Topology Sensitive Volume Mesh Simplification with Planar Quadric Error Metrics

Prashant Chopra, Joerg Meyer*
University of California, Irvine, Department of Electrical Engineering and Computer Science

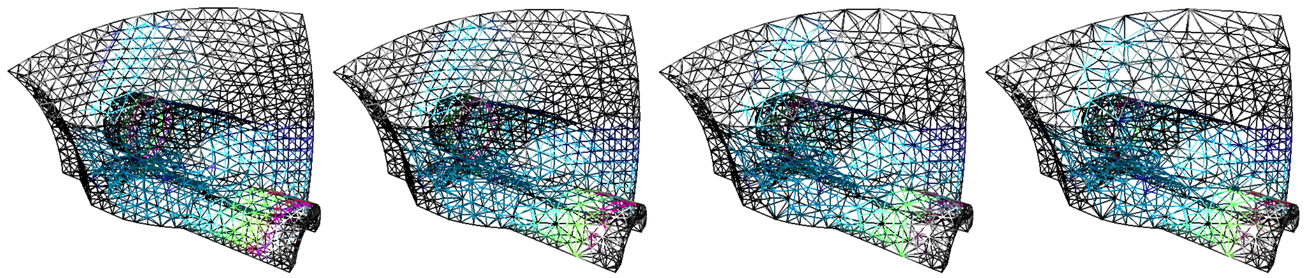


Figure 1: *Boundary envelopes* of four levels of detail of the *super phoenix* dataset (12,936 tetrahedral elements). The simplified mesh in the final image has only 46.58% of the original number of volume elements. CPU user time for simplification was less than one minute as measured on an SGI^{TM} O2+ MIPS R12000 400 MHz.

Abstract

We introduce an algorithm for fast topology-sensitive decimation of volume meshes. The algorithm employs a planar quadric error metric to guarantee minimum geometric error at every simplification step, while maintaining the input mesh's geometric consistency, and restraining changes to its topological genus, at low computational costs. The proposed method presents a new alternative to existing volume mesh decimation schemes, and lies between computationally intensive edge-collapse-based algorithms [1, 2, 3, 4], and a rapid topology-insensitive volume decimation approach [5]. The applications include, but are not limited to, rapid Level of Detail (LOD) generation, progressive volume rendering for scientific visualization applications, and representing volume meshes with minimal geometric complexity in computer games and entertainment media.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – surfaces and object representations.

Keywords: volume mesh simplification, multi-resolution, level-of-detail, unstructured meshes.

1. INTRODUCTION

Scientists and engineers frequently deal with complex volumetric models that are difficult to store, transfer, or render with available computing resources. Such models typically have an associated attribute domain space spanning over multiple dimensions (possibly time-varying scalar, vector, or tensor attributes, or even geometry) that adds to the complexity [6, 22, 8].

* 644E Engineering Tower, Irvine, CA 92697-2625
{pchopra | jmeyer}@uci.edu
http://vis.eng.uci.edu

In the recent past, this challenge has motivated computer scientists to explore algorithms that reduce the geometric complexity of such volume models. Almost all of the proposed methods either assume a tetrahedral mesh, or break down a tessellated volume into tetrahedra. Tetrahedra are preferred for discrete volume representation because of their simplicity, and because of their convex shape, which makes them suitable as volume rendering primitives. Most of these decimation algorithms evolved from efficient polygon-mesh simplification algorithms [9, 10, 11, 12, 13, 14, 15, 16, 17], and provide solutions for topology related inconsistencies that may occur during/after volume mesh simplification.

Recently, a rapid volume mesh decimation algorithm, *TetFusion*, was proposed [5]. This algorithm defines a tetrahedron as the most basic geometric primitive for collapse. However, it does not handle topological changes that might affect a mesh during simplification, and is limited to only interior tetrahedra of a volume (the boundary elements are not decimated at al*l*). We propose a volume mesh simplification algorithm that employs a *planar quadric error metric* to guarantee minimum geometric and attribute error upon each collapse. We show how a simple geometry decimation operation, along with an efficient geometric error metric, can take care of complex mesh inconsistency problems with minimal additional computational complexity. Our new algorithm ensures that a mesh stays within (and infinitesimally close to) its *boundary envelope* at all levels of resolution during simplification. The major contribution of this paper is a locally greedy progressive simplification algorithm for volume meshes, that

- is driven by a rapid space redistribution strategy,
- handles cases of mesh-inconsistency, while preserving the *boundary envelope* of the mesh, and
- employs a *planar quadric error metric* to guarantee minimum error in the geometric domain when collapsing the tetrahedral elements.

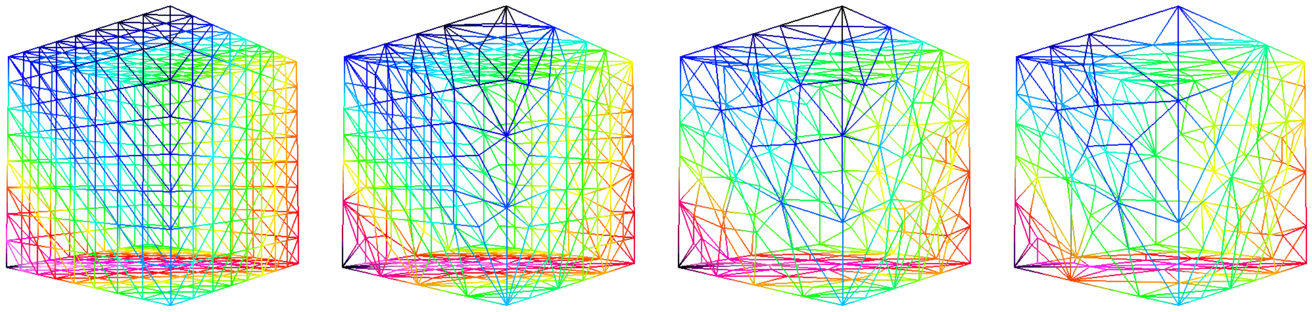


Figure 2: *Boundary envelopes* of four levels of detail of a synthetic *box* dataset (3072 tetrahedral elements originally). The original mesh was simplified by 64.48% in less than 10 seconds (on an SGI^{TM} O2+ MIPS R12000 400 MHz), with perfect boundary feature preservation. The color represents the scalar attribute associated with the mesh vertices; the primitives are rendered using smooth shading.

Section 2 describes related work in the general area of mesh simplification. It describes the challenges that arise when simplifying volume meshes, and the existing approaches used to address them. Section 3 describes our new algorithm: *QTetFusion*, which combines boundary and feature preservation in a single, computationally efficient algorithm. Section 4 presents the results of experimental runs of the algorithm on selected datasets. Section 4 concludes the paper with a discussion of merits and drawbacks of this new approach.

2. RELATED WORK

There have been a number of publications in the area of mesh simplification in recent years, targeting a reduction of the geometric complexity of both surface and volume meshes. Both refinement- and decimation-based strategies have been explored in this pursuit. We first briefly discuss important works in the area of polygonal mesh reduction, because they have historically influenced almost all of the existing volume mesh simplification algorithms (section 2.1). We then discuss existing volume decimation algorithms in section 2.2.

2.1 Surface Mesh Simplification

This section describes selected algorithms that simplify polygonal (surface) meshes. Most of the related work surveyed in this category is based on geometry reduction techniques (as opposed to refinement-based techniques), which iteratively decimate one or more primitives in a local or global error-metric guided fashion. Some of the methods then consolidate the mesh using the remaining (smaller) number of primitives. Some of the noted works in this area, that affected the efforts in volume mesh simplification also, are: Schroeder et al. [21] [22], Hoppe [13], Turk [27], and Garland and Heckbert [10] etc. For further literature, the authors suggest a reference to the survey report on multi-resolution modeling by Garland [9] for a detailed analysis of numerous relevant publications.

2.2 Volume Mesh Decimation

Only a few attempts have been made to address volume mesh simplification. We focus specifically on tetrahedral mesh decimation because of the common use of tetrahedra in representing unstructured volumetric models in the scientific community.

Trotts et al. [3] were amongst the first to implement a tetrahedral mesh decimation algorithm. They extend polygonal geometry reduction techniques to tetrahedral meshes, and present a tetrahedral collapse operation as a sequence of three edge collapses, while keeping the overall geometric error within a tolerance range. The error-metric used is based on a unique spline-segment representation of each tetrahedron. The paper discusses several problems and difficulties specific to tetrahedral mesh simplification. Further, because the decimation strategy is based upon successive ‘edge collapses’, the algorithm suffers from overwhelming overheads for the data structures to store edge information for massive volumetric datasets. The paper does not mention the high time complexity of the algorithm [3].

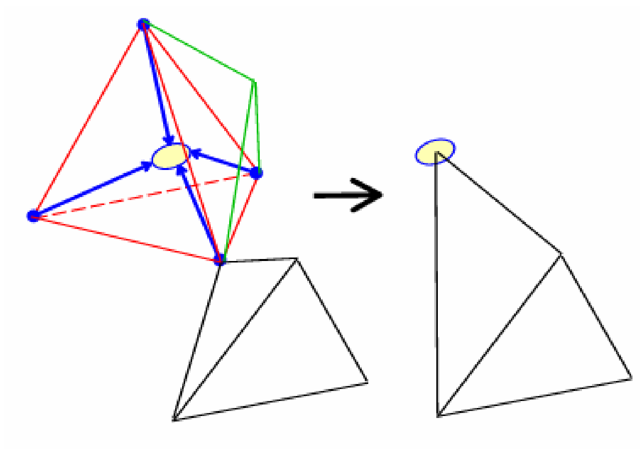
Staadt and Gross [2], in their work on progressive tetrahedralization, pursue a specialized class of simplicial complexes [13]: tetrahedra. The algorithm proposed is thus a conceptual extension of ‘edge collapse’ employed for progressive simplification of surface meshes [11]. The algorithm points out and provides solutions for previously undiscussed cases of ‘negative tetrahedra’ (flipping), self-intersections, and intersections at boundary regions. However, the dynamic mesh-consistency tests that are proposed as solutions are computationally expensive. As an example, the algorithm reportedly took about five hours to simplify a 576,576 tetrahedral elements mesh [2].

Trotts et al. [4], in an extension to their earlier work [3], incorporate a scalar-attribute preservation metric. The new algorithm ties the approximation error to the deviation of a simplified scalar field from the original values. They simplify the complex energy terms used for error evaluation in other methods. Noting the topological problems and degeneration cases that occur when using a sequence of three edge collapses (which require computationally expensive solutions), they revert back to a single edge collapse. Citing the results section from this publication, their best performing algorithm took 2,557 minutes (about 42 hours) to simplify a 449,890-element *blunt fin* dataset by 83.9%. The computational overhead introduced by these methods added to our motivation to explore alternative methods for simplification of volume meshes that are computationally less expensive.

Measuring domain error and accordingly calibrating simplification strategies for an accurate approximation is an important issue, which had been under-rated in volume mesh decimation research, until Cignoni et al. [1] presented a framework for integrated error evaluation for both domain and field approximations during simplification. They stress on accurate estimation of domain errors while employing 'edge-collapse'-based decimation strategies. The paper elaborately explores local accumulation, gradient difference, and brute force strategies to evaluate and predict domain errors while incrementally simplifying a mesh.

When addressing mesh approximation, topology preservation is another important concern for volumetric models (e.g., volume rendering applications in the engineering or scientific domain). In his recent work, Edelsbrunner [19] provides an extensive algorithmic background for ensuring topological correctness during edge-collapse based mesh simplification. Dey et al. [20] provide detailed criteria for topological correctness, which can be generalized to polyhedral meshes.

We are presenting a computationally efficient (tetrahedral) volume mesh simplification method that combines metrics for accurate data representation with techniques for restraining volumetric topology changes in a rapid tetrahedral mesh decimation algorithm.



(a) Before collapse (b) After collapse

Figure 3: An illustration of an instance of the *QTetFuse* operation. The upper-left red (*prey)* tetrahedron is the one to collapse onto its *fusion point*. The green tetrahedron degenerates into a line, which is removed consequently (*deleted tetrahedron*). The lower black (*affected*) tetrahedron stretches in the direction of the corresponding *fusion point*. Note that for the affected tetrahedron, the vertex it shares with the *prey tetrahedron* tends to move 'away' from the base plane formed by its other three vertices (flipping discouraged). If the shown *prey tetrahedron* is an *interior* one, at least eleven tetrahedra collapse as a result of its collapse (see section 3.2 a).

3. QUADRICS GUIDED FUSION

As discussed in the previous section, almost all of the existing work addressing volume mesh simplification evolved from 'edge-collapse'-based decimation strategies that were originally proposed for polygonal meshes. However, surface mesh simplification algorithms cannot simply scale up to handle higher order simplicial complexes because of additional geometric and topological constraints. Cases like degenerate simplices, violation of Delaunay tessellation, loss of topological type (undesired closing of holes or creation of new ones), intersection of volume elements at boundary regions, etc., should be specially taken care of during volume mesh simplification. Such cases have already been identified and can be avoided with special computational methods [20, 19]. However, algorithms that present computationally less intensive simplification schemes either do not handle all of these cases, or are spatially selective during decimation and hence cannot provide very high reduction ratios [5].

To overcome these shortcomings, we present *QTetFuse* as a reversible atomic decimation operation for tetrahedral meshes. The basic idea is to fuse the four vertices of a tetrahedron into a point (the *fusion point*, see the definitions in section 3.1). The fusion point is computed so that minimum geometric error is introduced during decimation. We employ a *planar quadric error metric* to measure and bound this error (described in section 3.3, see Figure 2 for an illustration).

3.1 Definitions

In this section, we introduce notations that are specific to the domain of this paper:

3.1 *Prey Tetrahedron*: A tetrahedron that is selected for decimation.

3.2 *Boundary Tetrahedron*: A tetrahedron with one or more of its vertices lying on the boundary surface. All of the tetrahedra that are non-*boundary* shall be called *interior tetrahedra* hereafter in this paper.

3.3 *Boundary Face*: Triangle face of a *boundary tetrahedron* all three of whose vertices lie on the boundary surface.

3.4 *Fusion Point*: Point of collapse of the four vertices of a prey tetrahedron. One prey tetrahedron may have more than one valid fusion point depending upon the specified planar quadric error tolerance value.

3.5 *Affected Tetrahedron*: A tetrahedron that shares exactly one vertex with a *prey tetrahedron*. This shared vertex stretches the affected tetrahedron towards, and onto the *fusion point* of the *prey tetrahedron* as a result of *TetFusion*.

3.6 *Prey Vertex:* The vertex of an *affected tetrahedron* that it shares with a *prey tetrahedron*.

3.7 *Base triangle*: A triangle formed by the vertices of an affected tetrahedron, excluding the prey vertex.

3.8 *Deleted Tetrahedron*: A tetrahedron that shares two or more vertices with a *prey tetrahedron,* which collapses as a result of the collapse of the *prey tetrahedron*.

Much of these definitions have been borrowed from the author's previous work TetFusion [5] for a clarity and conceptual extension.

3.2 Properties

This section discusses the inherent properties of *QTetFusion* (*Q*uadrics guided *Tet*rahedron *Fusion*) as a volume mesh decimation algorithm for tetrahedral meshes:

(a) High decimation: Each instance of *QTetFuse* operation decimates at least 11 tetrahedral elements for an *interior* or *non-boundary prey tetrahedron.* This number includes the *prey tetrahedron,* exactly 4 tetrahedra each sharing one of the four faces of the *prey tetrahedron,* and at least 6 more tetrahedra each one of which shares exactly one of the six edges with the *prey tetrahedron*. This generally means a 'higher' lower bound on the decimation ratio per step than an edge collapse operation. Using a tetrahedral collapse as a primitive operation avoids some of the topological problems and degeneration cases mentioned by Trotts et al. [4].

(b) No flipping: Upon each decimation step, one or more of the affected tetrahedra may suffer a 'flipping' or negation of the volume it represents. This can happen only when the vertex it shares with the corresponding *prey tetrahedron* (the *shared vertex*) flips sides with respect to its base plane (the plane formed by the other three vertices of the *affected tetrahedron*). To avoid such cases, an early rejection test is employed.

(c) Simplified mesh restricted to the inside (and infinitesimally close to) the *boundary envelope* **of the source mesh**: In the case of *QTetFusion*, self-intersections of boundary elements might occur when an *affected tetrahedron* pierces through one or more of the boundary faces of a *boundary tetrahedron*. We prevent such cases by restricting the simplified mesh to remain inside its *boundary envelope*, and by prohibiting flipping.

(d) Sensitivity towards topological genus of the mesh: Since the *boundary envelope* of a polyhedral mesh represents its topological genus; if the topological genus of the envelope is preserved, topology preservation for the enclosed volume mesh is guaranteed. Following from features (b) and (c) above, the algorithm guarantees that the simplified mesh remains confined (on)to its boundary envelope. This restrains the closure of any existing holes, as well as opening of any new ones. The latter is an inherent property of any edge-collapse based decimation operation however.

3.3 Planar Quadric Error Metric (*PQEM*)

This section describes the error metric we employ to bound the domain errors during simplification. Garland and Heckbert [9] developed a computationally efficient and intuitive algorithm employing a *Q*uadric *E*rror *M*etric (*QEM*) for rapid progressive simplification of polygonal meshes. The algorithm produces high quality approximations and can even handle 2-manifold surface meshes. We present a *P*lanar *Q*uadric *E*rror *M*etric (*PQEM*) based on the *QEM* for decimation of a volume element (see figure 3).
To obtain an error minimizing sequence of *QTetFuse* operations, we first need to associate a 'cost' of collapse with each tetrahedron in the mesh. As described in [9], we first associate a quadric error measure (a 4x4 symmetric matrix *Q*) with every vertex *v* of a tetrahedron that indicates the error that would be introduced if the tetrahedron were to collapse. For each vertex *v* of a tetrahedron, the measure of its squared distance with respect to all incident triangle faces (planes) is given by:

$$\Delta(v) = \Delta\left(\left[v_x v_y v_z 1 \right]^T \right) = \sum_{p = faces\,(v)} (a_p p^T v)^2 \tag{1}$$

where $p = [p_x\, p_y\, p_z\, d]^T$ represents the equation of a plane incident on *v* such that the weight a_p represents the area of the triangle defining *p*. Further, if *n* represents the normal vector of *p*, then *d* is given by

$$d = -nv^T \tag{2}$$

Equation (1) can be rewritten as a quadric:

$$\begin{aligned}\Delta(v) &= \sum_{p=faces(v)} v^T (a_p^2 pp^T) v \\ &= v^T \left(\sum_{p=faces(v)} (a_p^2 pp^T) \right) v \\ &= v^T \left(\sum_{p=faces(v)} (Q_p) \right) v \qquad (3)\end{aligned}$$

where Q_p is the area-weighted error quadric for *v* corresponding to the incident plane *p*.

Once we have error quadrics $Q_p(i)$ for all the four vertices of the tetrahedron *T* in consideration, we simply add them to obtain a single *PQEM* as follows:

$$PQEM(T) = \sum_{i=1}^{4} Q_P(i) \tag{4}$$

Now if *T* were to collapse to a point v_c, the total geometric error (for *T*) as approximated by this metric would be:

$$\Delta(T) = v_c^{\ T} PQEM(T) v_c \tag{5}$$

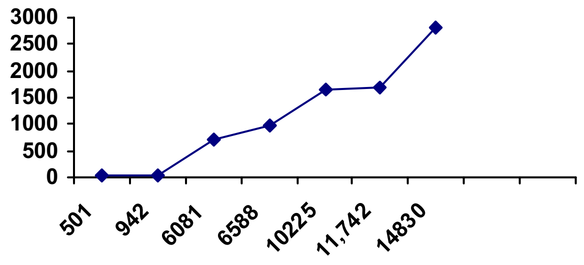


Chart 1: A chart depicting the CPU user time (seconds, vertical axis) based on the number of *QTetFuse* operations performed (horizontal axis) for the selected datasets.

3.4 Computing the *Fusion Point*

Consider a tetrahedron $T = \{v_1, v_2, v_3, v_4\}$. We compute a point of collapse (*fusion point* $\bar{v}$) for *T* that minimizes the total associated *PQEM* as defined in equation (5). According to [9], this can be done by taking the partial derivatives of $\Delta(T)$, and solving them for 0. The result is of the form

$$\bar{v} = Q_1^{-1}[0\ 0\ 0\ 1]^T \tag{6}$$

where

$$Q_1 = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Note that the terms q_{ij} are coefficients of the respective *PQEM*. There might be cases when the quadric matrix used in the equation is not invertible. In that case, we settle on the barycenter of *T* as the *fusion point*.

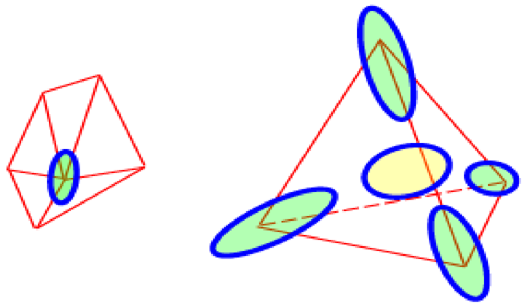

(a) Polygonal mesh (b) Polyhedral mesh

Figure 5: Error ellipsoids for affected vertices when the primitive to be decimated is (a) an edge and (b) a tetrahedron. Note that the yellow ellipsoid in b represents a level-set surface for the *P*lanar *Q*uadric *E*rror for the *prey tetrahedron* shown. This quadric error is the sum of quadric errors for the four constituent vertices of the tetrahedron.

3.5 The Algorithm

This section describes our locally greedy algorithm: *QTetFusion*. The method features a preprocessing phase that evaluates *PQEM*s for all the tetrahedra in the input mesh **M**, and stores them in a heap data structure. We employ a *Fibonacci* heap (because of its better amortized time complexity compared to a simple binomial heap) to maintain the priority queue of tetrahedral elements keyed on their *PQEM*s [21]. The main algorithm is outlined below:

```
while heap is not empty
    extract T with minimum $\Delta(T)$ from the heap
    if none of adjacentTetrahedra(T) would flip as a
    result of T's collapse,
        QTetFuse (T)
        update heap
```

Note that the pseudo procedure **adjacentTetrahedra(*T*)** returns a list of all the tetrahedra adjacent to *T*.

4. RESULTS AND CONCLUSION

Table-1 and *chart-1* summarize the results obtained from experimental runs of *QTetFusion* on selected datasets. The CPU usage results verify the hypothesis that *QTetFusion* lies between computationally intensive volume mesh simplification algorithms [1, 2, 3, 4], and its closest predecessor *TetFusion* [5]. For example in a comparative experiment for a reduction of 49.3% of the 187,395 elements *blunt fin* dataset, *TetFusion* took 15.480 seconds (SGI R10K 4x194 MHZ); while *QTetFusion* took 715.074 seconds (SGI R12K 450 MHZ) of CPU user time. However, *QTetFusion* presents three important advantages over *TetFusion* although bearing a slight additional computational complexity:

- Sensitivity towards changes in topological genus of the input mesh,
- reduction of boundary elements (previously not handled by *TetFusion*), and
- use of *planar quadric error metric* to ensure minimum local geometric error per decimation step (optimal fusion point instead of the volume element's barycenter).
- is feature sensitive (preserves boundary features like creases, sharp edges etc. by the use of weighted *planar quadric error metrics*),
- is more 'intuitive' for volumetric meshes than 'edge–collapse'-based methods.

Future work: The authors aim at the following goals towards a wider applicability of the algorithm in the future:

- development of error metrics for guided decimation of volume meshes with time-varying geometry and domain attributes, and
- incorporating offline geometry compression [22, 23, 24, 25, 26] with *QTetFusion* to develop a dynamic (on the fly) LOD management system for volume meshes as the one for polygonal meshes in [27].

Acknowledgements

This work has been sponsored by National Science Foundation under award no. 6066047–0121989 for the SPUR (Seismic Performance for Urban Regions) project [5, 11, 24]. We would like to thank our project partners at UC Berkeley, Carnegie Mellon University, and Mississippi State University for their continuing support.

We would like to thank Peter Williams, Lawrence Livermore National Laboratory for the *super phoenix* dataset, Ricardo Farias, LNCC Brazil, for providing the tetrahedral mesh subdivision code, and Michael Garland (UIUC) for an initial discussion on development of the algorithm. Thanks also to Ruparani Chittineni for helping with the final draft.

#	mesh	*n*	*R/R(%)*	*initHeap (s)*	*QTetFusion (s)*
1.	*super phoenix*	12,936	53.648	17.031	31.174
2.	*blunt fin*	187,395	49.294	295.071	715.074
3.	*comb chamber*	215,040	47.201	332.818	976.603
4.	*oxygen post*	513,375	46.462	810.119	2803.575

Table 1(a). Reduction ratios (***R/R***) and CPU user execution times (seconds) for heap initialization and actual *QTetFusion*.

#	mesh	*n*	n_{decim}	# QTetFuse	Avg. n_{decim}
1.	*super phoenix*	12,936	6,940	501	13.852
2.	*blunt fin*	187,395	92,375	6,081	15.684
3.	*comb chamber*	215,040	101,502	6,588	15.407
4.	*oxygen post*	513,375	238,527	14,830	16.084

Table 1(b). Number of *QTetFuse* operations and the number of tetrahedra decimated as a result (n_{decim}).

REFERENCES

[1] Cignoni, P., D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of Tetrahedral Meshes with Accurate Error Evaluation. In Thomas Ertl, Bernd Hamann, and Amitabh Varshney, editors, *Proceedings of IEEE Visualization 2000 (Salt Lake City, Utah, October 2000),* pages 85-92. IEEE Computer Society. 2000.

[2] Staadt, O. G., and M. H. Gross. Progressive Tetrahedralizations. In *Proceedings of IEEE Visualization 1998 (October 1998),* pages 397-402. 1998.

[3] Trotts, Isaac J., Bernd Hamann, and Kenneth I. Joy. Simplification of Tetrahedral Meshes. In *Proceedings of Visualization 1998 (October 1998)*, pages 287-296. 1998.

[4] Trotts, Isaac J., Bernd Hamann, and Kenneth I. Joy. Simplification of Tetrahedral Meshes with Error Bounds. *IEEE Transactions on Visualization and Computer Graphics* 5 (3), pages 224- 237. 1999.

[5] Chopra, P., and J. Meyer. TetFusion: An Algorithm for Rapid Tetrahedral Mesh Simplification. In *Proceedings of IEEE Visualization 2002*, held in Boston, MA, pages 133-140. 2002.

[6] Chopra, P., J. Meyer, and Michael L. Stokes. Immersive Visualization of A Very Large Scale Seismic Model. In *Sketches and Applications of SIGGRAPH 2001 (Los Angeles, California, August 2001)*, page 107. ACM SIGGRAPH, ACM Press. August 2001.

[7] Meyer, J., and Prashant Chopra. Building Shaker: Earthquake Simulation in a CAVE^{TM}. *Work in Progress, IEEE Visualization 2001 (San Diego, California, October 2001),* Abstract, page 3. 2001.

[8] Meyer, J., and Prashant Chopra. Strategies for Rendering Large-Scale Tetrahedral Meshes for Earthquake Simulation. *SIAM/GD 2001 (Sacramento, CA, November 2001)*, Abstract, page 30. 2001.

[9] Garland, M., and P. Heckbert. Surface Simplification Using Quadric Error Metrics. In *Proceedings of SIGGRAPH 1997,* pages 115-122. ACM SIGGRAPH, ACM Press. 1997.

[10] Guskov, I., Kiril Vidimce, Wim Sweldens, and Peter Schroeder. Normal Meshes. In *Proceedings of SIGGRAPH 2000 (New Orleans, Louisiana, July 2000),* pages 95-102. ACM SIGGRAPH, ACM Press. July 2000.

[11] Hoppe, Hugues. Progressive Meshes. In *Proceedings of SIGGRAPH 1996 (New Orleans, Louisiana, August 1996),* pages 99-108. ACM SIGGRAPH, ACM Press. August 1996.

[12] Kalvin, Alan D., and Russell H. Taylor. Superfaces: Polygonal Mesh Simplification with Bounded Error. In *IEEE Computer Graphics and Applications* 16 (3), pages 64-77. 1996.

[13] Popovic, J., and H., Hoppe. Progressive Simplicial Complexes. In *Proceedings of SIGGRAPH 1997,* pages 217-224. ACM SIGGRAPH, ACM Press. 1997.

[14] Renze, K. J., and J. H. Oliver. Generalized Unstructured Decimation. *IEEE Computer Graphics and Applications* 16 (6), pages 24-32. 1996.

[15] Schroeder, William J., Jonathan A. Zarge, and William E. Lorensen. Decimation of Triangle Meshes. *Computer Graphics* 26(2), pages 65-70. 1992.

[16] Schroeder, William J. A Topology Modifying Progressive Decimation Algorithm. In *Proceedings of IEEE Visualization 1997,* pages 205-212. 1997.

[17] Turk, Greg. Re-tiling Polygonal Surfaces. *Computer Graphics,* 26(2), pages 55-64. 1992.

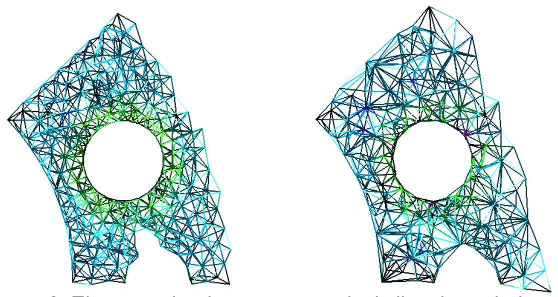


Figure 6: Elements that intersect a vertical slice through the volumes of three levels of detail of the *spx* dataset. The dataset gets reduced to 63.93% of the original number of tetrahedra (12,936) upon decimation.

[18] Garland, M. Multi-resolution modeling: Survey & Future Opportunities. In *EUROGRAPHICS 1999, State of the Art Report (STAR) (Aire-la-Ville, CH, 1999),* pages 111-131. Eurographics Association. 1999.

[19] Edelsbrunner, H. *Geometry and Topology for Mesh Generation*. Cambridge University Press (2001). ISBN 0-521-79309-2. 2001.

[20] Dey, T. K., H., Edelsbrunner, S., Guha, and D. V. Nekhayev. *Topology Preserving Edge Contraction*. Publications de l'Institut Mathematique (Beograd), Vol. 60 (80), pages 23-45. 1999.

[22] Alliez, Pierre, and Mathieu Desbrun. Progressive Compression For Lossless Transmission Of Triangle Meshes. In *Proceedings of SIGGRAPH'01 (Los Angeles, California, August 2001),* Computer Graphics Proceedings, Annual Conference Series, pages 198-205. ACM SIGGRAPH, ACM Press. August 2001.

[23] Gumhold, Stefan, Stefan Guthe, and Wolfgang Straßer. Tetrahedral Mesh Compression With The Cut-Border Machine. In *Proceedings of IEEE Visualization 1999 (San Francisco, California, October 1999),* pages 51-59. IEEE Computer Society Technical Committee on Computer Graphics, IEEE Computer Society. 1999.

[24] Isenburg, Martin, and Jack Snoeyink. Face Fixer: Compressing Polygon Meshes With Properties. In *Proceedings of SIGGRAPH 2000 (New Orleans, Louisiana, July 23-28, 2000),* pages 263-270. ACM SIGGRAPH, ACM Press. 2000.

[25] Pajarola, Renato, Jarek Rossignac, and Andrez Szymczak. Implant Sprays: Compression of Progressive Tetrahedral Mesh Connectivity. In *Proceedings of IEEE Visualization 1999 (San Francisco, California, October 1999),* pages 299-305. IEEE Computer Society Technical Committee on Computer Graphics, IEEE Computer Society. 1999.

[26] Szymczak, Andrzej, and Jarek Rossignac. Grow fold: Compression of Tetrahedral Meshes. In *Proceedings of the Fifth Symposium on Solid Modeling andAapplications (Ann Arbor, Michigan, June 1999)*, pages 54-64. ACM, ACM Press. 1999.

[27] DeCoro, C., and Renato Pajarola. XFastMesh: Fast View-dependent Meshing from External Memory. In *Proceedings of IEEE Visualization 2002*, pages 363-370. 2002.