Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2016) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

25th International Meshing Roundtable

Tetrahedralization of Multi-material Images with Quality and Hausdorff Distance Guarantees

Jing Xu*, Andrey N. Chernikov

Old Dominion University, Department of Computer Science, 4700 Elkhorn Ave, Norfolk, VA, 23508, USA

Abstract

We present a method for generating three-dimensional unstructured tetrahedral meshes of multi-material images. The method uses an octree as the background grid from which to build the final graded conforming meshes. The algorithm is fast and robust. It produces a small number of mesh elements and provides guaranteed bounds on the smallest dihedral angle and the two-sided Hausdorff distance between the boundaries of the mesh and the boundaries of the materials. The technique is illustrated with examples.

Keywords: multi-material; tetrahedralization; quality; fidelity

1. Introduction

Meshing techniques have been proposed that can mesh domains as general as piecewise smooth complexes. This class includes *Piecewise Linear Complex*, which is usually defined by an object surface. The challenge is that the quality of the input PLC affects the quality of the final mesh because the mesh has to match exactly to the boundaries of the model. This class also includes the smooth and piecewise smooth surfaces, and above all non-manifolds. One approach to solve these problems is that the input is assumed to be an implicit function $f : R^3 \rightarrow Z$ such that points in different regions of interest evaluate f differently. The most widely used guaranteed-quality approach is based on the Delaunay refinement [2,3,5]. However, in three-dimensions, it allows only for the circumradius-to-shortest-edge ratio bound of the tetrahedra. Even if this ratio is very small, Delaunay refinement can not avoid the almost flat tetrahedra [5]. Another approach employs a space-tiling background grid to guide the creation of a mesh [4,7,8], the focus of this note. The Marching Cubes algorithm [8] computes f at the vertices of a cubical grid, and processes the domain cube by cube that approximates the intersection of the isosurface with that cube using triangles. However, the resolution of the background grid is on pixel level, and the mesh has no grading and quality guarantee. Isosurface stuffing [7] is a guaranteed-quality tetrahedral meshing algorithm for general surfaces. It offers the fidelity guarantee from the mesh to the model, and if the surface is a smooth manifold with bounded curvature and the background grid

*Corresponding author.
E-mail address: jxu@cs.odu.edu, achernik@cs.odu.edu

is sufficiently fine, it also guarantees the fidelity from the model to the mesh. The authors also have a fully graded mesh generator which generates meshes whose surface tetrahedra vary in size too. However, the quality guarantee is no better than 1.66°. This work is based on the Lattice Decimation (LD) method [4]. The decimation step is a heuristic which works well in practice, but it does not guarantee the optimal solution. Moreover, it does not offer sufficient control over the grading of the resulting mesh. However, if the starting mesh for the decimation is more coarse, both of these issues can be mitigated.

This note develops a fast generation method for tetrahedral volume meshes which satisfy the following requirements: 1. The quality requirement, i.e., we guarantee that all dihedral angles are above a user-specified lower bound which can be set to any value up to 19.47°. 2. The fidelity requirement, i.e., the two-sided Hausdorff distance between the boundaries of the mesh and the boundaries of the materials respects the user specified fidelity bounds. 3. The number of tetrahedra in the mesh is much smaller because the proposed method generates relatively coarser mesh before the post-processing decimation procedure. 4. The mesh can be constructed within tight real-time time constraints. 5. It offers not only the volume grading, but also the surface grading.

2. Algorithm

This work builds upon the LD algorithm to construct the octree and fill the octree leaves with tetrahedra. The LD algorithm constructs an initial fine mesh with very high quality and fidelity. The mesh is constructed from the octree whose nodes are split recursively until no leaf contains voxels from multiple materials. In the contrast, the proposed algorithm allows for the octree leaves contain voxels from two materials, and approximates the boundary of the materials with a set of triangular patches such that the two-sided Hausdorff distance between those patches and boundary of the materials in the leaf respects the user specified fidelity bounds. The proposed algorithm is described as follows: the algorithm takes a two- or a three-dimensional bitmap as its input. Each voxel of the bitmap has a single label. Different labels correspond to different materials. The user also specifies as input the target fidelity bounds (two-sided Hausdorff distance) and the desired angle lower bound (less or equal to the angle bound 19.47°). The algorithm first constructs an octree from which it generates a waterproof surface mesh such that the two-sided Hausdorff distance between the boundaries of the mesh and the boundaries of the materials in each leaf respects the user specified fidelity bounds. Then the algorithm fills the octree leaves with high quality tetrahedra from the pre-defined look-up table. As the last step, the algorithm coarsens the mesh to a much lower number of elements while maintaining the fidelity and quality requirements. We elaborate each step below.

2.1. Construction of the octree

The algorithm first constructs the octree that completely encloses all the materials from the bitmap. The boundaries between the octree leaves correspond exactly to the boundaries between the voxels. Besides that, an extra space between the materials and the exterior boundaries of the octree should be equal to or larger than the user specified fidelity bounds. Then the algorithm iteratively generates waterproof surface meshes until the two-sided Hausdorff distance between the boundaries of the mesh and boundaries of the materials in each leaf satisfies the user specified fidelity bounds. The octree construction algorithm is shown in Figure 1.

Initially, the algorithm splits the octree such that each leaf contains no more than two materials, and the sizes of the leaves respect the 2-to-1 ratio. Then it generates a waterproof surface mesh from the pre-defined look-up table. The function FIDELITY_SATISFIED($H^*(I, M)$, $H^*(M, I)$) computes the two-sided Hausdorff distance between the boundaries of the mesh and boundaries of the materials in each leaf. It splits the leaf into 8 children if the Hausdorff distance of either side is larger than the user specified fidelity bounds, and returns true if no leaf was split. The algorithm repeatedly checks the 2-to-1 ratio and generates a waterproof surface mesh until the two-sided Hausdorff distance respects the user specified fidelity bounds.

2.1.1. Waterproof surface mesh

When the octree leaf sizes respect the 2-to-1 ratio, the algorithm generates triangular patches from each octree leaf such that all the patches form a waterproof surface mesh. We borrow the idea from the Marching Cubes algorithm [8]. It computes cut function f at the vertices of a cubical grid, then approximates the intersection of the isosurface with

```
OCTREE CONSTRUCTION($B$, $H(I, M)$, $H(M, I)$)
Input: $B$ is a 3D bitmap
       $H^*(I, M)$ and $H^*(M, I)$ are the fidelity upper bounds
Output: Water-proofed surface mesh $\mathcal{M}$ such that the two-sided Hausdorff distance between the image boundary
        and mesh boundary satisfies $H(I, M)$, $H(M, I)$, and the octree $T$
1: Construct the octree $T$ that completely encloses all the materials in $B$ with extra space MAX($H^*(I, M)$, $H^*(M, I)$)
2: Split $T$ until each leaf contains no more than two materials
3: repeat
4:     Split $T$ until the sizes of the leaves respect the 2-to-1 ratio
5:     $\mathcal{M} = \emptyset$
6:     Generate a waterproof surface mesh $\mathcal{M}$ from the pre-defined look-up table
7: until FIDELITY_SATISFIED($H^*(I, M)$, $H^*(M, I)$)
8: return $\mathcal{M}$ and $T$
```

Fig. 1: Pseudocode of the octree construction algorithm.

that cube. In contrast to the Marching Cubes algorithm, the vertices evaluate to three values, positive (where the vertex located in the material whose label is larger in the leaf), negative (where the vertex located in the material whose label is smaller in the leaf), and zero (where the vertex located exactly on the boundary of the material). The templates on cube would be cumbersome for our algorithm, because there would be 3^8 cases needed to be analyzed. Instead, we designed a two dimensional case table on square for each cube face. Our table generates edges on the cube faces, and we generate triangular patches by connecting those edges with the center of the cube. The idea of designing the table is to connect all the zero valued vertices. Those zero valued vertices include the vertices of the cube located exactly on the boundary of the material, and the vertices that are generated by calculating the central point along an edge of the cube where the two ends have the opposite values.

2.1.2. Two-sided Hausdorff distance

To measure the Hausdorff distance from the boundaries of the surface mesh to the boundaries of the image, we compute the Euclidean distance transform [9] (EDT) of the extended image (same size as the octree), and we split the octree until no leaf has the distance of EDT both larger and within the input fidelity bound. We mark the leaves that are within the input fidelity tolerance. We call this splitting a virtual splitting, and call the leaves by this splitting virtual leaves because this splitting is only used to verify the fidelity condition, and our mesh is not generated based on filling those leaves. If one of the triangular facets of the surface mesh in the leaf intersects at least one of the virtual leaves that marked as outside the fidelity tolerance, the fidelity condition is violated and the leaf is split.

To measure the Hausdorff distance from the boundaries of the image to the boundaries of the corresponding surface mesh, we compute the shortest distance from each point located on the image boundary in the leaf to the triangular patches of the surface mesh in the leaf. If one of the image boundary points has a shortest distance larger than the fidelity tolerance, the fidelity condition is violated and the leaf is split.

2.2. Filling in the octree

When the waterproof surface mesh respects the user specified fidelity bounds, the octree is constructed and we fill the octree leaves with high quality tetrahedra using the second look-up table. The second look-up table is based on the same idea as the first look-up table, however, instead of being used to generate edges on cube faces, it is used to generate triangles on cube faces. The final mesh is obtained by connecting those triangles on the cube faces with the center of the cube to form tetrahedra. From the stencils, the minimum dihedral angle bound is 19.47°. The proof of this bound is omitted here due to the limited length of this note.

2.3. Mesh decimation

Similar to the Lattice Decimation method [4], we use the vertex removal operation to coarsen the mesh. A vertex can not be merged along an edge to another vertex if it violates the following requirements: 1. The quality requirement, i.e., if at least one newly created angle is smaller than the input quality angle bound. 2. The fidelity requirement, i.e., if at least one newly created mesh boundary edges has at least one side Hausdorff distance larger than the input fidelity bound. 3. The material connectivity is not maintained.

3. Experimental results

We applied the proposed octree refinement and decimation (ORD) algorithm to real medical data. All the experiments were conducted on a 64 bit machine equipped with two 3.06 GHz 6-Core Intel Xeon CPU and 64 GB main memory. The algorithm was implemented in C++, in both two and three dimensions. The size of the *knee atlas* [6] is 512 * 512 * 119 voxels, and the voxel has side lengths of 0.27, 0.27, and 1 units in x, y, and z directions. The size of the *abdominal atlas* [1] is 512 * 512 * 219 voxels, and the voxel has side lengths of 0.96, 0.96, and 2.4 units in x, y, and z directions. Before meshing, we re-sampled them with voxels of equal side length corresponding to the minimum spacing size to obtain equally spaced images.

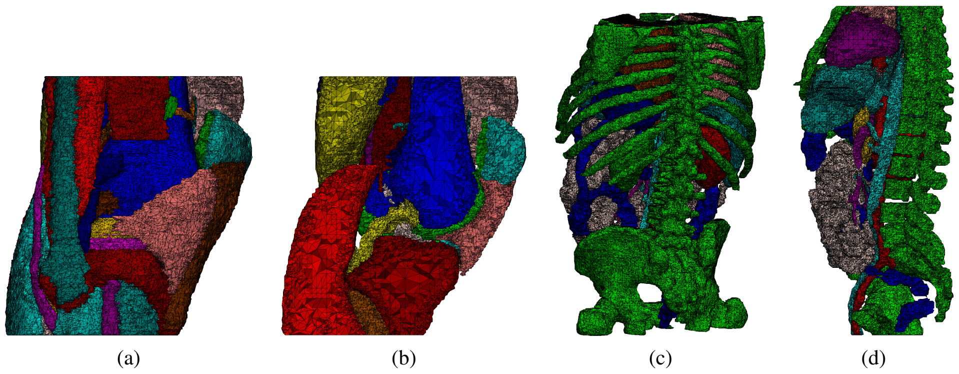

Fig. 2: Final meshes generated by the presented algorithm on *knee atlas* and *abdominal atlas* and their cut views.

Table 1: The comparison of final number of tetrahedra for ORD and LD

Hausdorff distance	$H^*(I,M) = 1, H^*(M,I) = 1$				$H^*(I,M) = 2, H^*(M,I) = 2$			
Input	*knee atlas*		*abdominal atlas*		*knee atlas*		*abdominal atlas*	
Algorithm	ORD	LD	ORD	LD	ORD	LD	ORD	LD
Before decimation	33,036,944	46,544,124	30,361,224	42,344,304	21,692,689	56,415,318	18,492,773	51,284,042
After decimation with $\theta = 19.47°$	13,938,895	19,032,469	13,027,911	17,608,762	5,185,603	14,198,097	4,062,233	12,434,028
After decimation with $\theta = 15.00°$	10,716,576	12,588,781	10,275,628	11,994,099	2,100,759	3,837,660	1,593,965	3,426,170
After decimation with $\theta = 10.00°$	9,634,021	11,160,507	9,319,252	10,646,685	1,136,779	2,335,844	856,455	2,081,046
After decimation with $\theta = 5.00°$	8,739,663	9,935,281	8,510,971	9,488,867	740,685	1,881,087	580,192	1,644,226

Hausdorff distance	$H^*(I,M) = 3, H^*(M,I) = 3$				$H^*(I,M) = 4, H^*(M,I) = 4$			
Input	*knee atlas*		*abdominal atlas*		*knee atlas*		*abdominal atlas*	
Algorithm	ORD	LD	ORD	LD	ORD	LD	ORD	LD
Before decimation	23,433,138	62,748,618	19,637,745	57,723,164	24,198,835	67,870,902	19,968,161	61,672,648
After decimation with $\theta = 19.47°$	5,347,342	15,474,919	4,164,029	13,672,489	5,402,816	15,245,351	4,243,052	13,238,118
After decimation with $\theta = 15.00°$	2,089,062	3,522,968	1,560,986	3,035,179	2,160,421	3,414,838	1,628,551	2,940,484
After decimation with $\theta = 10.00°$	1,096,040	1,771,489	825,285	1,592,756	1,166,676	1,629,635	885,394	1,383,768
After decimation with $\theta = 5.00°$	738,800	1,377,239	559,969	1,188,149	775,080	1,273,031	604,498	1,079,160

In Figure 3, we show the final meshes generated by the presented algorithm on *knee atlas* and *abdominal atlas* and their cut views. In Figure 3a, the symmetric Hausdorff distance was set 4 voxels and the minimum dihedral angle bound is 18°. In Figure 3c, we set the symmetric Hausdorff distance 5 voxels and the minimum dihedral angle bound 19.47°.

In Table 1 we show the number of tetrahedra for the final meshes of the input images, as we vary $H^*(I,M)$ and θ. For each fixed value of parameter $H^*(I,M)$, the first row shows the number of tetrahedra before decimation, and the rest of the rows show the final number of tetrahedra after decimation with different values of parameter θ. The tetrahedra generated by the ORD algorithm before decimation are much fewer than the tetrahedra generated by the LD algorithm before decimation. When $H^*(I, M) = 1$ and $H^*(M, I) = 1$, the tetrahedra in the final meshes of the ORD algorithm is slightly fewer than the tetrahedra generated by the LD algorithm. For all the other cases, the ORD algorithm generated a significantly smaller number of tetrahedra compared to the number of tetrahedra generated by the LD algorithm.

Figure 4 shows breakdowns of the total running time into the main computational components as the symmetric Hausdorff distance bound changes from 1 to 4 voxels. The octree construction of the ORD algorithm is more expensive than the one of the LD algorithm, because most efforts were spent there to generate a coarser mesh. However, the time was saved in the initial mesh construction step and decimation step because of the much coarser mesh.

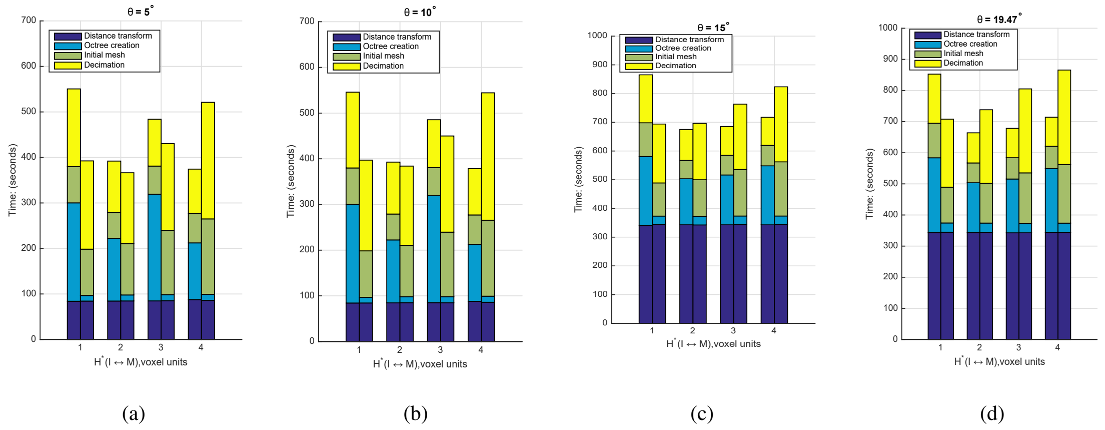

Fig. 3: Comparison of the breakdowns of the total running time into the main computational components for ORD and LD. For each value of parameter $H^*(I, M)$ in each graph, the left bar is the ORD time and the right bar is the LD time. (a) and (b) is for the *knee atlas* mesh, and (c) and (d) is for the *abdominal atlas* mesh.

4. Conclusion

We presented a novel approach for automatically constructing a guaranteed quality and fidelity mesh to represent geometry. The algorithm preserves not only external boundaries, but also the boundaries between multiple materials. Our future work focuses on the topology guarantee and its proof.

References

[1] Ircad Laparoscopic Center. http://www.ircad.fr/softwares/3Dircadb, 2013.
[2] Jean-Daniel Boissonnat and Steve Y. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67:405–451, 2005.
[3] Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay refinement for piecewise smooth complexes. *Discrete & Computational Geometry*, 43(1):121–166, 2010.
[4] Andrey Chernikov and Nikos Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing*, 33:3491–3508, 2011.
[5] Panagiotis Foteinos, Andrey Chernikov, and Nikos Chrisochoides. Guaranteed quality tetrahedral Delaunay meshing for medical images. *Computational Geometry Theory and Applications*, 47:539–562, 2014.
[6] Richolt J.A., Jakab M., and Kikinis R. Spl knee atlas, SPL 2015 Sep.
[7] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57.1–57.10, July 2007. Special issue on Proceedings of SIGGRAPH 2007.
[8] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987.
[9] Calvin R. Maurer, Jr., Rensheng Qi, and Vijay Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):265–270, February 2003.