Title
Simplification of three-dimensional density maps

Journal
IEEE Transactions on Visualization and Computer Graphics, 10

Authors
Natarajan, Vijay
Edelsbrunner, Herbert

Publication Date
2004

Peer reviewed

Simplification of Three-dimensional Density Maps

Vijay Natarajan and Herbert Edelsbrunner

***Abstract*— We consider scientific datasets that describe density functions over three-dimensional geometric domains. Such datasets are often large and coarsened representations are needed for visualization and analysis. Assuming a tetrahedral mesh representation, we construct such representations with a simplification algorithm that combines three goals: the approximation of the function, the preservation of the mesh topology, and the improvement of the mesh quality. The third goal is achieved with a novel extension of the well known quadric error metric. We perform a number of computational experiments to understand the effect of mesh quality improvement on the density map approximation. In addition, we study the effect of geometric simplification on the topological features of the function by monitoring its critical points.**

***Index Terms*— Computational Geometry, Volume Visualization, Hierarchy and Geometric Transformation.**

I. Introduction

As scientific datasets are growing larger in size, it gets more difficult to analyze and visualize them. A popular approach to overcome this difficulty is reducing the size of the data using geometric techniques and work with the resulting coarser representation. We consider the scenario where the data is represented as a tetrahedral mesh that describes a continuous real function by its values at the vertices.

Related prior work. Many of the techniques used for tetrahedral mesh simplification are extensions of those used to simplify triangle meshes. Surface mesh simplification has been studied extensively within the computer graphics and visualization communities. Hoppe [15] pioneered work in this field by introducing progressive meshes generated by repeated edge contraction [1]. Other approaches to surface simplification include vertex clustering [20], vertex removal [21] and triangle contraction [13]. Garland and Heckbert [10] describe a quadric error metric to determine the order of edge contractions and extend this error measure to surfaces with attributes [11]. Hoppe [16] proposes a different extension of the quadric error metric for surface with attributes. Lindstrom and Turk [17] generate simplified models using memory efficient edge contractions to handle large polygonal models. Heckbert and Garland [14] and Cignoni et al. [3] compare different surface simplification algorithms.

Trotts et al. [23] extend the triangle contraction operation described by Gieng et al. [13] to a tetrahedron contraction operation and construct multiple levels of tetrahedral meshes approximating a density function. Staadt and Gross [22] describe a robust implementation of edge contractions in tetrahedral meshes. They also address the definition of appropriate cost functions for specific applications, like the finite element method. Van Gelder et al. [24] compare a mass-based and a density-based metric for use in rapidly decimating a tetrahedral mesh. Cignoni et al. [4] compare various cost functions used to prioritize edge contractions.

Our results. We simplify a tetrahedral mesh representing a three-dimensional density function by a sequence of edge contractions. We modify the quadric error measure described by Garland and Heckbert [10] to combine three goals in prioritizing the contractions:

- the accurate approximation of the density function;
- the faithful preservation of global topological type of the mesh;
- the improvement of the mesh quality defined in terms of angles.

To pursue the first goal, we extend well established ideas from $\mathbb{R}^2$ to $\mathbb{R}^3$. The second goal needs no new results and is based on applying tests described in [5]. To pursue the third goal, we develop a new idea, namely the addition of particular hyperplanes to the quadrics that have a positive influence on the mesh quality. We perform various experiments which show that a small relative weight of the second goal furnishes dramatic improvements in mesh quality. Further increasing that weight adversely affects the approximation of the density function. Lindstrom and Turk [17] attempt to improve

Research by the two authors is partially supported by NSF under grants EIA-99-72879 and CCR-00-86013.

Vijay Natarajan is with the Department of Computer Science, Duke University, Durham, North Carolina.

Herbert Edelsbrunner is with the Departments of Computer Science and Mathematics, Duke University, Durham, North Carolina and Raindrop Geomagic, Research Triangle Park, North Carolina.

[1] We prefer "edge contraction" over the term "edge collapse" used in [15] because the latter collides with the standard use of collapses in combinatorial topology, where it is a combinatorial deformation retraction.

the mesh quality by introducing an additional constraint while determining the location of the vertex that replaces an edge upon its contraction. The effect of their method is limited because the new constraint applies only if the other constraints lead to an ambiguous solution, which happens typically at the mesh boundary. In contrast, our method influences the placement of every vertex and also affects the sequence in which the edges are contracted. By adding extra hyperplanes into the quadric error metric, we pro-actively influence the shape of the tetrahedra created after an edge contraction. This is stronger than previous techniques, like that described in [4], which merely do not perform an edge contraction if it results in badly shaped tetrahedra. We observe an interesting side-effect: a small but non-zero weight on the mesh quality improvement results in better approximations of the density map.

Another surprising finding concerns the relationship between geometric and topological simplification, the latter aiming at preserving the critical point structure of the function. While geometric simplification preserves the overall shape of the function, it sometimes introduces a large number of spurious critical points that confuse the topological picture. In other words, geometric simplification is compatible with but not a substitute for topological simplification.

Outline. We introduce definitions in Section II and discuss the recognition of topology preserving edge contraction in Section III. We then describe the quadric error measure and the choice of hyperplanes that drives the simplification algorithm in Section IV. We explain details of the implementation in Section V, give the results of our various experiments in Section VI, and conclude the paper in Section VII.

II. Definitions

Most of the definitions we need to describe our algorithm are standard and can be found in algebraic topology textbooks such as Munkres [19]. We discuss simplicial complexes, manifolds, triangulations, and edge contractions.

Simplicial complexes. A *k-simplex* σ is the convex hull of $k+1$ affinely independent points. A *face* τ of a simplex σ is defined by a nonempty subset of the $k+1$ points. We write $\tau \leq \sigma$ and call σ a *coface* of τ. We create new simplices by adding vertices to old ones: the *cone* from a vertex x to a k-simplex σ is the convex hull of x and σ, which is the $(k+1)$-simplex $x\sigma$. The operation is defined only if x is not an affine combination of the vertices of σ. A *simplicial complex* K is a finite collection of non-empty simplices for which $\sigma \in K$ and $\tau \leq \sigma$ implies $\tau \in K$ and $\sigma_1, \sigma_2 \in K$ implies that the intersection $\sigma_1 \cap \sigma_2$ is either empty or a face of both, σ_1 and σ_2. In this paper we deal exclusively with simplicial complexes that consist of 3-simplices (tetrahedra) and their faces. The *underlying space* of K is the union of simplices: $|K| = \bigcup_{\sigma \in K} \sigma$.

The *closure* of a subset $L \subseteq K$ is the smallest subcomplex that contains L, the *star* of L is the set of cofaces of simplices in L, and the *link* of L is the set of all faces of simplices in the star that are disjoint from simplices in L:

$$\begin{aligned} \overline{L} &= \{\tau \in K \mid \tau \leq \sigma \in L\}, \\ \operatorname{St} L &= \{\sigma \in K \mid \sigma \geq \tau \in L\}, \\ \operatorname{Lk} L &= \overline{\operatorname{St} L} - \operatorname{St} \overline{L}. \end{aligned}$$

Note that stars describe the neighborhood of L. For example, the star of a vertex v in a simplicial complex in $\mathbb{R}^3$ consists of v together with all edges, triangles and tetrahedra that contain v. The link of v consists of all triangles, edges and vertices that are faces of the simplices in the star and disjoint from v.

Manifolds and triangulations. Two topological spaces X and Y are *homeomorphic* or have the same *topological type* if there is a homeomorphism $X \to Y$. A topological space M is a 3-*manifold* if every point $x \in M$ has an open neighborhood homeomorphic to $\mathbb{R}^3$. A topological space N is a 3-*manifold with boundary* if every point $x \in N$ has an open neighborhood homeomorphic to $\mathbb{R}^3$ or to the *closed halfspace*, $\mathbb{H}^3 = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid x_1 \geq 0\}$. The boundary, $\operatorname{Bd} N$, consists of the points whose neighborhood is homeomorphic to $\mathbb{H}^3$. The boundary of a 3-manifold with boundary is always a 2-manifold without boundary.

A *triangulation* of a topological space X is a simplicial complex K whose underlying space is homeomorphic to X. We can determine if a given simplicial complex is a triangulation of a 3-manifold by checking if the link at every vertex is homeomorphic to the 2-dimensional sphere $\mathbb{S}^2$. The input to our simplification algorithm is a triangulation K of a 3-manifold with boundary. We extend K to the triangulation of a 3-manifold without boundary by connecting a dummy vertex to each boundary component. If $\operatorname{Bd} K$ is connected we just add one dummy vertex, ω, and we denote the extended triangulation by $\mathrm{K}_\omega = K \cup \omega \operatorname{Bd} K$. The link of a simplex $\sigma \in \mathrm{K}_\omega$ is denoted by $\operatorname{Lk}_\omega \sigma$. For a simplex $\sigma \in \operatorname{Bd} K$, the link of σ within the boundary is denoted by $\operatorname{Lk}_{\operatorname{Bd}} \sigma$. In a 3-manifold without boundary the stars and the links are particularly simple: the link of a vertex is a sphere, that of an edge is a circle, and that of a

triangle is a pair of vertices. Similarly in a 2-manifold, the link of a vertex is a circle and that of an edge is a pair of vertices.

Edge contraction. The basic operation in our algorithm contracts an edge to a vertex. Upon contraction, we replace the edge ab with the vertex c. This changes the triangulation only in the neighborhood of a and b. In particular, the cofaces of a and b are deleted and simplices connecting c to the boundary of the created void are added. Formally, these simplices are the cones from c to the simplices in the link of the set of simplices $L = \{ab, a, b\}$. Note that the link of $\overline{ab}$ is also a sphere consisting of simplices in the boundary of the above mentioned void. Fig. 1 illustrates an edge contraction but shows only a subset of the simplices that are removed and added.

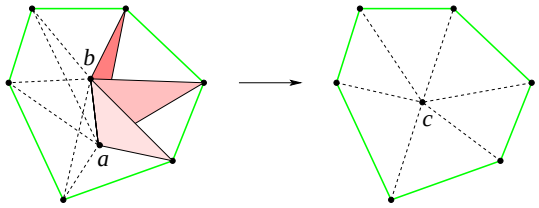

Fig. 1: Edge contraction: the edge ab is contracted to the vertex c. Only the changes in the star of the edge ab are shown.

III. Preserving Domain Topology

Our simplification algorithm performs a sequence of edge contractions on the tetrahedral mesh. Each edge contraction preserves the topological type of the mesh. The algorithm recognizes the edges that can be contracted without changing the topological type by looking at their neighborhoods. The ability to make this judgment based on local computations is crucial for the efficiency of our algorithm.

General link conditions. A 3-*complex* is a simplicial complex consisting of tetrahedra, triangles, edges and vertices. Dey et al. [5] derive local criteria, called link conditions, for recognizing when an edge contraction in a 3-complex preserves the topological type. The link conditions compare the link of the edge ab that is to be contracted with the links of its endpoints. Fig. 2 shows a situation in a 3-manifold where a contraction would change the topology. In the case of a 3-manifold with boundary, N, Dey et al. [5] show that the contraction of an edge ab preserves the topological type if the intersection of the links of the two vertices equals the link of the edge, and this is true both in the extended

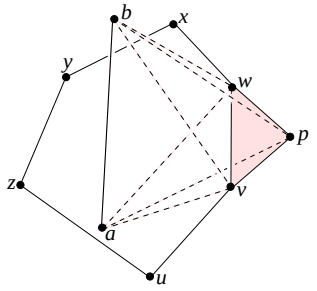

Fig. 2: Triangle vwp lies in the link of both a and b. After contracting ab to a new vertex c, the triangle vwp belongs to only one tetrahedron, namely $vwpc$. The neighborhood of a point in vwp is thus no longer homeomorphic to $\mathbb{R}^3$. The figure shows the ring of edges in $\operatorname{Lk} ab$, namely $uvwxyz$.

3-complex and in the boundary of the 3-complex:

$$\operatorname{Lk}_\omega a \cap \operatorname{Lk}_\omega b = \operatorname{Lk}_\omega ab; \quad \text{and} \tag{1}$$

$$\operatorname{Lk}_{\mathrm{Bd}} a \cap \operatorname{Lk}_{\mathrm{Bd}} b = \operatorname{Lk}_{\mathrm{Bd}} ab. \tag{2}$$

Specialized link conditions. In order to implement the above conditions, we would have to consider the cofaces of ω as special cases because these are not explicitly stored. To simplify the implementation, we eliminate ω from the condition. We have three cases, depending on whether $ab, a,$ and b belong to the boundary or the interior.

`Case` 1: $a, b \in \operatorname{Bd} N$ and $ab \notin \operatorname{Bd} N$. The contraction of ab would change the topological type of N by pinching. It is therefore prohibited.

`Case` 2: At least one of a or $b \notin \operatorname{Bd} N$. Without loss of generality, assume that b is not on the boundary. Since b and hence ab are not on the boundary, $\operatorname{Lk}_{\mathrm{Bd}} b$ and $\operatorname{Lk}_{\mathrm{Bd}} ab$ are not defined and Condition (2) does not apply. The only vertices and edges whose links contain ω or any of its cofaces are the ones on the boundary. This implies $\operatorname{Lk}_\omega b = \operatorname{Lk} b$ and $\operatorname{Lk}_\omega ab = \operatorname{Lk} ab$. Also, $\operatorname{Lk}_\omega a \cap \operatorname{Lk} b = \operatorname{Lk} a \cap \operatorname{Lk} b$ because $\operatorname{Lk} b$ does not contain ω. Condition (1) now simplifies to
$\operatorname{Lk} a \cap \operatorname{Lk} b = \operatorname{Lk} ab$.

`Case` 3: $ab \in \operatorname{Bd} N$. We necessarily also have a and b on the boundary. We partition $\operatorname{Lk}_\omega a$ into $\operatorname{Lk} a$ and the set $\omega\operatorname{Lk}_{\mathrm{Bd}} a$ that contains the simplices that are cofaces of ω. Similarly, we partition $\operatorname{Lk}_\omega b$ and $\operatorname{Lk}_\omega ab$ and obtain

$$(\operatorname{Lk} a \cap \operatorname{Lk} b) \,\dot\cup\, (\omega\operatorname{Lk}_{\mathrm{Bd}} a \cap \omega\operatorname{Lk}_{\mathrm{Bd}} b) = \operatorname{Lk} ab \,\dot\cup\, \omega\operatorname{Lk}_{\mathrm{Bd}} ab,$$

which is equivalent to Condition (1). Three of the terms contain no simplex in the star of ω and the other three contain only simplices in the star of ω. We can therefore express the condition as a conjunction of two conditions. We further simplify by removing ω from the second set

of three terms and get

$$\begin{aligned} \operatorname{Lk} a \ \cap \ \operatorname{Lk} b &= \operatorname{Lk} ab; \quad \text{and} \\ \operatorname{Lk}_{\operatorname{Bd}} a \ \cap \ \operatorname{Lk}_{\operatorname{Bd}} b &= \operatorname{Lk}_{\operatorname{Bd}} ab. \end{aligned}$$

We summarize the results of the case analysis by stating a modified link condition for a 3-manifold with boundary.

LEMMA 1 If N is a 3-manifold with boundary then the contraction of an edge $ab \in N$ preserves the topological type if one of the following is true:

1) at least one of a and b does not belong to $\operatorname{Bd} N$ and $\operatorname{Lk} a \cap \operatorname{Lk} b = \operatorname{Lk} ab$;
2) ab belongs to $\operatorname{Bd} N$, $\operatorname{Lk} a \cap \operatorname{Lk} b = \operatorname{Lk} ab$, and $\operatorname{Lk}_{\operatorname{Bd}} a \cap \operatorname{Lk}_{\operatorname{Bd}} b = \operatorname{Lk}_{\operatorname{Bd}} ab$.

Lemma 1 applies to our data, which in all cases consists of a tetrahedral mesh of a cube in $\mathbb{R}^3$. In Section V, we describe the procedure that checks the link conditions and explain how to make it more efficient than the straight implementation of the formulas.

IV. PRIORITIZING CONTRACTIONS

We use a cost associated with each edge to determine the order of contractions. A vertex is a point in $\mathbb{R}^4$, with three spatial coordinates and the fourth giving the function value. Each vertex and edge is associated with a finite set of hyperplanes in $\mathbb{R}^4$. The cost of an edge is the minimum, over all points of $\mathbb{R}^4$, of the sum of square distances between the point and the hyperplanes associated with the edge and its endpoints. This cost can be computed from the hyperplanes using an extension of the quadric error measure proposed by Garland and Heckbert [10]. Hoppe [16] extends the quadric error metric for surface attributes by performing a projection in $\mathbb{R}^3$ and computing geometric and attribute errors. This approach is particularly efficient when the number of attributes is large unlike our setting with only one attribute. We chose to use the simple and more direct extension of Garland and Heckbert's quadric error metric, namely performing a projection in $\mathbb{R}^4$ for computing the error.

Hyperplanes. The purpose of the hyperplanes associated with a vertex is to locally preserve the density function. We use hyperplanes spanned by tetrahedra of the mesh. Initially, each vertex is associated with the set of hyperplanes spanned by the tetrahedra in its star. When we create a new vertex c by contracting the edge ab we associate the union of the sets of a and b to c. The purpose of the hyperplanes associated with an edge is to locally improve the quality of the mesh. We use perpendicular bisectors of edges. Specifically, the hyperplanes associated with ab are the bisectors of the edges in the link of $L = \{ab, a, b\}$. For each edge in this link, we take the bisecting plane in $\mathbb{R}^3$ and extend it vertically to a hyperplane in $\mathbb{R}^4$. Figure 3 illustrates this idea one dimension lower, where the link of a contractible closed edge is a circle. The rationale

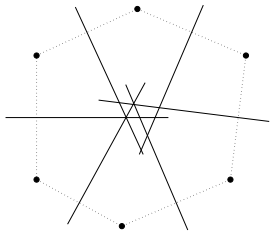

Fig. 3: The dotted link of a closed edge and the solid bisectors of its edges.

for this choice of hyperplanes is to encourage almost spherical links of new vertices. As a consequence, the new tetrahedra are almost isosceles, with three almost equally long edges. Since there are no preferred vertices, we really encourage regular tetrahedra. The contraction of an edge ab causes a change in the link of all vertices in $\operatorname{Lk} a \cup \operatorname{Lk} b$ and thus requires an update in the sets of hyperplanes associated with the edges incident to these vertices.

Fundamental quadric. Let H be a set of hyperplanes and $x = (x_1, x_2, x_3, x_4)^T$ a point in $\mathbb{R}^4$. Let the unit normal of a hyperplane $h \in H$ be $v_h = (v_1, v_2, v_3, v_4)^T$ and the offset $\delta_h = -p^T \cdot v_h$, for any point $p \in h$. The square of the distance between h and x is given by:

$$\begin{aligned} D_h &= ((x-p)^T \cdot v_h)^2 = (x^T \cdot v_h - p^T \cdot v_h)^2 \\ &= (x^T \cdot v_h + \delta_h)^2 = (\mathbf{x}^T \cdot \mathbf{v}_h)^2 \\ &= (\mathbf{x}^T \cdot \mathbf{v}_h)(\mathbf{v}_h^T \cdot \mathbf{x}) = \mathbf{x}^T (\mathbf{v}_h \cdot \mathbf{v}_h^T) \mathbf{x}, \end{aligned}$$

where $\mathbf{x} = (x_1, x_2, x_3, x_4, 1)^T$ and $\mathbf{v}_h = (v_1, v_2, v_3, v_4, \delta_h)^T$. The sum over all $h \in H$ is

$$D = \sum_{h \in H} D_h = \mathbf{x}^T \left(\sum_{h \in H} \mathbf{v}_h \cdot \mathbf{v}_h^T \right) \mathbf{x}.$$

The 5-by-5 matrix $\mathbf{Q} = \sum_{h \in H} \mathbf{v}_h \cdot \mathbf{v}_h^T$ is symmetric and positive semi-definite and is called the *fundamental quadric* of H. Instead of storing sets of hyperplanes, we store their fundamental quadrics. This representation requires two types of updates whenever we contract an edge ab. First the quadric of the new vertex c is computed as the sum of the quadrics of a and b. This new quadric really represents a multi-set of hyperplanes because a hyperplane associated with both endpoints is now counted twice. The difference to the quadric

of the set (without double-counting) is however small since a single hyperplane cannot be counted more than four times. Second, the contraction changes all edges that have a or b as endpoint. We update the quadrics stored at edges associated with the changed bisectors by subtracting the contributions of the old and adding the contributions of the new bisectors.

Weighted hyperplanes. Although the edge contraction operation preserves the continuity of the model, it does not handle the boundary very well. Following the solution proposed by Garland and Heckbert [10], we rectify this by adding boundary constraints. For each boundary triangle, we include the hyperplane passing through the triangle and perpendicular to the hyperplane spanned by the tetrahedron that contains the triangle as a face. Further, these new hyperplanes are weighted with a large penalty value preventing vertices from moving too far from the boundary. Weighted hyperplanes can be easily incorporated into the current setting. The square distance between a vertex x and a hyperplane h with weight w_h is now $D_h = w_h \mathbf{x}^T(\mathbf{v}_h \cdot \mathbf{v}_h^T)\mathbf{x}$. The sum of square distances to all hyperplanes can be derived as before. We compute new quadrics by adding old ones, same as before. The only change is in the step where we compute the initial quadrics.

Another place where we use weights is in controlling the influence of the mesh quality improving hyperplanes on the simplification process. Each such hyperplane is weighed by a constant $\varphi \geq 0$, which we refer to as the *mesh quality factor*. For example, $\varphi = 0$ corresponds to no influence from these hyperplanes and $\varphi = 1$ corresponds to equal influence of both types of hyperplanes.

Optimal vertex placement. The cost of an edge ab depends also on the location of the new vertex. In the generic case, there is a unique location $c \in \mathbb{R}^4$ that minimizes the error. This minimum is given by setting the partial derivatives to zero, for $i = 1, 2, 3, 4$:

$$\begin{aligned} \frac{\partial D(x)}{\partial x_i} &= \frac{\partial \mathbf{x}^T}{\partial x_i} \cdot \mathbf{Q} \cdot \mathbf{x} + \mathbf{x}^T \cdot \mathbf{Q} \cdot \frac{\partial \mathbf{x}}{\partial x_i} \\ &= \mathbf{q}_i^T \cdot \mathbf{x} + \mathbf{x}^T \cdot \mathbf{q}_i \\ &= 0, \end{aligned}$$

where $\mathbf{q}_i^T$ is the i-th row and $\mathbf{q}_i$ the i-th column of $\mathbf{Q}$. The solution is given by $c = -Q^{-1} \cdot q$, where Q is the upper left 4-by-4 submatrix of $\mathbf{Q}$ and q is the 4-vector consisting of the upper four entries in the fifth column of $\mathbf{Q}$.

In the non-degenerate case, Q has rank four. We detect degeneracies by estimating the rank of Q before computing c. Ranks three, two and one correspond to a line, plane and hyperplane of minima in $\mathbb{R}^4$, respectively. Note that Q has at least one non-zero diagonal element and hence the rank is never zero. In each of the three degenerate cases, we add the contributions of additional hyperplanes to increase the rank of the matrix. We discuss how we estimate the rank and omit the discussion of how we find appropriate hyperplanes that increase the rank. We estimate the rank by comparing the coefficients of the characteristic polynomial given by

$$\det(Q - \lambda I) = \sigma_4 - \lambda\sigma_3 + \lambda^2\sigma_2 - \lambda^3\sigma_1 + \lambda^4.$$

Here, σ_4 is the determinant and σ_1 is the trace of Q. The other two coefficients are sums of 3-by-3 and 2-by-2 minors of Q. We note that the coefficients are cheaper to compute than the eigenvalues and may be substituted for the latter in estimating the rank of the matrix. Specifically, we consider Q to have rank three, two and one, respectively, if the absolute value of $256\sigma_4/\sigma_1^4$, $16\sigma_3/\sigma_1^3$ and $8\sigma_2/\sigma_1^2$ is small.

Let ab be the edge being contracted. In the case of a rank three matrix, we add the contribution of the hyperplane normal to the line of minima that passes through the midpoint of ab to the fundamental quadric, hoping that the rank increases to four. In the case of a rank two matrix, we have a plane of minima. We compute two hyperplanes that are orthogonal to each other and the plane and that pass through the midpoints of ab and add their contributions to the quadric. We compute the first hyperplane by taking the cross product of the two independent rows, and the second by taking the wedge product of the now three independent rows. Similarly, in the case of a rank one matrix, we get three new hyperplanes and add their contributions to the quadric. In each of the above cases, we progressively increase the weight of these hyperplanes until the rank increases to four. If the rank does not improve after several iterations, then we select the midpoint of ab as the new vertex location.

V. Implementation

In this section, we describe our implementation of the edge contraction operation. Various choices were made in determining the order of contractions, recognizing topology preserving contractions, and updating all data structures. We begin with the data structures and the outline of the simplification procedure.

Data structure and algorithm. We use a heap to implement the priority queue for the edges of the mesh. Along with the edges, we store the cost of contraction and the optimal vertex location. We store the mesh in a triangle-edge data structure [18], which is a version of

the more general edge-facet data structure by Dobkin and Laszlo [6]. It is made up of triangles, each represented by the six possible orderings of its three vertices. As illustrated in Fig. 4, each ordering maintains a pointer to the next triangle reached by a rotation about the edge of its first two vertices. The list of triangles is stored in an array. The coordinates of the vertices are stored in another array, and the indices of the vertices in this array are used as vertex names.

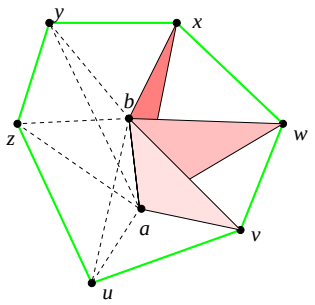

Fig. 4: Each ordering abv stores a pointer to the next triangle: abv.next $= abw$. By following these pointers, we traverse the ring of triangles in the star of ab. After contracting ab to c, the ring of triangles becomes a ring of edges around c.

Function SIMPLIFY performs a sequence of edge contractions to simplify the tetrahedral mesh K. It continues until the mesh reaches a user-specified number of at most v_0 vertices or no edge can be contracted without changing the topological type, whichever occurs first.

```
Mesh SIMPLIFY (Mesh K)
    initialize priority queue PQ with set of edges in K;
    while #vertices in K exceeds v_0 do
        pop the minimum cost edge in PQ and call it ab;
        if PRESTOP (ab) then
            delete edges in St a and St b from PQ;
            update costs of edges in St x, x ∈ Lk a ∪ Lk b;
            K = CONTRACT (K, ab, c);
            insert edges in St c into PQ;
        endif
    endwhile;
    return K.
```

The remainder of this section explains the two main functions used in this algorithm.

Preserving domain topology. Function PRESTOP uses the two conditions in Lemma 1 to determine whether or not the contraction of ab preserves the topological type of K. We prevent redundant tests by first checking which part of the edge is on the boundary and then test zero, one or two conditions.

```
boolean PRESTOP (Edge ab)
  if a, b ∈ Bd K and ab ∉ Bd K then
    return FALSE
  endif;
  if ab ∉ Bd K then return LINKCOND1 (ab) endif;
  if ab ∈ Bd K then
    return (LINKCOND1 (ab) and LINKCOND2 (ab))
  endif.
```

Functions LINKCOND1 implementing Condition (1) and LINKCOND2 implementing Condition (2) use enumerations of the simplices in the link of a vertex or edge. In Function LINKCOND2, we also need the restrictions of these links to the boundary of K, and to facilitate their computation, we label each triangle in $\mathrm{Bd}\,K$. Each link is computed by a local search procedure that starts at an ordered triangle provided by the vertex or edge for which we compute the link. Next, we describe the implementation of Function LINKCOND1, which determines whether or not the intersection of the links of a and b contains simplices that do not belong to the link of ab. We use a marking mechanism to keep track of the processed vertices.

```
boolean LINKCOND1 (Edge ab)
  foreach v ∈ Lk a do MARK (v) endfor;
  foreach v ∈ Lk ab do UNMARK (v) endfor;
  foreach v ∈ Lk b do
    if ISMARKED (v) then return FALSE endif
  endfor;
  return TRUE.
```

After testing the three links, we unmark all vertices again. We repeat the same test for edges in the intersection of the links restricting it to edges that connect two vertices in $\mathrm{Lk}\,ab$. It is not necessary to test triangles. Condition (2) is tested in a similar manner by Function LINKCOND2, which traverses the links of a, b and ab on the boundary of the mesh.

Contracting edges. Function CONTRACT updates the mesh K by contracting an edge ab as follows:

```
Mesh CONTRACT (Mesh K, Edge ab, Vertex c)
  foreach triangle axy ∈ St a do
    if x, y ≠ b then add cxy to K endif
  endfor;
  foreach triangle bxy ∈ St b do
    if x, y ≠ a and cxy ∉ K then add cxy to K endif
  endfor;
  delete all triangles in St a and St b from K;
  return K.
```

Note that the contraction of ab may change the status of other edges in the mesh. We are interested in edges

xy that violate the conditions of Lemma 1 before the contraction of ab and that satisfy these conditions after the contraction of ab. We say these edges *turn contractible*. We detect the edges that have the potential for turning contractible and add them to the priority queue, using labels to avoid duplicate entries. We prove below that only edges in a relatively small subset of the link of c can turn contractible. This result is essential for an efficient detection of these edges.

LEMMA 2 If the contraction of ab causes another edge xy to turn contractible then the end points of xy, x and y, are contained in $\mathrm{Lk}\, ab$.

PROOF. For every edge xy, we have $\mathrm{Lk}\, xy \subseteq \mathrm{Lk}\, x \cap \mathrm{Lk}\, y$ with the equality holding if xy does not violate the conditions in Lemma 1. Suppose $\mathrm{Lk}\, x \cap \mathrm{Lk}\, y$ consists of the cycle $\mathrm{Lk}\, xy$ plus some additional simplices. The only way the contraction of ab can cause xy to turn contractible is by removing these extra simplices. Now, for $\mathrm{Lk}\, x \cap \mathrm{Lk}\, y$ to shrink, we need a and b in both links. Since $a \in \mathrm{Lk}\, x$ iff $x \in \mathrm{Lk}\, a$, this is equivalent to $x, y \in \mathrm{Lk}\, a \cap \mathrm{Lk}\, b$. Lemma 2 follows because ab satisfies both link conditions, particularly $\mathrm{Lk}\, a \cap \mathrm{Lk}\, b = \mathrm{Lk}\, ab$. □

VI. EXPERIMENTS AND RESULTS

There are two parameters that affect the performance of our algorithm: the target vertex count, v_0, and the relative weight of the hyperplanes that were added into the quadric to improve the mesh quality, φ. We perform various experiments in order to determine an optimum value for φ. We evaluate the results by computing approximation errors, visualizing the simplified mesh through isosurfaces and looking at critical point statistics. We compute the approximation error of a simplified mesh as the root mean square and maximum of the error at each vertex of both the simplified mesh as well as the original mesh. The error at a vertex is the difference between its density values in the original and simplified meshes. We apply our simplification algorithm to four datasets. Table I lists the size of the datasets. Isosurfaces of the original and several simplified versions of the data can be seen in Figures 10 to 13. The first dataset contains cryo electron microscopy data of a ribosome. The second is an MRI scan data from the Chapel Hill volume rendering test data set, volume I. The third is density data of a turbine blade. The fourth is an electron density dataset of a hydrogen molecule. Each of the datasets is available to us for input as a tetrahedral mesh with function values specified at the mesh vertices and linearly interpolated within the mesh elements. We eliminate the effects of large scale differences between the spatial coordinates and the function values by normalizing the data within the unit hypercube in $\mathbb{R}^4$.

dataset	#vertices	#tetrahedra
`ribosome`	512,000	2,958,234
`head`	70,262	391,608
`turbine`	126,976	714,420
`hydrogen`	32,768	178,746

TABLE I: List of datasets used for evaluation.

Tetrahedral shape improvement. The objective of the first experiment is to determine the effect of varying φ on the mesh quality. We do this by applying the simplification using various values of φ and computing the dihedral, solid, and face angles of the tetrahedra in the simplified meshes. The average values of the three types of angles remain almost constant at around $1.20, 0.49$, and 1.05 radians. We see in Fig. 5 that the introduction of a positive value for φ sharpens the distribution of angles around their respective averages. Fig. 6 provides visual evidence of the improvement in the tetrahedral shape quality. These results are for experiments run on the `hydrogen` dataset. The other datasets behave similarly.

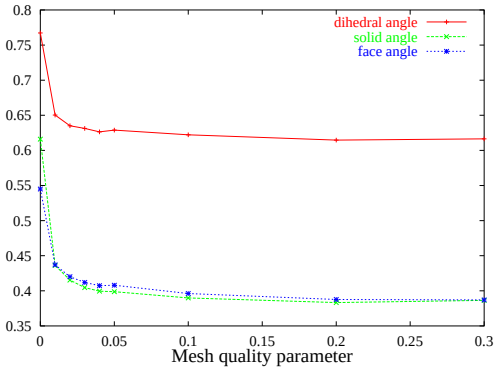

Fig. 5: Graph of the standard deviation for the dihedral, solid and face angle measurements (in radians) for various values of the mesh quality parameter φ. Note the initial dip followed by almost no change.

Approximation error vs tetrahedral shape. As mentioned earlier, we add weighted hyperplanes into the error quadric in the hope of improving the shape of tetrahedra in the mesh. However, adding these hyperplanes reduces the weight on the density map error that should be minimized for a good approximation. The objective of our second experiment is to study the effect of varying φ on the approximation error. Fig. 7 shows the root mean square and max error for various values of φ.

Topology preservation vs tetrahedral shape. In this

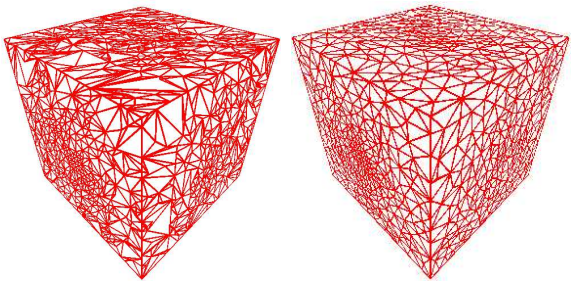

Fig. 6: The simplified meshes of the `hydrogen` dataset obtained for $\varphi = 0.0$ (left) and $\varphi = 0.01$ (right). Note the dramatic improvement in the shape of the elements for a non-zero shape quality factor.

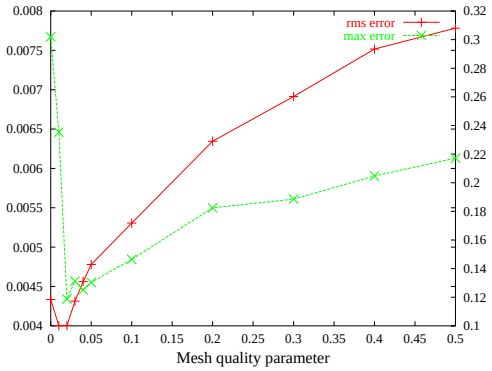

Fig. 7: Graphs of the root mean square error (with values shown on the left) and maximum error (with values shown on the right) of the simplified meshes for various values of the mesh quality parameter φ. Note the initial dip for very small values of φ after which the errors increase monotonically.

experiment, we study the effect of varying φ on the smallest achievable vertex count. Violation of the link condition seems to require badly shaped tetrahedra, so we expect that we can reach smaller sizes if we increase φ. Fig. 8 is a graph of the number of vertices in the smallest mesh reachable by the simplification algorithm for various values of the parameter φ. Note the expected dip followed by an almost horizontal section. This is consistent with the earlier observation of a dramatic improvement in the shape quality of the mesh tetrahedra even with small values of φ followed by no significant change on further increasing φ.

Density map preservation. The results of the above experiments suggests we choose a mesh quality factor in the range where the graphs show significant improvement in the shape quality of mesh tetrahedra. We set $\varphi = 0.02$ and apply the simplification algorithm to the four datasets. Figures 10, 11, 12, and 13 display a small sample of isosurface to provide a feeling for the effect the simplification has on the datasets. Significant artifacts begin to appear when the target vertex count drops to

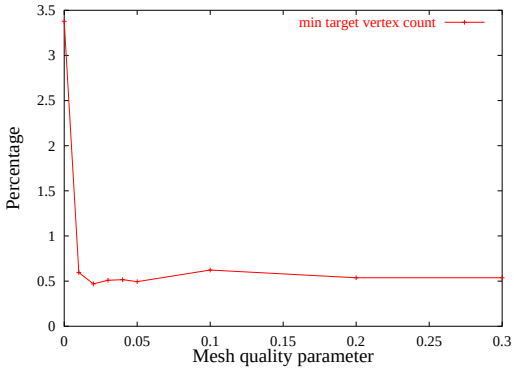

Fig. 8: Graph of the number of vertices in the smallest mesh reachable by the algorithm. Similar to the mesh quality, we observe a dramatic initial improvement followed by almost no change.

10% or below. This is reflected in the root mean square and max errors shown in Table II. The table also show the time taken to perform the simplification.

Critical point statistics. The topography of a density map is often expressed in terms of its critical points, which in the generic case are of one of four types: minima, 1-saddles, 2-saddles and maxima. As defined in [7], the *lower link* of a vertex u is the subcomplex of the link induced by the vertices with smaller function value than u. Using reduced Betti numbers for measuring the connectivity of the lower link, we classify u as regular or critical. In the piecewise linear case, a critical vertex can have non-trivial multiplicity even in the generic case, which is reflected in our statistics shown in Figure 9. Contrary to our initial expectations, the simplification first increases the number of critical points before decreasing them. We explain this phenomenon by the temporary creation of spurious critical points in relatively flat regions of the distribution. The criticality of these vertices is based on small fluctuations of the density function. We substantiate this rationalization by measuring the importance of a critical point as the amount of change in function value necessary to turn it into a regular point. Formally, we compute the persistence of the critical vertices as defined in [9]. In the graphs of Figure 9, we reflect this information by ignoring critical points whose persistence is less than a threshold that increases from back to front. We see that a very small threshold suffices to erode the gain in critical points caused in the initial simplification phase.

Sanity checks. To ensure that the implementation does not have subtle flaws that create biases or other artifacts is always a challenge when working with non-trivial datasets. Typically, one looks for unusual behavior while

`ribosome`

%	#vert	rms	max	time
100	512,000			
50	256,000	0.000	0.016	3,009
30	153,600	0.001	0.223	4,178
20	102,400	0.002	0.684	4,771
10	51,200	0.003	0.392	5,403
5	25,600	0.006	0.140	5,760
3	15,360	0.007	0.205	5,914
2	10,240	0.008	0.218	5,997
1	5,120	0.011	0.310	6,087

`hydrogen`

%	#vert	rms	max	time
100	32,768			
50	16,384	0.001	0.030	135
30	9,830	0.002	0.153	187
20	6,553	0.003	0.099	213
10	3,276	0.005	0.146	242
5	1,638	0.014	0.228	257
3	983	0.017	0.301	264
2	655	0.021	0.302	267
1	327	0.032	0.442	271

`turbine`

%	#vert	rms	max	time
100	126,979			
50	63,488	0.000	0.052	606
30	38,092	0.001	0.088	825
20	25,395	0.004	0.349	938
10	12,698	0.014	0.289	1,058
5	6,348	0.027	0.549	1,126
3	3,809	0.042	0.678	1,155
2	2,539	0.057	0.669	1,170

`head`

%	#vert	rms	max	time
100	70,262			
50	35,131	0.007	0.314	315
30	21,078	0.041	0.784	438
20	14,052	0.028	0.791	504
10	7,026	0.035	0.660	575
5	3,513	0.049	0.965	616
3	2,107	0.069	0.965	633

TABLE II: rms and max errors associated with each of the simplified meshes for the four datasets and the running time in seconds.

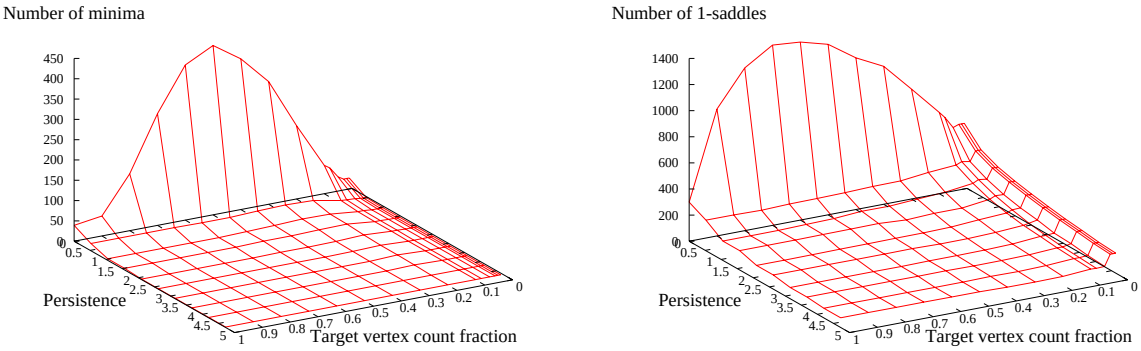

Fig. 9: Graphs of the numbers of minima and 1-saddles in the `hydrogen` data. The numbers change from left to right as we simplify the density and from back to front as we eliminate critical points of low persistence. The graphs for the numbers of 2-saddles and maxima are very similar.

testing the code against special data. In addition, we check the code by collecting evidence that the output is structurally correct. We perform low and high level structural checks of the mesh. At the lowest level, we test whether the triangles in K are connected the right way. Details of such tests can be found in Mücke [18]. At a higher level, we compute the Euler characteristic: $\chi = s_0 - s_1 + s_2 - s_3$, where s_i is the number of i-simplices in K. The Euler characteristic of a 3-ball is 1, and since our algorithm maintains the topological type, it must be 1 throughout the process. There is a relation between the Euler characteristic and the critical points, namely $\chi = c_0 - c_1 + c_2 - c_3$, where c_0, c_1, c_2 and c_3 count the minima, 1-saddles, 2-saddles and maxima, respectively. The compactification that changes the 3-ball into the 3-sphere changes the Euler characteristic to 0. The alternating sum of critical points thus furnishes another test for the correctness of the simplified mesh.

Mesh quality revisited. We conclude this section with a brief comparison between two different implementations of the mesh quality improvement using bisecting hyperplanes. The explained algorithm uses a *memoryless* implementation in which the bisectors at each step are taken for edges in the current mesh. Alternatively, we

could accumulate the bisectors of edges in the original mesh, similar to the way we accumulate hyperplanes spanned by tetrahedra in the original mesh. Initially, each vertex stores the quadric defined by the hyperplanes bisecting the edges in the link. Note that in the case of a vertex, the link is precisely the boundary of the star. Later in the process the vertex represents a collection of vertices in the original mesh and stores the quadric defined by the hyperplanes bisecting edges in the boundary of the union of vertex stars. The link condition guarantees that this union is a topological ball and its boundary is a topological sphere. The quadric of this set of hyperplanes can be computed by accumulation, but there is a complication caused by the need to remove hyperplanes that bisect edges in the interior of the ball. We cope with this complication by inclusion-exclusion:

- initialize a quadric for each simplex of the original mesh defined by the bisectors of the edges in the boundary of the star of the simplex;
- upon contracting the edge ab to the vertex c, compute the quadric as $\mathbf{Q}(c) = \mathbf{Q}(a) + \mathbf{Q}(b) - \mathbf{Q}(ab)$.

Similarly, we use inclusion-exclusion to compute the quadrics of newly formed edges and triangles. We note that inclusion-exclusion is also the preferred way to accumulate the quadrics of shape preserving hyperplanes, except that the simpler method of just adding quadrics commits only the negligible error of double-counting certain hyperplanes. We implemented the alternative method and compared it with the memoryless one, finding possibly predictable differences in performance:

(i) the alternative method is faster than the memoryless method by a factor of about three;
(ii) the approximation error for the alternative method is marginally higher than that of the memoryless method;
(iii) the reachable limit of maximum simplification increases from about 0.5% for the memoryless method to about 3% for the alternative method.

In summary, the quality of the simplification is somewhat worse for the alternative method, but the running time is somewhat better. In this paper, we place more weight on the quality of the computed result than on speed and thus decided to present detailed experimental results only of the memoryless method of mesh quality improvement.

VII. Discussion

We described an algorithm for simplifying a density function represented by a tetrahedral mesh of a three-dimensional geometric domain. The main ingredients of the algorithm are topology preserving edge contractions and quadratic cost functions that attempt to preserve the density map as well as improve the mesh quality. There has been some recent work on simplification of tetrahedral meshes that preserve the topology of the isosurfaces [2] or control the topology simplification of the isosurfaces [12]. In this context, we want to clarify that we aim to preserve the topology of the domain. The decrease in critical point count for the simplified models reflects the topology simplification of the isosurfaces. We performed various computational experiments to determine relationships between the parameters that control the algorithm. We ran our algorithm on four datasets and evaluated the results by computing the approximation error, some isosurfaces, and the number of critical points, all as variables depending on the amount of simplification. We conclude this paper with a short list of future projects suggested by the work in this paper.

(1) Use the hierarchy of critical points to get a comparison between two similar density functions that is more qualitative than the approximation error computed in this paper. Study the behavior of this comparison as the functions become progressively less similar.
(2) Compare the numerical method that maintains the mesh boundary using extra hyperplane constraints with a combinatorial method based on the general link condition. The latter method would preserve the face and edge structure of the mesh boundary and treat boundary vertices with higher priority.

The natural next step in simplifying a density function is a synthesis of geometric and topological methods, similar to the work of Bremer et al. [1] and Edelsbrunner et al. [8] for two-dimensional functions. This amounts to constructing the three-dimensional Morse-Smale complex [7] and simplifying it in a sequence of cancellations ordered by persistence [9]. While doable, the implementation of this method is technically challenging and we are looking for alternatives. For example, is it possible to achieve similar results by re-prioritizing the edge contractions in our current algorithm to include topological information about the critical points?

Acknowledgments

The authors thank the anonymous referees for their valuable comments and suggestions.

References

[1] P.-T. Bremer, H. Edelsbrunner, B. Hamann and V. Pascucci. A multi-resolution data structure for two-dimensional Morse functions. *In* "Proc. IEEE Conf. Visualization, 2003", 139–146.

[2] Y. J. CHIANG AND X. LU. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum* **22** (2003), 493–504.

[3] P. CIGNONI, C. MONTANI AND R. SCOPIGNO. A comparison of mesh simplification algorithms. *Computers and Graphics* **22** (1998), 37–54.

[4] P. CIGNONI, D. CONSTANZA, C. MONTANI, C. ROCCHINI AND R. SCOPIGNO. Simplification of tetrahedral meshes with accurate error evaluation. *In* "Proc. IEEE Conf. Visualization, 2000", 85–92.

[5] T. K. DEY, H. EDELSBRUNNER, S. GUHA AND D. V. NEKHAYEV. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N. S.)* **66** (1999), 23–45.

[6] D. P. DOBKIN AND M. J. LASZLO. Primitives for manipulation of three dimensional subdivisions. *Algorithmica* **4** (1989), 3–32.

[7] H. EDELSBRUNNER, J. HARER, V. NATARAJAN AND V. PASCUCCI. Morse-Smale complexes for piecewise linear 3-manifolds. *In* "Proc. 19th Ann. Sympos. Comput. Geom., 2003", 361–370.

[8] H. EDELSBRUNNER, J. HARER AND A. ZOMORODIAN. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom.* **30** (2003), 87–107.

[9] H. EDELSBRUNNER, D. LETSCHER AND A. ZOMORODIAN. Topological persistence and simplification. *Discrete Comput. Geom.* **28** (2002), 511–533.

[10] M. GARLAND AND P. S. HECKBERT. Surface simplification using quadric error metrics. *In* "Proc. SIGGRAPH, 1997", 209–216.

[11] M. GARLAND AND P. S. HECKBERT. Simplifying surfaces with color and texture using quadric error metrics. *In* "Proc. IEEE Conf. Visualization, 1998", 263–269.

[12] T. GERSTNER AND R. PAJAROLA. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. *In* "Proc. IEEE Visualization, 2000", 259–266.

[13] T. S. GIENG, B. HAMANN, K. I. JOY, G. L. SCHUSSMAN AND I. J. TROTTS. Constructing hierarchies for triangle meshes. *IEEE Trans. Visualization Comput. Graphics* **4** (1998), 145–161.

[14] P. S. HECKBERT AND M. GARLAND. Survey of polygonal surface simplification algorithms. *In* "SIGGRAPH '97 Course Notes, 1997".

[15] H. HOPPE. Progressive meshes. *In* "Proc. SIGGRAPH, 1996", 99–108.

[16] H. HOPPE. New quadric metric for simplifying meshes with appearance attributes. *In* "Proc. IEEE Conf. Visualization, 1999", 59–66.

[17] P. LINDSTROM AND G. TURK. Fast and memory efficient polygonal simplification. *In* "Proc. IEEE Conf. Visualization, 1998", 279–286.

[18] E. P. MÜCKE. *Shapes and Implementations in Three-Dimensional Geometry*. PhD. thesis, Dept. Comput. Sci., Univ. Illinois, Urbana, Illinois, 1993.

[19] J. R. MUNKRES. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, California, 1984.

[20] J. ROSSIGNAC AND P. BORREL. Multi-resolution 3D approximations for rendering complex scenes. *Modeling in Computer Graphics: Methods and Applications*, Springer-Verlag, eds. B. Falcidieno and T. Kunii, 1993, 455–465.

[21] W. J. SCHROEDER, J. A. ZARGE AND W. E. LORENSEN. Decimation of triangle meshes. *In* "Proc. SIGGRAPH **26**, 1992", 65–70.

[22] O. G. STAADT AND M. H. GROSS. Progressive tetrahedralizations. *In* "Proc. IEEE Conf. Visualization, 1998", 297–402.

[23] I. J. TROTTS, B. HAMANN, K. I. JOY AND D. F. WILEY. Simplification of tetrahedral meshes. *In* "Proc. IEEE Conf. Visualization, 1998", 287–295.

[24] A. VAN GELDER, V. VERMA AND J. WILHELMS. Volume decimation of irregular tetrahedral grids. *Computer Graphics International* (1999), 222–230.

Vijay Natarajan is currently a Ph.D. candidate in computer science at Duke University. He received the B.E. degree in computer science and M.Sc. degree in mathematics, both from Birla Institute of Technology and Sciences, Pilani in 1999. His research interests include scientific visualization, computational geometry and topology, geometric modeling and meshing.

Herbert Edelsbrunner is Arts and Sciences Professor of Computer Science and Mathematics at Duke University. He is also Director of Raindrop Geomagic, a 3D geometric processing software company he co-founded in 1996. He holds M.S. and Ph.D. degrees in technical mathematics from the Graz University of Technology in Austria. Prior to his appointment at Duke he was Faculty from 1981 to 85 at the Graz University of Technology and from 1985 to 1999 at the University of Illinois at Urbana-Champaign. He was awarded the Waterman Award from the National Science Foundation in 1991. He is author of two books, 'Algorithms in Combinatorial Geometry' published in 1987 by Springer-Verlag, and 'Geometry and Topology for Mesh Generation' published in 2001 by Cambridge University Press. His research is primarily in algorithms, discrete and computational geometry, computational topology, and structural molecular biology.

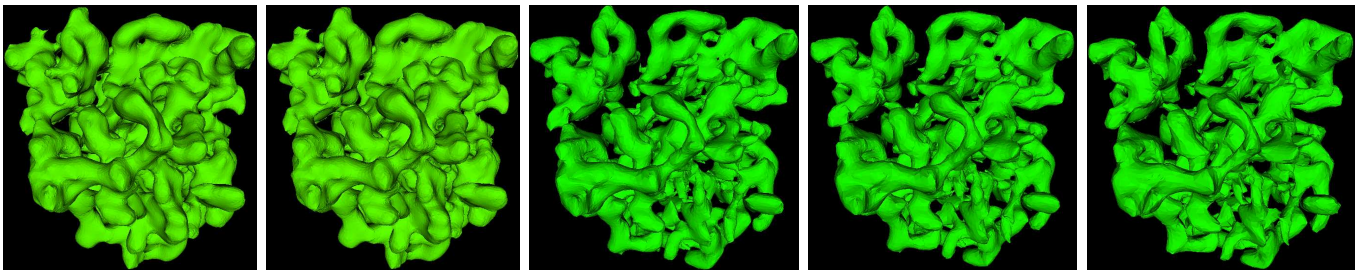

Fig. 10: From left to right: isosurfaces extracted from the original `ribosome` dataset and after simplifying the mesh to 50%, 30%, 20% and 10% its original size.

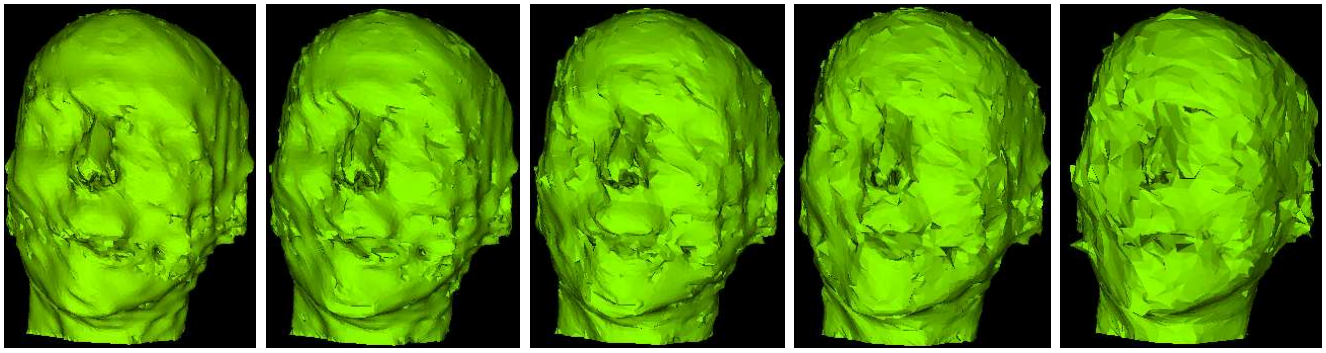

Fig. 11: From left to right: isosurfaces extracted from the original `head` dataset and after simplifying the mesh to 50%, 30%, 20% and 10% its original size.

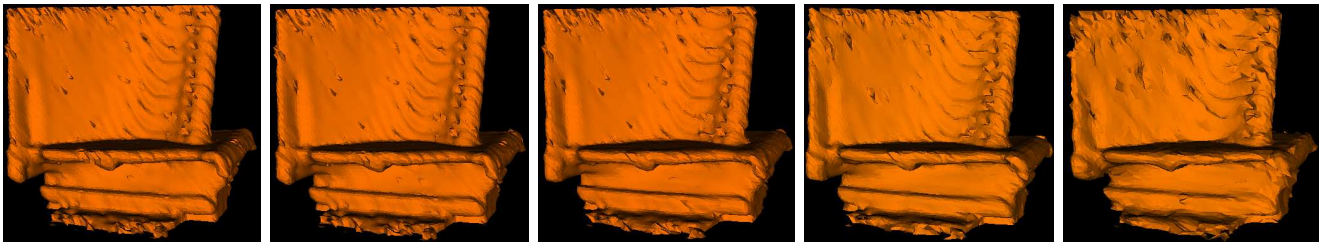

Fig. 12: From left to right: isosurfaces extracted from the original `turbine` dataset and after simplifying the mesh to 50%, 30%, 20% and 10% its original size.

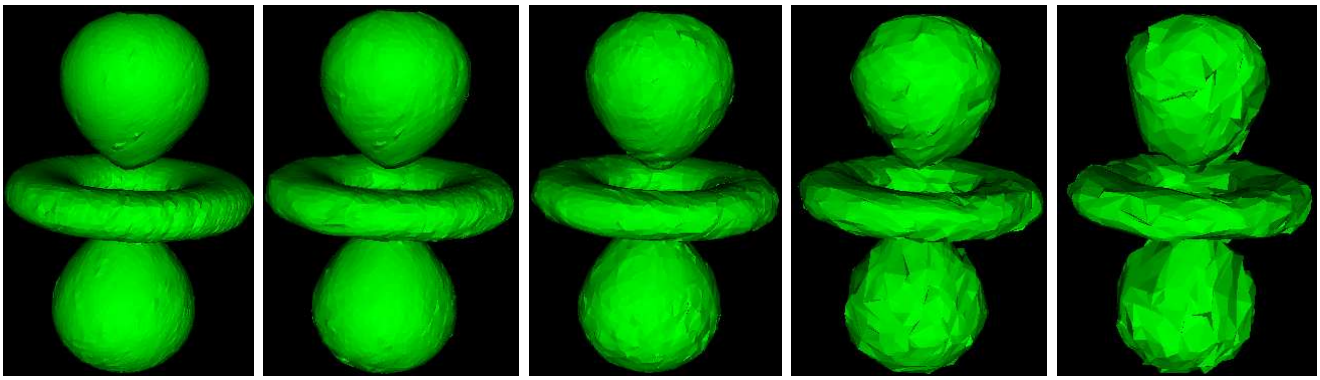

Fig. 13: From left to right: isosurfaces extracted from the original `hydrogen` dataset and after simplifying the mesh to 50%, 30%, 10% and 5% its original size.