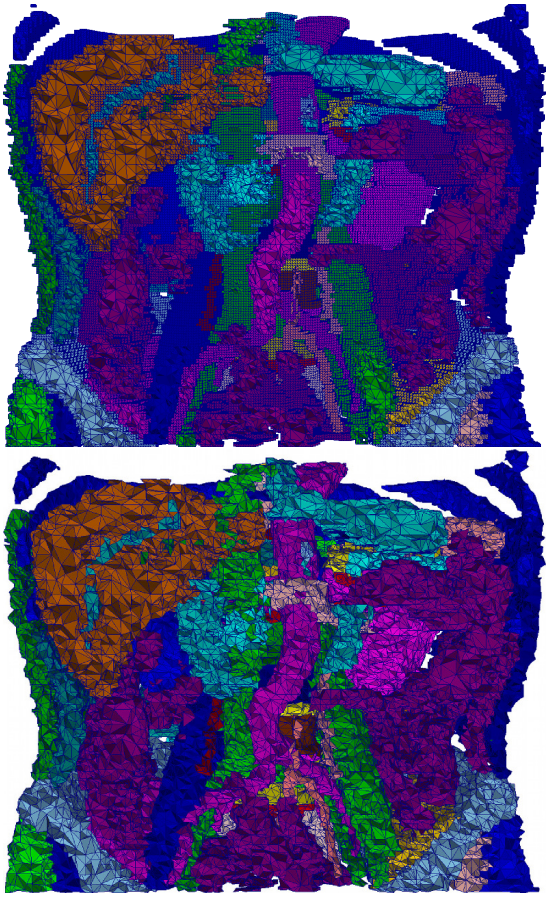

Figure 5: A slice through the LD mesh of the abdominal atlas for $\theta^* = 15°$. Top: $H^*(I \leftrightarrow M) = 0$, bottom: $H^*(I \leftrightarrow M) = 2$.

computation is performed on squares, cosines, and other functions of the original values.

3. We wrote customized memory allocation functions, such that objects that are created in large numbers but occupy little memory each (vertices, tetrahedra, nodes of the tree) are allocated in contiguous memory buffers. This improvement decreases memory fragmentation and allocation overheads.

4. We arranged the sequences of complex pass-fail condition evaluations such that the least expensive and the most likely to fail conditions are evaluated first, while the most expensive ones are evaluated last.

We evaluated the proposed algorithm using two publicly available complex real-world medical images: an abdominal atlas [30], and a brain atlas [31]. The atlases come with a segmentation, such that each voxel is assigned a label which corresponds to one of 75 abdominal and 149 brain tissues. All tests were performed on a desktop with an Intel Core i7 CPU @ 2.80 GHz and 8 GB of main memory. Machines with similar configurations are relatively inexpensive nowadays and can be obtained by virtually any practitioner.

The size of the abdominal atlas is $256 \times 256 \times 113$ voxels and the size of the brain atlas is $256 \times 256 \times 159$ voxels. In both cases each voxel has side lengths of 0.9375, 0.9375, and 1.5000 units in x, y, and z directions respectively. Before meshing the atlases, we resampled them with voxels of equal side length corresponding to the original 0.9375 units. As a result, in both cases we obtained equally spaced images that were used for meshing. Figures 5 and 7 show some resulting meshes for the atlases for various fidelity bounds.

In Table 1 we list the final number of tetrahedra, the smallest dihedral angle, and the total running time for both images, as we vary the H^* and θ^* parameters. To obtain a point of reference for these numbers, we conducted a separate experiment using a state-of-the art open source tetrahedral mesh generator Tetgen [28]. Tetgen is designed to work with Piecewise Linear Complexes (PLCs), and not images. Therefore, to make it process the same tissue geometries, we extracted the voxel faces corresponding to the boundaries between different tissues and between the tissues and the surrounding space, and saved them in the PLC format files that we passed to Tetgen. The main difference between the two methods is that Tetgen does not provide any guarantees on the dihedral angle (since it is designed to improve only circumradius-to-shortest edge ratio of tetrahedra for general PLCs), and its empirical smallest dihedral angle will generally be different for other input geometries. As can be expected, the meshes produced by Tetgen had low smallest dihedral angles, around 5°. At the same time, our LD algorithm can provide guaranteed smallest dihedral angles up to 35.26° for all input images.

For all of our time measurements we excluded all data preprocessing, such as image resampling, surface extraction,

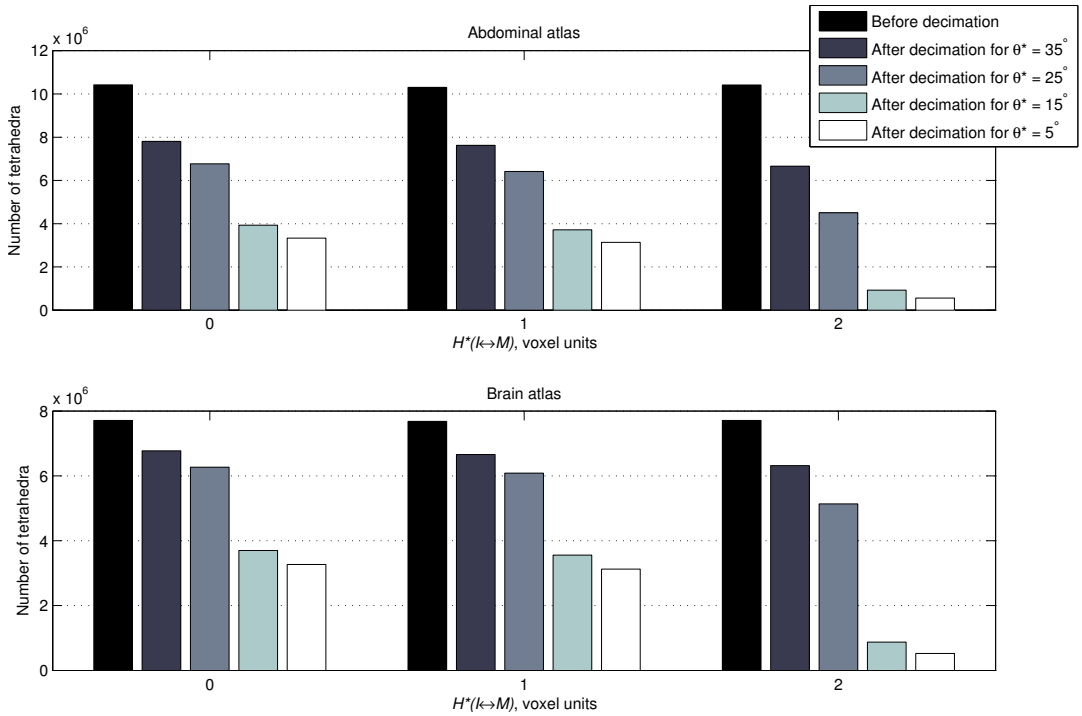

Figure 6: Final number of tetrahedra using LD, for varied θ^* and $H^*(I \leftrightarrow M)$.

and input/output. We see that our LD implementation is faster than Tetgen by a significant margin for both atlas images. Figures 8 and 9 show breakdowns of the total LD time into the main computational components as the symmetric Hausdorff distance bound changes from 0 to 2 voxels. We see little change both in the running time and in its distribution with the variation of H^* and θ^* parameters.

As far as the number of tetrahedra, the difference between Tetgen and LD is insignificant, although in both cases in favor of LD, for the bound of $H^*(I \leftrightarrow M) = 0$ which allows for a comparison with respect to the same fidelity, and $\theta^* = 5^\circ$ which is close to the empirical Tetgen angles. In Figure 6 we show the final number of tetrahedra for both atlases produced with the LD implementation, as we vary $H^*(I \leftrightarrow M)$ and θ^*.

4. SUMMARY

We presented a novel guaranteed quality and fidelity image-to-mesh conversion algorithm and its efficient sequential implementation. The algorithm preserves not only external boundaries, but also the boundaries between multiple tissues which makes the resulting meshes suitable for finite element simulations of multi-tissue regions with different physical tissue properties.

Our experimental evaluation shows that the decimation procedure produces meshes with fewer tetrahedra for weaker fidelity and dihedral angle bounds. This follows from the fact that weaker constraints allow for more opportunities for vertex removal. We expect that the decimation procedure can be further improved by adding extra degrees of flexibility from vertex movement and edge swapping operations. We are currently working on the development of this extended algorithm.

The algorithm and the implementation we presented are sequential. Our future work includes the development of the corresponding parallel algorithm and the code to increase the processing speed and the size of the images that can be handled. One stage of the algorithm, the distance transform, has already been parallelized [29]. However, according to the Amdahl's law, to achieve good speedup, we need to parallelize the other stages as well. We also plan to address the smoothness of mesh boundaries in order to improve the accuracy of such simulations as the blood flow.

5. ACKNOWLEDGMENTS

This work was supported (in part) by the NSF grants CCF-0916526, CCF-0833081, and CSI-719929, as well as by the Richard T. Cheng Endowment. We thank the anonymous reviewers for helpful comments.

6. REFERENCES

[1] D. Boltcheva, M. Yvinec, and J.-D. Boissonnat. Mesh generation from 3d multi-material images. In *Proceedings of the 12th International Conference on Medical Image Computing and Computer-Assisted Intervention: Part II*, pages 283–290, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver exudation. *J. ACM*, 47(5):883–904, 2000.

[3] L. P. Chew. Guaranteed-quality Delaunay meshing in 3D. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 391–393, Nice, France, 1997.

[4] P. Chopra and J. Meyer. Tetfusion: an algorithm for rapid tetrahedral mesh simplification. In *Proceedings*

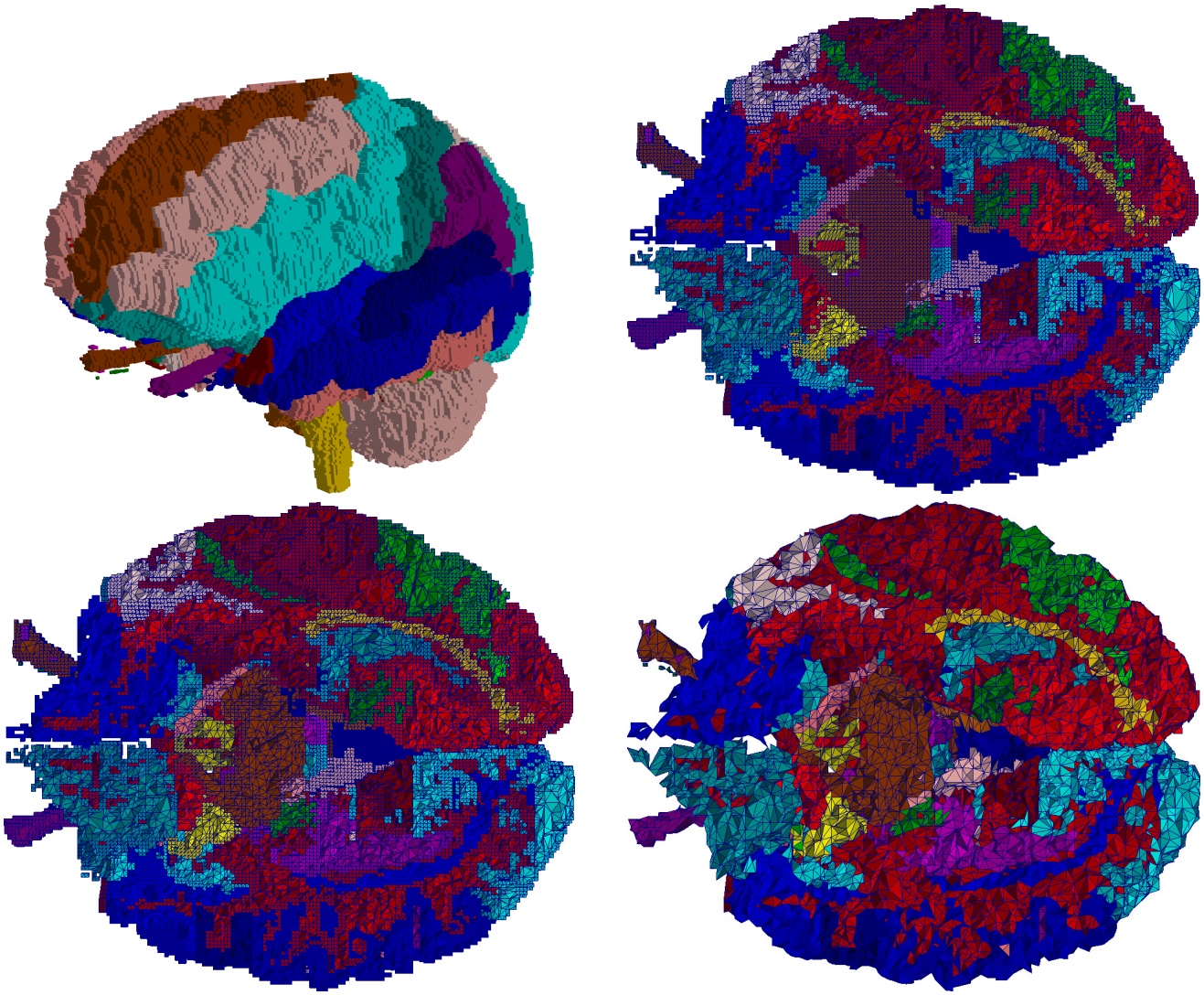

Figure 7: Top-left: the brain atlas. Top-right: a slice through the LD mesh of the atlas for $\theta^* = 15^\circ$ and $H^*(I \leftrightarrow M) = 0$. Bottom-left: same for $H^*(I \leftrightarrow M) = 1$. Bottom-left: same for $H^*(I \leftrightarrow M) = 2$.

of the conference on Visualization '02, VIS '02, pages 133–140, Washington, DC, USA, 2002. IEEE Computer Society.

[5] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17:167–174, 1998.

[6] O. Clatz, H. Delingette, I. F. Talos, A. J. Golby, R. Kikinis, F. A. Jolesz, N. Ayache, and S. K. Warfield. Robust non-rigid registration to capture brain shift from intra-operative MRI. *IEEE Transactions on Medical Imaging*, 24(11):1417–1427, 2005.

[7] E. Claus, A. Horlacher, L. Hsu, R. Schwartz, D. Dello-Iacono, F. Talos, F. Jolesz, and P. Black. Survival rates in patients with low-grade glioma after intraoperative magnetic resonance image guidance. *Cancer*, 103:1227–33, 2005.

[8] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 119–128, New York, NY, USA, 1996. ACM.

[9] G. Doğan, P. Morin, and R. H. Nochetto. A variational shape optimization approach for image segmentation with a mumford-shah functional. *SIAM J. Sci. Comput.*, 30:3028–3049, October 2008.

[10] A. Fedorov and N. Chrisochoides. Tetrahedral mesh generation for non-rigid registration of brain mri: Analysis of the requirements and evaluation of solutions. In *Proceedings of the 17th International Meshing Roundtable*, pages 55–72, Pittsburgh, PA, October 2008. Springer.

[11] P. Foteinos, Y. Liu, A. Chernikov, and N. Chrisochoides. An evaluation of tetrahedral mesh generation for non-rigid registration of brain mri. In *Computational Biomechanics for Medicine V, 13th International Conference on Medical Image*

Code	Input bounds		Resulting mesh statistics				Total time	
	$H^*(I \leftrightarrow M)$	θ^*	Number of tetrahedra		Smallest dih. angle			
			AA	BA	AA	BA	AA	BA
Tetgen	0 (implicit)	n/a	3,501,569	3,398,654	4.725	5.002	169	128
LD	0	5	3,332,477	3,267,276	5.002	5.003	122	74
		15	3,932,131	3,698,787	15.002	15.002	120	74
		25	6,768,472	6,266,442	25.066	25.066	120	73
		35	7,806,292	6,773,951	35.264	35.264	110	68
	1	5	3,134,565	3,126,393	5.000	5.005	134	82
		15	3,714,781	3,555,290	15.002	15.002	128	82
		25	6,415,049	6,082,604	25.066	25.066	127	75
		35	7,625,360	6,657,475	35.264	35.264	115	68
	2	5	557,242	521,952	5.000	5.002	111	71
		15	924,642	874,764	15.000	15.000	116	77
		25	4,506,340	5,137,417	25.061	25.066	135	83
		35	6,658,700	6,318,544	35.097	35.264	119	71

Table 1: Experimental evaluation of Tetgen and LD. AA stands for abdominal atlas and BA stands for brain atlas. Angles are measured in degrees, and time is measured in seconds.

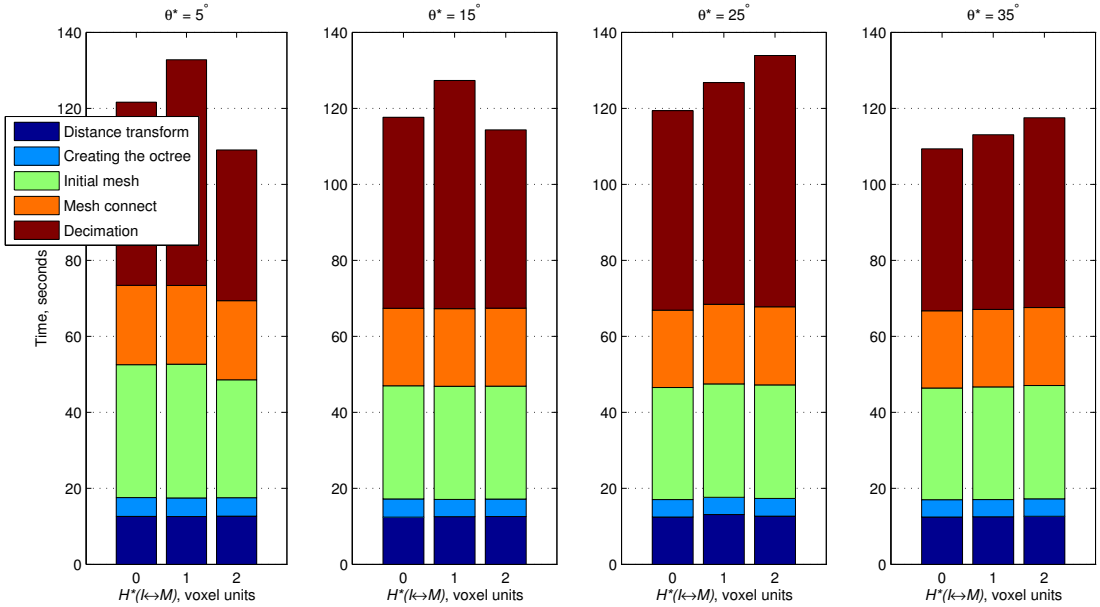

Figure 8: A breakdown of the total LD time for the abdominal atlas, for varied θ^* and $H^*(I \leftrightarrow M)$.

Computing and Computer Assisted Intervention (MICCAI) Workshop, pages 126–137, September 2010.

[12] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the conference on Visualization '98*, VIS '98, pages 263–269, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[13] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing. Application to Finite Elements*. HERMES, 1998.

[14] O. Goksel and S. E. Salcudean. Image-based variational meshing. *IEEE Transactions on Medical Imaging*, 30(1):11–21, January 2011.

[15] U. Hartmann and F. Kruggel. A fast algorithm for generating large tetrahedral 3d finite element meshes from magnetic resonance tomograms. In *Proceedings of the IEEE Workshop on Biomedical Image Analysis*, WBIA, pages 184–192, Washington, DC, USA, 1998. IEEE Computer Society.

[16] S. Hentschel and R. Sawaya. Optimizing outcomes with maximal surgical resection of malignant gliomas. *Cancer control : journal of the Moffitt Cancer Center*, 10(2):109–114, 2003.

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 19–26, New York, NY, USA, 1993. ACM.

[18] F. Labelle and J. R. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57.1 – 57.10, 2007.

[19] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshes in 3D. In *Proceedings of the 12th*

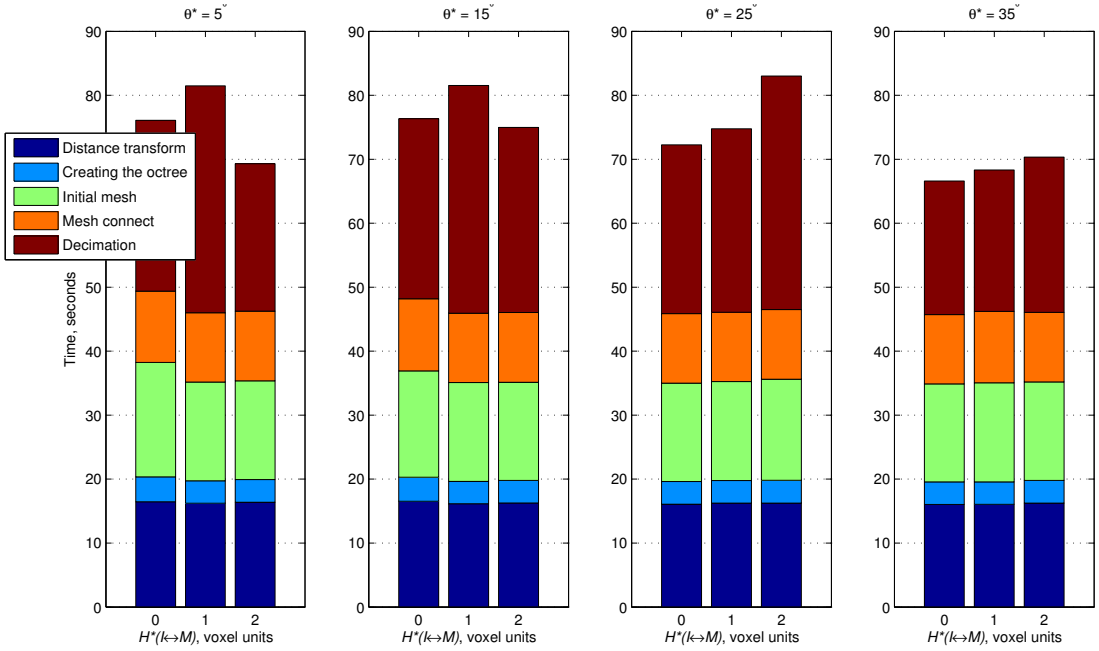

Figure 9: A breakdown of the total LD time for the brain atlas, for varied θ^* and $H^*(I \leftrightarrow M)$.

annual ACM-SIAM symposium on Discrete algorithms, pages 28–37, Washington, D.C., 2001.

[20] Y. Liu, P. Foteinos, A. Chernikov, and N. Chrisochoides. Multi-tissue mesh generation for brain images. In *Proceedings of the 19th International Meshing Roundtable*, pages 367–384, Chattanooga, TN, October 2010. Springer.

[21] C. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[22] R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, 2000.

[23] J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven. High-quality consistent meshing of multi-label datasets. In *Proceedings of the 20th international conference on Information processing in medical imaging*, pages 198–210, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] A. C. Reid, S. A. Langer, R. C. Lua, V. R. Coffman, S.-I. Haan, and R. E. Garcia. Image-based finite element mesh construction for material microstructures. *Computational Materials Science*, 43(4):989–999, 2008.

[25] C. Roarty and N. Grosland. Adaptive meshing for orthopaedic finite element contact problems. *Iowa Orthop Journal*, 24:21–29, 2004.

[26] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 65–70, New York, NY, USA, 1992. ACM.

[27] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[28] H. Si. Tetgen version 1.4.3. http://tetgen.berlios.de/.

[29] R. Staubs, A. Fedorov, L. Linardakis, B. Dunton, and N. Chrisochoides. Parallel n-dimensional exact signed Euclidean distance transform. *Insight Journal*, 2006. http://hdl.handle.net/1926/307.

[30] I. Talos, M. Jakab, and R. Kikinis. SPL abdominal atlas 2010. http://www.spl.harvard.edu/publications/item/view/1918, 10 2010.

[31] I. Talos, M. Jakab, R. Kikinis, and M. Shenton. SPL-PNL brain atlas. http://www.spl.harvard.edu/publications/item/view/1265, 03 2008.

[32] J. Tournois, R. Srinivasan, and P. Alliez. Perturbing slivers in 3D Delaunay meshes. In B. W. Clark, editor, *Proceedings of the 18th International Meshing Roundtable*, pages 157–173. Springer, 2009.

[33] I. Trotts, B. Hamann, and K. Joy. Simplification of tetrahedral meshes with error bounds. *Visualization and Computer Graphics, IEEE Transactions on*, 5(3):224–237, 1999.

[34] J. Yan, P. Shi, and D. Zhang. Mesh simplification with hierarchical shape analysis and iterative edge contraction. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):142–151, 2004.

[35] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5083–5106, 2005.