Tetrahedral Mesh Generation for Medical Imaging*

Andriy Fedorov[1,2,3], Nikos Chrisochoides[1,2,3], Ron Kikinis[2], and Simon Warfield[2,3]

[1] Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA
[2] Surgical Planning Laboratory, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 02115, USA
[3] Computational Radiology Laboratory, Brigham and Women's Hospital, Children's Hospital, Harvard Medical School, Boston, MA 02115, USA

Abstract. We describe the open source implementation of an adaptive tetrahedral mesh generator particularly targeted for non-rigid FEM registration of MR images. While many medical imaging applications require robust mesh generation, there are few codes available. Moreover, most of the practical implementations are commercial. The algorithm we have implemented has been previously evaluated for simulations of highly deformable objects, and the preliminary results show its applicability to the targeted application. The implementation we describe is open source and will be available within Insight Toolkit.

1 Introduction

Finite Element Method (FEM) representations are widely used in medical imaging to enable intraoperative registration [1, 2] and biomechanical modeling of the tissue. The quality of geometric discretization is crucial for the effectiveness of these applications. Although numerous mesh generation methods have been described to date, there are few which can deal with medical data input. Even fewer algorithms have been implemented and evaluated. Software packages that can produce high quality meshes are usually commercial [3, 4].

In this paper we describe the on-going collaborative effort to produce open source mesh generation code within Insight Toolkit (ITK) [5]. Our motivating application is non-rigid image registration for preoperative planning and intraoperative navigation during neurosurgical procedures.

The paper is structured as follows. In Section 2 we discuss existing approaches to mesh generation. Section 3 describes the algorithm we have implemented. In Section 4 we present our implementation. Finally, we evaluate the implementation in Section 5 and conclude in Section 6.

* This investigation was supported in part by NSF grants ACI-0312980, EIA-0203974, ITR 0426558, a research grant from the Whitaker Foundation, a research grant from CIMIT, grant RG 3478A2/2 from the NMSS, and by NIH grants R21 MH67054, R01 LM007861 and P41 RR13218.

2 Related work

The target application for our implementation is FEM-based non-rigid registration of intraoperative MRI images for brain tumor resection procedures [2] and biomechanical simulation of the brain tissue deformation. The algorithm we choose has to satisfy the following requirements: (1) it should work directly with medical data (segmentations or greyscale images); (2) meshes must conform to the region of interest and have good quality (e.g., we can use minimal dihedral angle and aspect ratio [6] of a tetrahedron to evaluate its quality); (3) the algorithm should be capable of producing adaptive meshes; (4) intraoperative procedures require the algorithm to be very fast.

Both the bad quality and the large number of the mesh elements can negatively affect the execution time of an FEM simulation [1, 6]. Coarse discretization and poor shape of the elements can also introduce incorrect results and numerical errors. Hence, it is desired to have adaptive meshes which which provide guaranteed quality discretizations. Only Delaunay-based tetrahedralizations can produce meshes of guaranteed quality [7]. Unfortunately, Delaunay meshes may contain slivers (nearly flat tetrahedra with well-separated points). Advancing front techniques (AF) start from the boundary triangulation advancing the front inside the object. Bad quality tetrahedra may be introduced at the point where the fronts collide. AF methods are also highly dependent on the quality of surface discretization. Most of the practical techniques combine benefits of the Delaunay and advancing front approaches [3].

We observe, that there is no well-established methodology for mesh generation in the medical community. The reader is referred to [1, 6, 8] for surveys of existing approaches to mesh generation and their applicability to clinical applications. Implementations that produce high quality meshes are usually in the commercial domain [3, 4].

3 Algorithm

The baseline algorithm which we have implemented was developed by Molino for simulations of highly deformable objects [9]. The algorithm starts with building a uniform *body-centric cubic lattice* (BCC) enclosing the object, and applies iterative adaptive refinement procedure to create the mesh. The mesh obtained as the result of the adaptive refinement has guaranteed quality. Once the refinement procedure is complete, the topology of the *candidate mesh* is finalized. The tetrahedra which are guaranteed to be completely outside the object are discarded, as well as the tetrahedra which do not contain "enveloped" vertices. The resulting mesh does not conform to the object surface. Physics based FEM compression of the candidate mesh is used to align the mesh boundary. Both the refinement and compression procedures are using a level set definition of the object. At any point of space a scalar function $\phi(x)$ is defined as the signed distance function with negative values for the interior and positive – for the exterior of the object. For the details of the algorithm the reader is referred to the original

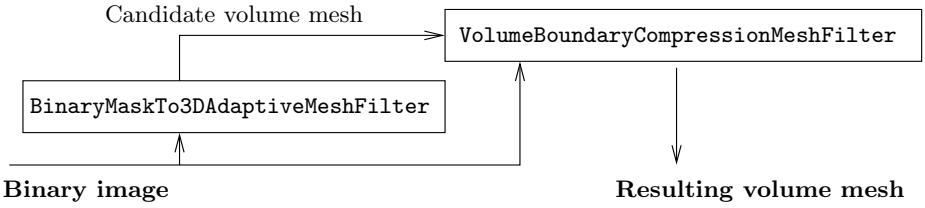

Fig. 1. Mesh generation pipeline.

paper describing the method [9]. Although the algorithm we have selected does not provide guarantees about the final mesh quality, Molino reports superior quality of the mesh in presence of high deformations.

4 Implementation

The implementation is structured as two ITK [5] classes, and we take advantage of a number of algorithms and tools implemented within ITK. We also use PETSc [10] (linear solver), and robust geometric primitives implemented by Shewchuk [11] (orientation test). Both of these codes are in the public domain.

The two main steps of the algorithm are (1) creation of the candidate mesh (*Mesher*), and (2) compression of the mesh surface to the object boundary (*Smoother*). We implemented these two stages as two separate ITK classes, as shown in Figure 1. The *Mesher* class is inherited from ITK `ImageToMeshFilter`. The input image is resampled to unit voxels, and then the distance and curvature maps are computed on the resulting image. Starting from the initial BCC lattice, at each resolution level all of the tetrahedra in the mesh are passed through the set of subdivision tests. The subdivision tests implemented within the *Mesher* filter check if a tetrahedron is crossing the object boundary, or is located in a region of high surface curvature. These tests are facilitated by the image distance and curvature maps. The API also allows to specify custom subdivision tests.

Once the tetrahedra which require subdivision are marked, we need to ensure that the mesh conforms to the object boundary. The mesh conformancy checks are repeated iteratively until we do not have new edges split. We use an edge-based data structure for mesh representation: each tetrahedra contains six pointers to its edges, so that we can identify the subdivision configuration in the constant time. After the specified number of refinement levels is reached, we discard tetrahedra which are guaranteed to lie outside the object of interest or which are not sufficiently interior to the object surface [9].

The *Smoother* filter is implemented as ITK `MeshToMeshFilter`. It accepts the binary mask and the mesh which approximates that mask. The "candidate" mesh is modeled as a physical body using FEM. We define the displacements of the boundary vertices toward the object surface using the distance map, and solve for displacements of the mesh vertices. The compression stage is performed using either ITK FEM module, or PETSc linear solver. We show some example cross-sections of the produced meshes in Figure 2.

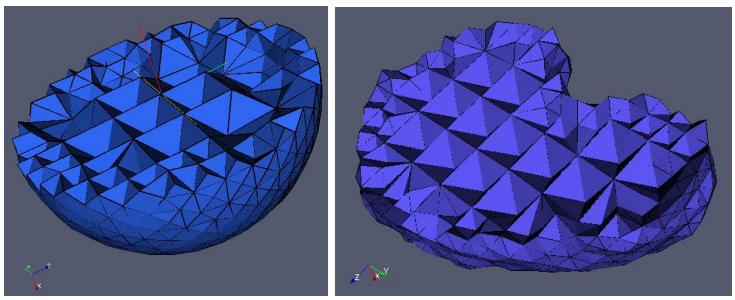

Fig. 2. Cross-sections of the adaptive meshes produced by the implementation.

The code is being developed within NAMIC SandBox SVN repository, and can be checked out using the following command: `svn checkout http://www.na-mic.org:8000/svn/NAMICSandBox/TetrahedralMeshGeneration`.

5 Experimental results

There are at least three important requirements the application imposes: (1) the resulting mesh should be of high quality; (2) the algorithm should be fast; (3) the mesh should precisely approximate the meshed object. Therefore, we evaluate the implementation along these lines.

The experiments were performed on an Intel Xeon workstation equipped with two hyperthreaded 3.06GHz CPUs, cache size 512Kb, 3.31Gb physical memory. The evaluation was done on three brain tissue segmentations from the retrospective neurosurgery cases. The datasets were selected from those used in [2], so that for each of the segmentations we had a tetrahedral mesh produced with the commercial software GHS3D [12].

Most of the running time is spent in data preparation: image resampling, distance transform and curvature calculation take approximately 5 minutes, while the refinement and boundary compression is complete in 10-25 seconds depending on the mesh size. In order to evaluate the quality of the resulting meshes we used tetrahedron aspect ratio and minimal dihedral angle [6]. The quality of the surface approximation was evaluated using the symmetrical Hausdorff distance [13] [4] between the mesh surfaces produced by our implementation and those extracted from the meshes generated by GHS3D. The results of the evaluation are summarized in Table 1.

The average Hausdorff distance is around 2 mm in all three cases. However, the maximum distance value reaches 7 mm, which is not acceptable (we consider the GHS3D mesh to be the "golden standard" for this comparison). At the same time, the meshes produced by our implementation have significantly better geometric properties in most cases. It is also important to note that the process

[4] We used the open-source implementation of the method available at `http://mesh.berlios.de`.

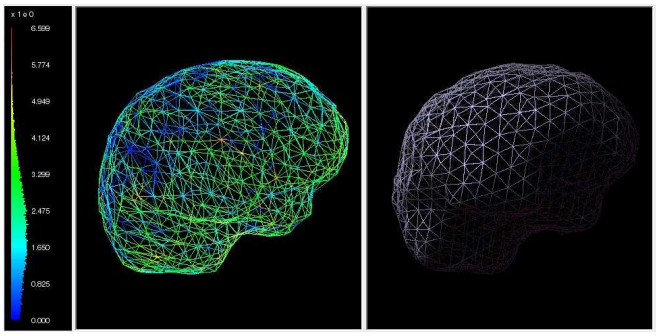

Fig. 3. Tetrahedral mesh generated with INRIA GHS3D (left) and the presented mesher (right) for case 2; the color-coding displays Hausdorff symmetrical distance between the two mesh surfaces.

of generating the mesh with GHS3D is quite complicated and involves multiple steps, as described in [2]. Our implementation allows to intuitively control the mesh size, operates directly on the medical data and does not use any commercial components.

6 Conclusions

We presented an implementation of the mesh generation algorithm based on adaptive refinement and FEM physics deformation. Our implementation is under development within NAMIC SandBox and will be available as a package within ITK. Preliminary evaluation shows applicability of the implementation for the targeted clinical application. We will continue our work on this method in order to obtain more experimental data, improve the surface approximation quality, and evaluate its applicability on other clinical applications. The presented

case #	*Size, tets*	*Hausdorff distance, mm*			*Aspect ratio*			*Min dihedral angle*		
		min	*ave*	*max*	*min*	*ave*	*max*	*min*	*ave*	*max*
1		0.0001	2.04	6.8						
1, *GHS3D*	7886				1.05	1.6	11.64	6.8	47.8	81.5
1, *RGM*	7565				1.02	1.36	2.5	25.8	55.6	75.6
2,		0.0001	2.1	7.1						
2, *GHS3D*	8202				1.05	1.62	6.7	11.16	47.7	83.3
2, *RGM*	7556				1.02	1.4	2.75	23.6	54.8	76.7
3		0.0001	1.74	7.14						
3, *GHS3D*	8235				1.02	1.6	19.1	3.07	47.7	83.2
3, *RGM*	7743				1.01	1.37	4.27	14.36	55.4	77.05

Table 1. Quantitative comparison of the generated meshes (Red Green Mesher, *RGM*) with the meshes produced by the INRIA GHS3D software.

implementation is the "pilot" code to analyze the viability of the approach. The surface representation accuracy has to be improved, and we believe we can achieve 1-2 mm accuracy of the segmentation approximation with our implementation. Although our mesher is very fast, we also plan to work on parallel distributed and shared memory implementations of the mesher [14] to facilitate parallel biomechanical simulations. Advanced biomechanical models may require significantly larger meshes than those we used for our evaluation.

Acknowledgments We would like to thank Aloys du Bois d'Aische, Matthieu De Craene, Matthieu Ferrant, Will Schroeder and Luis Ibanez for their help in developing this implementation.

References

1. Ferrant, M., Nabavi, A., Macq, B.M., Jolesz, F.A., Kikinis, R., Warfield, S.K.: Registration of 3D intraoperative MR images of the brain using a finite element biomechanical model. IEEE TMI **20** (2001) 1384–1397
2. Clatz, O., Delingette, H., Talos, I.F., Golby, A.J., Kikinis, R., Jolesz, F.A., Ayache, N., Warfield, S.K.: Robust non-rigid registration to capture brain shift from intra-operative MRI. Accepted for publication in IEEE TMI (2005)
3. Simulog: Mesh generation. <`http://www.simulog.fr/tetmesh`> (2005)
4. Amira: Mesh generation. <`http://www.amiravis.com`> (2005)
5. ITK: Insight toolkit. <`http://itk.org`> (2005)
6. Timoner, S.: Compact Representations for Fast Nonrigid Registration of Medical Images. PhD thesis, MIT (2003)
7. Shewchuk, J.R.: Tetrahedral mesh generation by Delaunay refinement. In: Proceedings of the 14th Symposium on Computational Geometry. (1998) 86–95
8. Goksel, O.: Ultrasound Image and 3D Finite Element based Tissue Deformation Simulator for Prostate Brachytherapy. MS thesis, The University of British Columbia (2004)
9. Molino, N., Bridson, R., Teran, J., Fedkiw, R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: Proceedings of the 12th International Meshing Roundtable. (2003) 103–114
10. PETSc: Portable, extensible toolkit for scientific computation. <`http://www-unix.mcs.anl.gov/petsc/petsc-2/`> (2005)
11. Shewchuk, J.R.: Fast robust predicates for computational geometry. <`http://www-2.cs.cmu.edu/~quake/robust.html`> (2005)
12. Simulog: GHS3D. <`http://www.simulog.fr/mesh/gener2.htm`> (2005)
13. Aspert, N., Santa-Cruz, D., Ebrahimi, T.: MESH: Measuring error between surfaces using Hausdorff distance. In: Proceedings of the IEEE International Conference on Multimedia and Expo 2002 (ICME). Volume I. (2002) 705–708
14. Chrisochoides, N.: A survey of parallel mesh generation methods. To appear in Numerical Solutions of Partial Differential Equations on Parallel Computers, Eds. M. Bruaset, P. Bjorstad, A. Tveto, <`http://www.cs.wm.edu/~nikos/pmesh_survey.pdf`> (2005)