Eurographics Symposium on Geometry Processing (2005)
M. Desbrun, H. Pottmann (Editors)

Extraction and Simplification of Iso-surfaces in Tandem †

Dominique Attali [1] , David Cohen-Steiner [2], and Herbert Edelsbrunner [3]

[1] LIS-CNRS, Grenoble, France
[2] Geometrica-INRIA, Sophia-Antipolis, France
[3] Duke University and Raindrop Geomagic, North Carolina, USA

Abstract

The tandem algorithm combines the marching cube algorithm for surface extraction and the edge contraction algorithm for surface simplification in lock-step to avoid the costly intermediate step of storing the entire extracted surface triangulation. Beyond this basic strategy, we introduce refinements to prevent artifacts in the resulting triangulation, first, by carefully monitoring the amount of simplification during the process and, second, by driving the simplification toward a compromise between shape approximation and mesh quality. We have implemented the algorithm and used extensive computational experiments to document the effects of various design options and to further fine-tune the algorithm.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Boundary representations, Hierarchy and geometric transformations F.2.2 [Analysis of Algorithms and Problem Complexity]: Geometrical problems and computations I.4.10 [Image Processing and Computer Vision]: Volumetric

1. Introduction

The work of Hoppe [Hop96] and of Garland and Heckbert [GH97] opened a new chapter on surface simplification as a central theme in geometric processing. In this paper, we contribute to the growing body of work on variants, extensions, and refinements of the original algorithm.

Historical perspective. Motivated by the demands of large datasets, the generation of simplified representations has long been a topic within computer graphics. As applied to geometric shapes, the focus has been on triangulated surfaces embedded in three-dimensional space [HDD*93, RB93, SZL92]. A breakthrough in simplifying such surfaces has been achieved in the late 1990s when Garland and Heckbert combined the edge contraction operation developed by Hoppe and co-workers [HDD*93, Hop96] with anisotropic square norms representing the accumulated error [GH97]. The confluence of these two ideas formed the starting point of developments in several directions:

- variants of the algorithm, including the restriction to topology preserving edge contractions [DEGN99] and the formulation of memoryless error measures [LT98];
- evaluation of the generated triangulations, including the mathematical analysis of the original algorithm [HG99] and the development of a software tool for experimental comparison [CRS98];
- extensions to higher dimensions, including surfaces with attributes [GH98, Hop99] and tetrahedral meshes [SG98, THJW98];
- memory-efficient processing orders that focus the algorithm to a moving window and this way simplify triangulations that may be too large to fit into main memory [ILGS03, Lin00, WK03].

Because of the central importance of simplification as a means to abstract essential information from large datasets, it is likely these themes will continue to be at the forefront of geometry processing research.

Our contributions. The work reported in this paper started with the realization that the limitation of edge contractions to memory-efficient processing orders leads to artifacts in the generated triangulations. Two questions arise: "how do we capture or quantify the artifacts?" and "how do we avoid

† Research by the first two authors was partially supported by the IST Program of the EU under Contract IST-2002-506766 (Aim@Shape). Research by the third author was partially supported by NSF grant CCR-00-86013 (BioGeometry).

them?". The answers to these questions are not independent. Specifically, we use the insights gained in quantifying artifacts towards modifying the algorithm to counteract their creation. Our approach is a mix of theoretical and experimental analysis and design. We base our experimental work on a particular memory-efficient processing order in which the marching cube algorithm for surface extraction is combined with the simultaneous simplification of the triangulation. Building a surface layer by layer, the marching cube algorithm has been intensively studied as a tool for extracting iso-surfaces from density data [LC87, JLSW02, KBSS01]. We call our approach the *tandem algorithm* because it alternates the extraction of a layer with the simplification of the accumulated partial surface. Our contributions are:

1. the formulation and implementation of the tandem algorithm;
2. the refinement of the processing order to counteract the creation of artifacts;
3. the refinement of the error quadric of Garland and Heckbert to control the mesh quality;
4. the quantification of mesh isotropy and directional bias as aspects of mesh quality.

We note that the method to control mesh quality is similar to but different from the one described in [NE04]. On the technical side, our work uses anisotropic square norms, whose mathematical formulation is developed in Appendix A. Even though we have based our work on the tandem algorithm, our results apply to other memory-efficient surface simplification algorithms and to surface simplification in general.

Outline. Section 2 describes the marching cube algorithm for surface extraction and the edge contraction algorithm for surface simplification. Section 3 explains the tandem algorithm, which combines the extraction and simplification of the surface into one step. Section 4 introduces measures that assess the quality of the extracted and simplified surfaces and compares the classical and the tandem algorithms. Section 5 concludes the paper.

2. Classical Algorithm

The classical algorithm for constructing the iso-surface of a density function on $\mathbb{R}^3$ first extracts a fine resolution representation, which it then simplifies to a more appropriate coarse resolution. We begin with a description on how the density function is given.

Density data. The most common representation of a density function $F : \mathbb{R}^3 \to \mathbb{R}$ consists of a regular cube grid and specifies the function value at every vertex. To be specific, let the grid consist of all vertices with integer coordinates $0 \leq i, j, k \leq m$ and let $F[i, j, k]$ store the function value at the point $(i, j, k) \in \mathbb{R}^3$. We sometimes refer to the third coordinate as the *rank*. The *k-th cross-section* consists of all vertices with rank k. The total number of vertices in the grid is $(m+1)^3$ and the number of vertices within a cross-section is $(m+1)^2$.

We can decompose the cubes in the grid into tetrahedra (e.g. the six tetrahedra around a space-diagonal) and extend the function values at the vertices to a continuous function $F : [0,m]^3 \to \mathbb{R}$ by linear interpolation [Mun84]. Alternatively, we may use bi-linear interpolation on the faces and tri-linear interpolation within the cubes of the grid [Far97]. A *level set* of such a continuous function consists of all points that map to a common value $C_0 \in \mathbb{R}$:

$$F^{-1}(C_0) \;=\; \{x \in [0,m]^3 \mid F(x) = C_0\}.$$

A level set is often referred to as an *iso-surface* because under reasonable genericity assumptions it is a 2-manifold with or without boundary embedded in $\mathbb{R}^3$. In the piecewise linear setting, the assumption that C_0 be different from $F(u)$ for all grid vertices u suffices to guarantee a 2-manifold. Furthermore, if $F(u) \neq F(v)$ for all grid vertices $u \neq v$ then all level sets are surfaces albeit occasionally not 2-manifolds. Figure 1 illustrates the definitions.

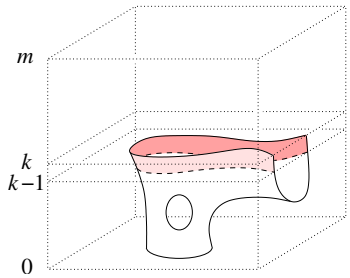

Figure 1: *Grid of data and partially extracted iso-surface with shaded layer between the k-th and $(k-1)$-st cross-sections. The portion of the surface is a connected 2-manifold of genus one with two boundary components.*

Surface extraction. Given a grid specifying a continuous density function F, and a constant $C_0 \in \mathbb{R}$, the marching cube algorithm constructs the iso-surface $\mathbb{M} = F^{-1}(C_0)$ [LC87]. We assume an implementation that returns a triangulation K of $\mathbb{M}$, which it constructs one layer at a time. To define what this means, we use the fact that each constructed triangle is contained in a single cube of the grid. It follows that given k, every vertex, edge, and triangle can be uniquely classified as *below*, *in* or *above* the plane of the k-th cross-section. Consider the subset of vertices, edges and triangles that lie in or below the k-th cross-section and call this subset minus the subset in or below the $(k-1)$-st cross-section the *k-th layer* of K. As illustrated in Figure 1, it consists of the level set within the plane of the k-th cross-section (which is a 1-manifold) and the open strip between the k-th and the $(k-1)$-st cross-sections. Assuming Function EXTRACT adds a layer to the current triangulation, we can write the marching cube algorithm as a simple `for`-loop:

```
void MARCHINGCUBE
  for k = 0 to m do EXTRACT(k) endfor.
```

Calling the plane of the k-th cross-section the *front*, we can summarize by saying the algorithm constructs the iso-surface by sweeping the front from bottom to top.

Surface simplification. The triangulation is simplified by iteratively contracting edges. The contraction of an edge $ab \in K$ removes ab together with the at most two incident triangles. The vertices a and b are glued to each other to form a new vertex c. Similarly, the two remaining edges of each incident triangle are glued to form a new edge. Each contraction has a cost, which is a measure of the numerical error it introduces. Costs are used to prioritize contractions. Initially, all edges of K are stored in a priority queue $\mathcal{Q}$. To describe the algorithm, we introduce four functions:

- MIN returns the edge with minimum cost (highest priority) and removes it from $\mathcal{Q}$;
- MINCOST returns the cost of the edge with minimum cost, without removing it from $\mathcal{Q}$;
- LAMBDA decides whether or not the contraction of an edge preserves the topological type of the triangulation, as explained in [DEGN99];
- CONTRACT performs the contraction of an edge, which includes the removal of edges from $\mathcal{Q}$ and the insertion of new edges into $\mathcal{Q}$.

We let the process continue until the minimum cost exceeds a constant error threshold $E_0 > 0$. We could contract edges until the number of vertices shrinks below a target threshold, but for reasons that will become clear later, we prefer to control the algorithm with an error threshold.

```
void SIMPLIFY(float E_0)
  while Q ≠ ∅ and MINCOST ≤ E_0 do ab = MIN;
    if LAMBDA(ab) then CONTRACT(ab) endif
  endwhile.
```

Since edges are contracted greedily, we should not expect that the resulting simpler triangulation is in any sense optimal.

3. Tandem Algorithm

If we simplify the surface right after extracting it, we might as well combine the two steps into one in a way that avoids ever storing the entire extracted surface. This is the idea of the tandem algorithm, which applies the two processes in lock-step. After describing the overall structure of the algorithm, we look at refinements that prevent artifacts caused by simplifying surface pieces with incomplete information.

Algorithm prototype. The tandem algorithm alternates between extracting one layer and further simplifying the triangulation, K, of the current surface portion. Initially, K is empty and so is the priority queue, $\mathcal{Q}$, that schedules the edge contractions. We use Function EXTRACT to add a layer to K, Function INSERT to enter the edges of a layer into $\mathcal{Q}$, and the constant $E_0 > 0$ to control the simplification process.

```
EXTRACT(0);
for k = 1 to m do EXTRACT(k);
  INSERT(k−1); SIMPLIFY(E_0)
endfor;
INSERT(m); SIMPLIFY(E_0).
```

We note that Functions INSERT and SIMPLIFY are delayed so that the top of the k-th layer remains unchanged until after the $(k+1)$-st layer has been added. Figure 2 illustrates the algorithm whose design is motivated by two partially contradicting objectives: efficiency in use of time and memory and quality of the constructed triangulation. The main challenge to efficiency is the size of the extracted triangulation, which can be huge. By alternating extraction and simplification steps, we avoid that the entire extracted triangulation has to be held in memory at any one time. The reduction in storage implies we fit the triangulation into internal memory, at all times, which improves the speed of the software by avoiding out-of-core computations. However, simplifying without complete information has potential drawbacks. The remainder of this section discusses refinements to the tandem algorithm that alleviate any detrimental effects.

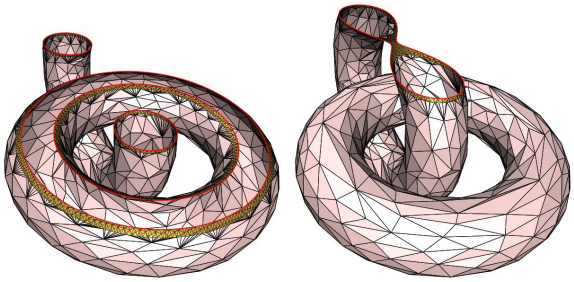

Figure 2: *Two partially extracted and simplified triangulations of the* `Link` *dataset constructed by the tandem algorithm without time-lag. The triangles of the respective last layer are shaded.*

Preimages. We need some definitions before we can describe the techniques that counteract the artifacts caused by scheduling with incomplete information. Most importantly, we think of an edge contraction $ab \mapsto c$ as a function that maps the vertices a and b to c. All other vertices are mapped to themselves. The simplification is the composition of edge contractions and can therefore be interpreted as a surjective map from the vertex set of the initial triangulation to the vertex set of the final triangulation:

$$\text{Simpl} : \text{Vert}\, K_{\text{init}} \to \text{Vert}\, K_{\text{final}}.$$

The preimage of a vertex u in the final triangulation is the collection of vertices in the initial triangulation that map to

u: $\mathrm{Simpl}^{-1}(u)$. This definition makes sense for the classical simplification algorithm, but also for the tandem algorithm by letting K_{init} be the union of the extracted and unsimplified layers.

An important concept is the set of triangles in the initial triangulation that are incident to at least one vertex in $\mathrm{Simpl}^{-1}(u)$, denoted by $\mathrm{Pre}\,u \subseteq K_{\mathrm{init}}$. Assuming all contractions preserve the topological type, it is not difficult to prove that $\mathrm{Pre}\,u \cap \mathrm{Pre}\,v \neq \emptyset$ iff uv is an edge in K_{final}. Similarly, $\mathrm{Pre}\,u \cap \mathrm{Pre}\,v \cap \mathrm{Pre}\,w \neq \emptyset$ iff uvw is a triangle in the final triangulation. In other words, the final triangulation is isomorphic to the nerve of the sets $\mathrm{Pre}\,u$ [Ede01]. Each such set of triangles defines a patch $U_u = \bigcup \mathrm{Pre}\,u$ on the initial surface. Since the final surface approximates the initial surface, the general shape of the patches is related to the general shape of the triangles in K_{final}. The lack of any information beyond the front plane encourages patches that are elongated in the directions contained in the sweep plane. We observe the same tendency in edges and triangles of the final triangulation.

Time-lag. We aim at scheduling the edge contractions in a way that counteracts this bias in the shape and direction of the edges and triangles. The goal is to obtain a triangulation as close as possible to that constructed by the classical algorithm. We therefore delay the contraction of an edge until we think the front has passed its endpoints' preimages under the classical schedule. Since the classical schedule is not available, we can only estimate when exactly this happens. In the implementation, we estimate for each vertex u the height, or third coordinate of the center and the radius of the patch U_u. To initialize this estimate, we set $\mathrm{height}(u) = \mathrm{rank}(u)$ and $\mathrm{rad}(u) = 1$. When we create a new vertex c by contracting the edge ab, we set

$$\begin{aligned} \mathrm{height}(c) &= (\mathrm{height}(a) + \mathrm{height}(b))/2; \\ \mathrm{rad}(c) &= (\|a-b\| + \mathrm{rad}(a) + \mathrm{rad}(b))/2. \end{aligned}$$

Calling $\mathrm{reach}(c) = \mathrm{height}(c) + \mathrm{rad}(c)$ the *reach* of the patch, we prevent the contraction $ab \mapsto c$ as long as $\mathrm{rank}(\mathrm{front}) < \mathrm{reach}(c)$. If both a and b lie in the front plane, we have $\mathrm{height}(c) = \mathrm{rank}(\mathrm{front})$, and since $\mathrm{rad}(c) > 0$, the contraction of ab is surely inhibited. Similarly, if a lies in the front plane but b lies below, we have $\mathrm{reach}(c) > \mathrm{rank}(\mathrm{front})$ so the contraction of ab is again inhibited. This implies that the top of the k-th layer remains untouched until after the $(k+1)$-st layer has been extracted, as before.

We keep the edges of the prevented contractions in a waiting queue, $\mathcal{W}$, ordered by reach. Whenever the front advances, we move the edges in $\mathcal{W}$ with reach less than or equal to the rank of the new front to $\mathcal{Q}$. We use Function DELAY to add the edges of the currently last layer k to $\mathcal{W}$, and we use Function ACTIVATE to move edges with reach at most k from $\mathcal{W}$ to $\mathcal{Q}$. With this notation, we can now write the new version of the tandem algorithm:

```
void TANDEM(float E0)
  for k = 0 to m do EXTRACT(k);
    DELAY(k); ACTIVATE(k); SIMPLIFY(E0)
  endfor;
  ACTIVATE(∞); SIMPLIFY(E0).
```

The last call to Function ACTIVATE happens after the entire surface has been extracted and moves the remaining edges in the waiting queue to $\mathcal{Q}$. We then continue to simplify until the cost of every edge in $\mathcal{Q}$ exceeds the error threshold, E_0. Figure 3 illustrates the effect of the time-lag strategy on the triangulation. See Figure 4 for a side-by-side comparison of triangulations obtained with the time-lag and the classical algorithm.

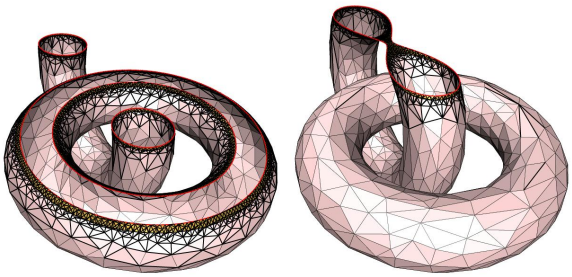

Figure 3: *Compare the partial triangulations constructed with time-lag with the ones in Figure 2 constructed without time-lag. The most striking difference is the more gradual change in edge length between the simplified portion and the yet unsimplified last layer. All edges are shown and the ones in the waiting queue are drawn thicker than the others.*

Error threshold. Similar to the original edge contraction algorithm, we use an error threshold that bounds the square distance between vertices and planes. However, instead of the sum (or integral), we use the root of the average square distance, whose maximum tends to be close to the mean distance between the extracted and the simplified surfaces; see Section 4. To explain this in detail, let c be a vertex in the current triangulation. Assuming we only apply contractions that preserve the topological type, the corresponding patch, U_c, is a topological disk. Every point $y \in U_c$ belongs to a triangle in $\mathrm{Pre}\,c$, and we let P_y be the plane spanned by this triangle. Writing $d(x,P_y)$ for the Euclidean distance between a point x and its closest point on P_y, the average square distance of x from U_c is

$$\begin{aligned} h_c(x) &= \frac{1}{\mathrm{area}(U_c)} \int_{y \in U_c} d^2(x,P_y)\ \mathrm{d}y \\ &= \frac{1}{W_c} \sum_{t \in \mathrm{Pre}\,c} w_t d^2(x,P_t), \end{aligned}$$

where the weights measure area, $w_t = \mathrm{area}(t)$, $W_c = \sum_t w_t = \mathrm{area}(U_c)$, and P_t is the plane spanned by the triangle. As described in [GH97], this average can be written as $h_c(x) =$

$\mathbf{x}^T\mathbf{H}_c\mathbf{x}/W_c$, where $\mathbf{x}$ is the point x with an added fourth coordinate equal to 1, and $\mathbf{H}_c = \sum_t w_t \mathbf{n}_t \mathbf{n}_t^T$, with $\mathbf{n}_t$ the unit normal of the plane P_t again with an added fourth coordinate, this time equal to the signed distance of the origin from the plane. Generically, $\mathbf{H}_c$ is positive definite, implying h_c has a unique minimum. Letting $ab \mapsto c$ be the contraction that creates c, it is natural to choose that minimum as the location for the vertex c in $\mathbb{R}^3$. This motivates us to refer to

$$\varepsilon_0(c) \;=\; \sqrt{h_c(c)} \;=\; \sqrt{\mathbf{c}^T\mathbf{H}_c\mathbf{c}/W_c}$$

as the *shape measure* of the resulting vertex and to use it as the priority of the edge ab in $\mathcal{Q}$.

To compute the shape measure, we may use inclusion-exclusion and get $\mathbf{H}_c = \mathbf{H}_a + \mathbf{H}_b - \mathbf{H}_{ab}$ and $W_c = W_a + W_b - W_{ab}$, as explained in [Ede01]. This requires we store a 4-by-4 matrix and a weight with every vertex, edge, and triangle in K. To reduce the amount of storage, we may alternatively compute the new matrix and weight as $\mathbf{H}_c = \mathbf{H}_a + \mathbf{H}_b$ and $W_c = W_a + W_b$, which amounts to letting each triangle $t \in \operatorname{Pre} c$ contribute once, twice, or three times depending on how many of its vertices belong to $\operatorname{Simpl}^{-1}(c)$. The multiplicity of each triangle is the same in both, so it makes sense to take the ratio and interpret it as the average square distance, now with appropriate multiplicities. All experiments presented in Section 4 are done with the latter, more memory efficient implementation.

Mesh isotropy. A disadvantage of using the shape measure by itself is the occasional creation of long and skinny triangles. These are indeed most economical in some cases, like the approximation of locally cylindrical surface pieces. However, such triangles are often undesirable if the mesh is used for downstream computations. We therefore modify the shape measure, steering the algorithm toward a compromise between accuracy of the approximation and quality of the mesh. Using a memoryless strategy, we consider the contraction $ab \mapsto c$, let T_{ab} be the collection of triangles that contain a or b or both, and let $S_{ab} = \bigcup T_{ab}$ be the patch replaced during the edge contraction. The square distance function from points of this patch is then

$$\begin{aligned} g_c(x) \;&=\; \int_{y \in S_{ab}} \|x-y\|^2 \,\mathrm{d}y \\ &=\; \sum_{t \in T_{ab}} w_t \left(\|x-\hat{t}\|^2 + \operatorname{avg}(t) \right), \end{aligned}$$

where $w_t = \operatorname{area}(t)$ is the area of the triangle t, $\hat{t}$ is its centroid, and $\operatorname{avg}(t)$ is the average square distance of a point $y \in t$ from the centroid. Letting p,q,r be the vectors from $\hat{t}$ to the vertices, we have $\operatorname{avg}(t) = \frac{1}{12}(\|p\|^2 + \|q\|^2 + \|r\|^2)$; see Appendix A. As before, we can write the function in matrix form, $g_c(x) = \mathbf{x}^T\mathbf{G}_c\mathbf{x}$, where $\mathbf{G}_c = \sum_t w_t \mathbf{G}_t$ and $\mathbf{G}_t$ is the quadric defined by three orthogonal planes passing through $\hat{t}$, with $\operatorname{avg}(t)$ added to the entry in the lower right corner. It is clear that $\mathbf{G}_c$ is positive definite and that g_c has a unique minimum, namely the centroid of S_{ab}. In contrast to $\mathbf{H}_c$, which can be singular, $\mathbf{G}_c$ has always full rank and good condition number. Figure 4 illustrates the effect of mixing the shape and the anisotropy measure in simplifying a surface triangulation.

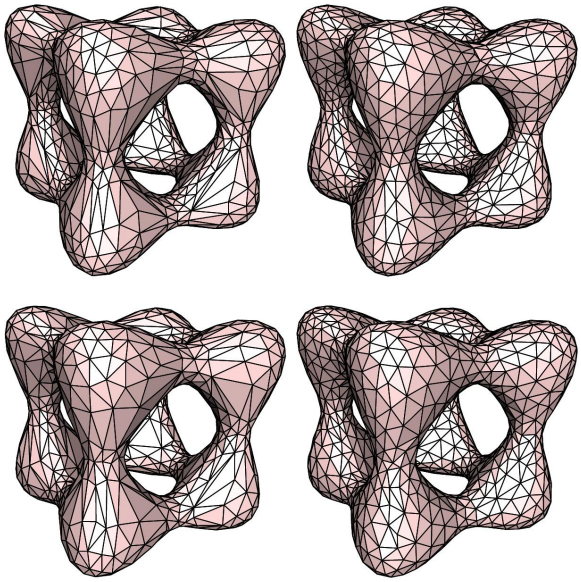

Figure 4: *Four triangulations of the* `Pillow Box` *dataset computed above by the classical algorithm and below by the tandem algorithm with time-lag, both for error threshold* $E_0 = 0.2$. *Isotropy parameter* $\alpha = 0$ *on the left and* $\alpha = 0.2$ *on the right.*

In order to balance the influence of h_c and g_c, we normalize the latter using $W = 3\operatorname{area}(S_{ab})W_c^{1/2}/E_0$. Letting $\alpha \in [0,1]$ be a constant, we choose the location of the new vertex created by the contraction $ab \mapsto c$ at the unique minimum of $(1-\alpha)h_c + \alpha g_c/W$. Accordingly, the priority of the edge is

$$\begin{aligned} \varepsilon_\alpha(c) \;&=\; \sqrt{(1-\alpha)h_c(c) + \alpha g_c(c)/W} \\ &=\; \sqrt{\mathbf{c}^T[(1-\alpha)\mathbf{H}_c/W_c + \alpha\mathbf{G}_c/W]\mathbf{c}}. \end{aligned}$$

We call $\varepsilon_1(c)$ the *anisotropy measure* of the vertex. The *isotropy parameter*, α, represents a compromise between shape measure and anisotropy measure, defined by $\varepsilon_\alpha^2 = (1-\alpha)\varepsilon_0^2 + \alpha\varepsilon_1^2$. Besides the desired effect of improving the mesh quality, g_c has also the undesired effect of moving vertices off the original surface. We therefore use a conservative strategy, allowing only contractions for which the shape measure does not exceed the error threshold, $\varepsilon_0 \le E_0$.

4. Computational Experiments

In this section, we present the results of various computational experiments we performed. With the primary goal of

understanding the impact of specific design decisions, we measure the running time, the used memory, the distance between surfaces, the mesh quality, and the directional bias of the edges and triangles in the mesh.

Datasets and sizes. We begin by introducing the datasets we use to present our experimental findings. Each set is a density map, F, specified at the integer vertices of a grid of size $(m+1)^3$. For each set, we find a density threshold, C_0, and extract the initial surface, $\mathbb{M} = F^{-1}(C_0)$, using the marching cube algorithm. Table 1 gives the size of each dataset, the percentage of cubes crossed by the surface, and the number of triangles in the extracted surface triangulation. The datasets in the first group are synthetic, being generated

dataset	grid-size	Xed cubes	#triangles
`Pillow Box`	128^3	3.53%	148,288
`Link`	128^3	1.71%	71,904
`Young Bone`	256^3	4.89%	1,641,524
`Old Bone`	256^3	2.04%	684,268

Table 1: *The number of triangles in the extracted triangulation is roughly twice the number of cubes crossed by the surface. We also note that doubling the resolution implies a roughly four-fold increase in the number of crossed cubes and a two-fold decrease in their percentage within the grid.*

from simple mathematical functions; see Figures 2, 3, and 4. The datasets in the second group are reconstructions of microscopic pieces of human bone; see Figure 5.

Running time and memory. We did all experiments on a PC with Intel Pentium M processor with clock speed of 1400MHz and available memory of 511 MB. Table 2 shows the running time needed to extract and simplify the surfaces. In each case, the tandem algorithms outperform the classical algorithm, which first extracts and second simplifies the surface. The difference is less dramatic for the synthetic datasets, for which we compute the density whenever we need it. The difference is more apparent for the experimentally obtained human bone datasets, which are stored in tables from which density values are looked up.

Figure 6 illustrates the difference in amount of memory consumed by the classical and the tandem algorithms. Besides the grid specifying the density function, which is the same for all algorithms, the main need for memory arises from storing the triangulation. We count the triangles after each extraction and after each simplification step. The alternation between extracting and simplifying practiced by the tandem algorithms expresses itself in the 'saw-tooth' character of the graph for the memory requirement. In contrast, the classical algorithm grows the triangulation one layer at a time and shrinks the size in a single final simplification step.

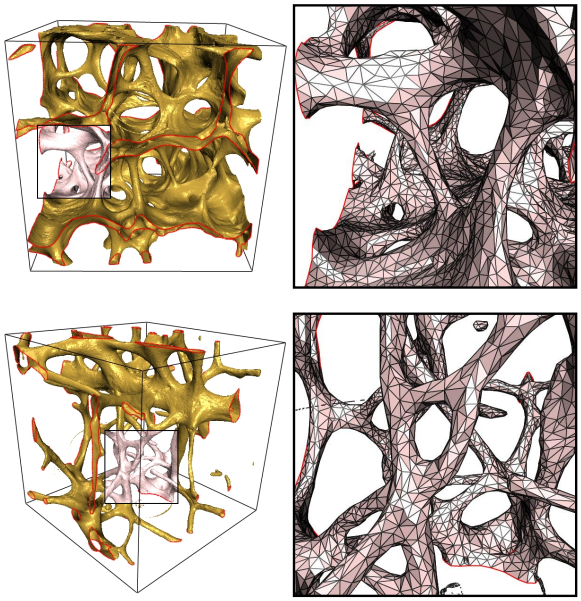

Figure 5: *Above: the surface extracted from* `Young Bone` *using the tandem algorithm with time-lag. We have 241,354 triangles on the left, which is further reduced to 39,599 triangles to show some of the details on the right. Below: the same for the* `Old Bone`*, with 233,139 triangles on the left and 19,725 triangles on the right.*

dataset	running time (in seconds) mc	ec	tm-w/o	tm-w
`Pillow Box`	15.36	10.16	21.71	23.27
`Link`	11.37	4.72	14.29	15.12
`Young Bone`	31.09	122.02	108.37	126.76
`Old Bone`	22.92	48.36	52.29	60.34

Table 2: *Running time averaged over ten runs of the classical algorithm (split into the marching cube (mc) and the edge contraction steps (ec)) and the tandem algorithm without (tm-w/o) and with time-lag (tm-w). For the synthetic datasets, the error threshold is one tenth of the side-length of an integer cube, $E_0 = 0.1$, and for the human bone datasets it is one fifth that side-length, $E_0 = 0.2$.*

Approximation error. Following [CSAD04], we measure the distance between two surfaces, $\mathbb{M}$ and $\mathbb{N}$, using averages of point-to-surface distances. More precisely, for every positive integer $p \geq 1$, the *directed L_p-distance* is

$$L_p(\mathbb{M}|\mathbb{N}) = \left(\frac{1}{\text{area}(\mathbb{M})} \int_{x \in \mathbb{M}} d^p(x,\mathbb{N}) \, \mathrm{d}x \right)^{1/p},$$

where $d(x,\mathbb{N})$ is the infimum Euclidean distance between

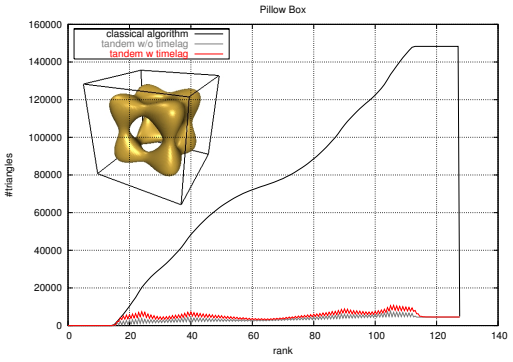

Figure 6: *Evolving size of the triangulated surface for the classical algorithm and for the tandem algorithms with and without time-lag. We observe four mild local maxima for each tandem algorithm, which correspond to the layers containing the four sets of surface saddles.*

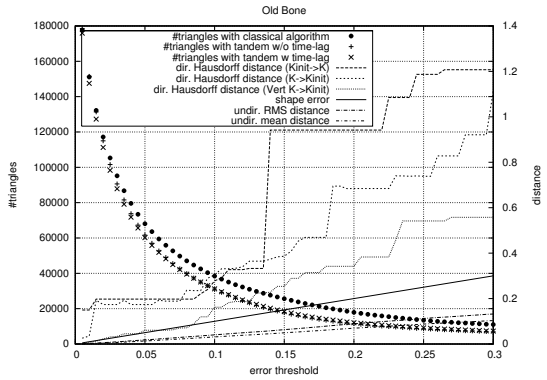

Figure 7: *Distances between the initial surface and the one generated by the classical algorithm as functions of the error threshold, E_0. The dataset is* `Old Bone` *but with lower resolution on a grid of size* 148^3. *In addition, we show the number of triangles generated by the three algorithms.*

x and points $y \in \mathbb{N}$. In the limit, we have $L_\infty(\mathbb{M}|\mathbb{N}) = \sup_x \inf_y \|x-y\|$. The *(undirected) L_p-distance* is the larger of the two measurements,

$$L_p(\mathbb{M},\mathbb{N}) \;=\; \max\{L_p(\mathbb{M}|\mathbb{N}), L_p(\mathbb{N}|\mathbb{M})\}.$$

We are particularly interested in $p = 1, 2, \infty$, referring to L_1 as the *mean distance*, L_2 is the *root mean square* or *RMS distance*, and to L_∞ as the *Hausdorff distance*. We estimate these distances using the Metro software described in [CRS98], which approximates the integrals by sums of distances from points sampled on the surfaces. We compare the distances with the shape measure used in the simplification process. Recall that $\varepsilon_0(c)$ is the root mean square distance of vertex c from the planes spanned by triangles that have at least one vertex in the preimage of the vertex. Defining the *shape error* as $\text{Error}(K) = \max_{u \in K} \varepsilon_0(u)$, we have $\text{Error}(K) \leq E_0$, by construction.

We are interested in the relation between the shape error and the distance measured between the initial and the final surface triangulations. As illustrated in Figure 7, we find linear relations between the shape error and the mean and the RMS distances, which are both smaller than $\text{Error}(K)$. Not surprisingly, the Hausdorff distance is less predictable than the others. We also find that the Hausdorff distance from the final to the initial triangulation is usually smaller than if measured in the other direction. In hind-sight, this is also not surprising since the algorithm minimizes the former distance by carefully choosing the locations of the vertices in $K = K_{\text{final}}$. As expected, we observe an inverse relationship between the number of triangles and the error threshold for each algorithm. Somewhat unexpected, however, is that the tandem algorithms consistently outperform the classical algorithm and reach triangulations with fewer triangles for almost every error threshold. We have no convincing explanation for this observation.

Mesh quality. In the context of numerical methods, triangulations are often referred to as *meshes* and used as the basis for analysis and simulation. For reasons that have to do with the convergence and stability of numerical algorithms, meshes consisting of 'well-shaped' triangles are generally preferred over meshes that contain long and skinny triangles. To quantify this difference, we define the *aspect ratio* of a triangle $t = abc$ as $\rho(t) = \sqrt{\lambda_2/\lambda_1}$, where $\lambda_1 \geq \lambda_2$ are the eigenvalues of the matrix $M_{abc} = \frac{1}{3}[aa^T + bb^T + cc^T]$, assuming the centroid is at the origin, $\frac{1}{3}(a+b+c) = 0$. As explained in Appendix A, the eigenvalues are non-negative reals, with $\lambda_1 > 0$ unless $a = b = c = 0$, and $\lambda_1 = \lambda_2$ iff the triangle is equilateral. Letting n be the number of triangles, we define the *anisotropy* as one minus the average aspect ratio:

$$\text{Anisotropy}(K) \;=\; 1 - \frac{1}{n}\sum_t \rho(t).$$

We have $0 \leq \text{Anisotropy}(K) \leq 1$ and $\text{Anisotropy}(K) = 0$ iff all triangles are equilateral.

Of primary interest is how the isotropy parameter relates to the anisotropy of the constructed mesh. As shown in Figure 8, the anisotropy decreases before it reaches a plateau. We see this pattern both for the classical algorithm and the tandem algorithm with time-lag, with the latter generally reaching lower values of anisotropy. Without time-lag, the anisotropy for the tandem algorithm is considerably higher than for the others. We believe the main reason are severe self-intersections caused by the sudden change in edge length near the front, which is already visible in Figure 2. The number of triangles increases steadily for all three

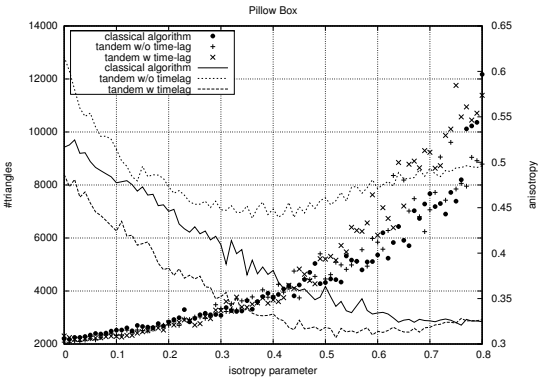

Figure 8: *The number of triangles and the anisotropy of the generated meshes as a function of the isotropy parameter,* α. *The error threshold is* $E_0 = 0.2$.

algorithms, with the classical algorithm generally producing the smallest triangulations. This can be explained by well-shaped triangles being less efficient in approximating a shape within a given error threshold.

Directional bias. Replacing each point of a surface $\mathbb{M}$ in $\mathbb{R}^3$ by its two unit normals defines a centrally symmetric density on the sphere of directions, $\mathbb{S}^2$. If $\mathbb{M}$ is a sphere itself we get the uniform density. Most other surfaces define non-uniform densities. Given a triangulation K of $\mathbb{M}$, we measure the non-uniformity with the average tensor product

$$M_n \;=\; \frac{1}{\text{area}(K)} \sum_{t \in K} w_t n_t n_t^T,$$

where w_t is the area and n_t is one of the two unit normals of the triangle t. The corresponding square norm is $d_n(x) = x^T M_n x$. As explained in Appendix A, $\|x\| = 1$ implies that $d(x)$ is the average square length of the unit normal vector components in the direction x. This is visualized by the ellipsoid of points $M_n x$, with $x \in \mathbb{S}^2$. The axes of the ellipsoid are in the directions of the eigenvectors of M_n and their half-lengths are equal to the eigenvalues, $\mu_1 \geq \mu_2 \geq \mu_3 \geq 0$. For example, if K triangulates the boundary of a pancake (a flattened sphere) then the normals are concentrated near the poles of $\mathbb{S}^2$, assuming they approximate the normals of the smooth boundary of the pancake. The square distance function, $d(x)$, is large at and near the poles and small along and near the equator that separates the poles. It follows that M_n has one large and two small eigenvalues.

We are interested in measuring the directional bias in the triangulation as opposed to the surface represented by the triangulation. In particular, we are curious to what extent the sweep-direction biases the shape of the triangles created by the tandem algorithm. We aim at something like the average triangle shape, which we express using the average tensor product

$$M_s \;=\; \frac{2}{\text{area}(K)} \sum_{t \in K} w_t \frac{M_{abc}}{\text{trace } M_{abc}},$$

where a, b, c are the vectors from the centroid to the three vertices. The corresponding square norm is $d_s(x) = x^T M_s x$. To get a feeling for this measure, consider an equilateral triangle, t. If we add its contribution to d_s to its contribution to d_n we get the weighted square distance from the origin, which is an isotropic square norm. We therefore define $M = M_n + M_s$ and the corresponding square norm $d(x) = d_n(x) + d_s(x)$. If all triangles are equilateral then M is the identity matrix and $d(x) = \|x\|^2$. Adding M_n to M_s can therefore be interpreted as removing the non-uniformity of the density of normals that is due to the shape of the surface. The matrix M and its square norm d thus represent the non-uniformity in the triangulation that remains after removing from M_s the non-uniformity inherent in the surface. We measure what remains by the *directional bias* defined as

$$\text{Bias}(K) \;=\; 1 - \frac{27 \det M}{(\text{trace } M)^3}.$$

Indeed, if M is the identity matrix then its three eigenvalues are $\mu_1 = \mu_2 = \mu_3 = 1$, $\det M = 1$, $\text{trace } M = 3$, and $\text{Bias}(K)$ vanishes. We have $0 \leq \text{Bias}(K) \leq 1$, in general, and $\text{Bias}(K) = 0$ if all triangles are equilateral. Note however that zero directional bias is also possible with non-equilateral triangles, for example if K has the symmetry group of a Platonic solid. We note that the `Pillow Box` has the symmetry group of a cube which implies that d_n is isotropic, and if its triangulation has the same symmetry group then $d(x) = \|x\|^2$.

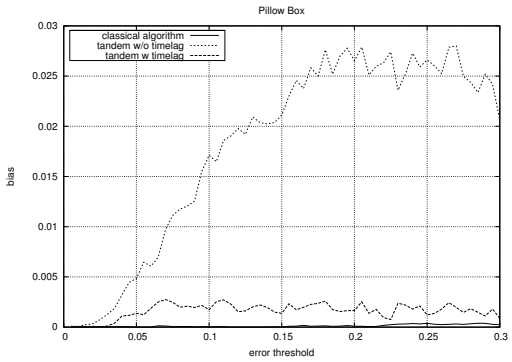

Figure 9: *The directional bias of the triangulation as a function of the error threshold.*

The results of our experiments are illustrated in Figure 9. Clearly, the time-lag strategy has a major impact, removing most of the directional bias we see in the triangulation generated by the tandem algorithm without time-lag. As mentioned above, the measurement of the directional bias is reliable only for surfaces with sufficiently rich symmetry

groups, such as the `Pillow Box`. For this dataset, M_n is indeed very close to isotropic. In contrast, the human bone datasets are not symmetric and their matrices M_n are not necessarily isotropic, which complicates the interpretation of M. Nevertheless, we can compare the directional biases computed for different triangulations and again see a striking improvement if we use the time-lag strategy.

5. Conclusion

The literature on surface simplification has recently focused on out-of-core algorithms, advocating the localization of the process to a varying patch window that eventually exhausts the surface [ILGS03, Lin00, WK03]. This is similar to our sweep approach to local simplification. None of these papers has looked at artifacts caused by the localization of the process, and we believe that they would all benefit from adapting the time-lag strategy and the anisotropy measure described in this paper.

Acknowledgement

The authors thank Françoise Peyrin from Creatis in Lyon for the two human bone datasets obtained with the Synchrotron Radiation Microtomography from the ID19 beamline at ESRF in Grenoble.

References

[CRS98] P. CIGNONI, C. ROCCHINI AND R. SCOPIGNO. Metro: measuring error on simplified surfaces. *Comput. Graphics Forum* **17** (1998), 167–174.

[CSAD04] D. COHEN-STEINER, P. ALLIEZ AND M. DESBRUN. Variational shape approximation. *Comput. Graphics*, Proc. SIGGRAPH, 2004, 905–914.

[DEGN99] 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.

[Ede01] H. EDELSBRUNNER. *Geometry and Topology for Mesh Generation.* Cambridge Univ. Press, England, 2001.

[Far97] G. FARIN. *Curves and Surfaces for CAGD. A Practical Guide.* Fourth edition, Academic Press, San Diego, California, 1997.

[GH97] M. GARLAND AND P. S. HECKBERT. Surface simplification using quadratic error metrics. *Comput. Graphics*, Proc. SIGGRAPH, 1997, 209–216.

[GH98] M. GARLAND AND P. S. HECKBERT. Simplifying surfaces with color and texture using quadratic error metrics. *In* "Proc. IEEE Conf. Visualization, 1998", 279–286.

[HDD*93] H. HOPPE, T. DEROSE, T. DUCHAMP, J. MCDONALD AND W. STÜTZLE. Mesh optimization. *Comput. Graphics*, Proc. SIGGRAPH, 1993, 19–26.

[HG99] P. S. HECKBERT AND M. GARLAND. Optimal triangulation and quadric-based surface simplification. *J. Comput. Geom.: Theory Appl.* **14** (1999), 49–65.

[Hop96] H. HOPPE. Progressive meshes. *Comput. Graphics*, Proc. SIGGRAPH, 1996, 99–108.

[Hop99] H. HOPPE. New quadric metric for simplifying meshes with appearance attributes. *In* "Proc. IEEE Conf. Visualization, 1999", 59–66.

[ILGS03] M. ISENBURG, P. LINDSTROM, S. GUMHOLD AND J. SNOEYINK. Large mesh simplification using processing sequences. *In* "Proc. IEEE Conf. Visualization, 2003", 465–472.

[JLSW02] T. JU, F. LOSASSO, S. SCHAEFER AND J. WARREN. Dual contouring of hermite data. *Comput. Graphics*, Proc. SIGGRAPH, 2002, 339–346.

[KBSS01] L. KOBBELT, M. BOTSCH, U. SCHWANECKE AND H.-P. SEIDEL. Feature sensitive surface extraction from volume data. *Comput. Graphics*, Proc. SIGGRAPH, 2001, 57–66.

[Kim98] C. KIMBERLING. Triangle centers and central triangles. *Congr. Numer.* **129** (1998), 1–295.

[LC87] W. E. LORENSEN AND H. E. CLINE. Marching cubes: a high resolution 3D surface construction algorithm. *Comput. Graphics* **21**, Proc. SIGGRAPH, 1987, 163–169.

[Lin00] P. LINDSTROM. Out-of-core simplification of large polygonal models. *Comput. Graphics*, Proc. SIGGRAPH, 2000, 259–262.

[LT98] P. LINDSTROM AND G. TURK. Fast and memory efficient polygonal simplification. *In* "Proc. IEEE Sympos. Visualization 1998", 279–286.

[Mun84] J. R. MUNKRES. *Elements of Algebraic Topology.* Addison-Wesley, Redwood City, California, 1984.

[NE04] V. NATARAJAN AND H. EDELSBRUNNER. Simplification of three-dimensional density maps. *IEEE Trans. Visual. Comput. Graphics* **10** (2004), 587–597.

[RB93] J. ROSSIGNAC AND P. BORREL. Multi-resolution 3D approximations for rendering polygonal scenes. In *Modeling in Computer Graphics: Methods and Applications*, B. Falcidieno and T. Kunii (eds.), 1993, 455–465.

[SG98] O. G. STAADT AND M. H. GROSS. Progressive tetrahedralizations. *In* "Proc. IEEE Conf. Visualization, 1998", 297–402.

[SZL92] W. J. SCHROEDER, J. A. ZARGE AND W. E. LORENSEN. Decimation of triangle meshes. *Comput. Graphics* **26**, Proc. SIGGRAPH, 1992, 65–70.

[THJW98] I. J. TROTTS, B. HAMANN, K. I. JOY AND D. F. WILEY. Simplification of tetrahedral meshes. *In* "Proc. IEEE Conf. Visualization, 1998", 287–295.

[WK03] J. WU AND L. KOBBELT. A stream algorithm for

the decimation of massive meshes. *In* "Proc. Graphics Interfaces, 2003", 185–192.

Appendix A:

In this appendix, we review standard results on matrices defined as average tensor products of sets of vectors.

Anisotropic square norms. Given a set of n vectors $u_i \in \mathbb{R}^2$, the average tensor product is

$$M = \frac{1}{n}\sum_i u_i u_i^T,$$

which is a positive semi-definite symmetric 2-by-2 matrix. The corresponding *anisotropic square norm* $d : \mathbb{R}^2 \to \mathbb{R}$ is defined by $d(x) = x^T Mx$. It has a straightforward interpretation in terms of square distances of the points u_i from lines passing through the origin. To see this, let L be such a line with unit normal x and note that $(x^T u_i)^2 = x^T u_i u_i^T x$ is the square distance of u_i from L. It follows that $d(x)$ is the average square distance of the points u_i from L.

Alternatively, we can think of M as the linear map $M : \mathbb{R}^2 \to \mathbb{R}^2$ defined by $M(x) = Mx$. Let e_1 and e_2 be the two eigenvectors. Since M is positive semi-definite symmetric, the two corresponding eigenvalues are non-negative reals, $\lambda_1 \geq \lambda_2 \geq 0$. By definition of eigenvectors and eigenvalues, we have $Me_j = \lambda_j e_j$, for $j = 1, 2$. It follows that the image of the unit circle is the ellipse with axes of half-lengths λ_j in the directions e_j, as illustrated in Figure 10. This ellipse

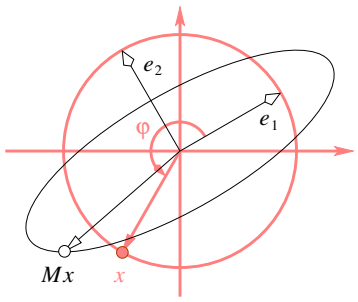

Figure 10: *The unit circle and its image under the linear map M.*

is a depiction of the average square length of the vectors u_i in all directions. Indeed, letting φ be the angle such that $x = e_1 \cos\varphi + e_2 \sin\varphi$, we have

$$\begin{aligned} d(x) &= x^T(Mx) \\ &= [\cos\varphi, \sin\varphi] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix} \\ &= \lambda_1 \cos^2\varphi + \lambda_2 \sin^2\varphi. \end{aligned}$$

In words, the average square distance depends only on the direction and the eigenvalues of the transformation matrix.

The shape of a triangle. Consider a triangle with vertices $a, b, c \in \mathbb{R}^2$ and centroid $\frac{1}{3}(a+b+c) = 0$ at the origin. The average tensor product of the three vectors is

$$M_{abc} = \frac{1}{3}\left[aa^T + bb^T + cc^T\right].$$

We call the eigenvectors of M_{abc} the *principal directions* and the square root of the ratio of eigenvalues, $\sqrt{\lambda_2/\lambda_1}$, the *aspect ratio* of abc. If $\lambda_1 = \lambda_2$ then the principal directions are ambiguous and the aspect ratio is 1. To get started, we observe that unit aspect ratio characterizes equilateral triangles. Let uvw be an equilateral triangle whose vertices are at unit distance from the centroid at the origin. We note that

$$\frac{1}{3}\left[uu^T u + vv^T u + ww^T u\right] = \frac{u}{3} - \frac{v+w}{6} = \frac{u}{2},$$

which implies that u is an eigenvector of M_{uvw}. By symmetry, v and w are also eigenvectors. Three different eigenvectors are only possible in the ambiguous case, when M_{uvw} has two equal eigenvalues. Any other triangle abc with centroid at the origin defines a transformation matrix A such that $a = Au$, $b = Av$, $c = Aw$. The matrix defined by the new triangle is $M_{abc} = \frac{1}{3}[Auu^T A^T + Avv^T A^T + Aww^T A^T] = AM_{uvw}A^T$. It has two equal eigenvalues iff AA^T does. The latter condition is equivalent to A being a similarity and to abc being equilateral.

We note in passing that the image of the circle passing through u, v, w is an ellipse that passes through a, b, c and has its center at the centroid of abc. This is known as the *Steiner circumellipse*, which is the unique area-minimizing ellipse that passes through the three points [Kim98]. Heckbert and Garland use the ratio of axes of this particular ellipse to define the aspect ratio of a triangle [HG99]. It is not difficult to show that this notion of aspect ratio agrees with the one based on M_{abc} given above.

Equivalent formulations. We conclude this appendix by noting that the same matrix, M_{abc}, can be obtained by summing over vectors different from a, b, c. Take for example the edge vectors $p = b - a, q = c - b, r = a - c$ of the triangle and note that $pp^T + qq^T + rr^T = 3aa^T + 3bb^T + 3cc^T$, which implies $M_{pqr} = 3M_{abc}$. We may also average over all points in the triangle $t = abc$ and get the *covariance matrix*,

$$M_t = \frac{1}{\text{area}(t)}\int_{x\in t} xx^T \, \mathrm{d}x,$$

which can be shown is equal to one quarter the average tensor product of the vertices, $M_t = \frac{1}{4}M_{abc}$. We omit the proof. The trace of the covariance matrix is

$$\text{trace } M_t = \frac{1}{\text{area}(t)}\int_{x\in t} x^T x \, \mathrm{d}x,$$

which is the average square distance of a point x from the centroid at the origin. Since the trace of M_t is one fourth of the trace of M_{abc}, we now have a convenient formula for the average square distance from the centroid, namely $\text{avg}(t) = \frac{1}{12}[a^T a + b^T b + c^T c]$.

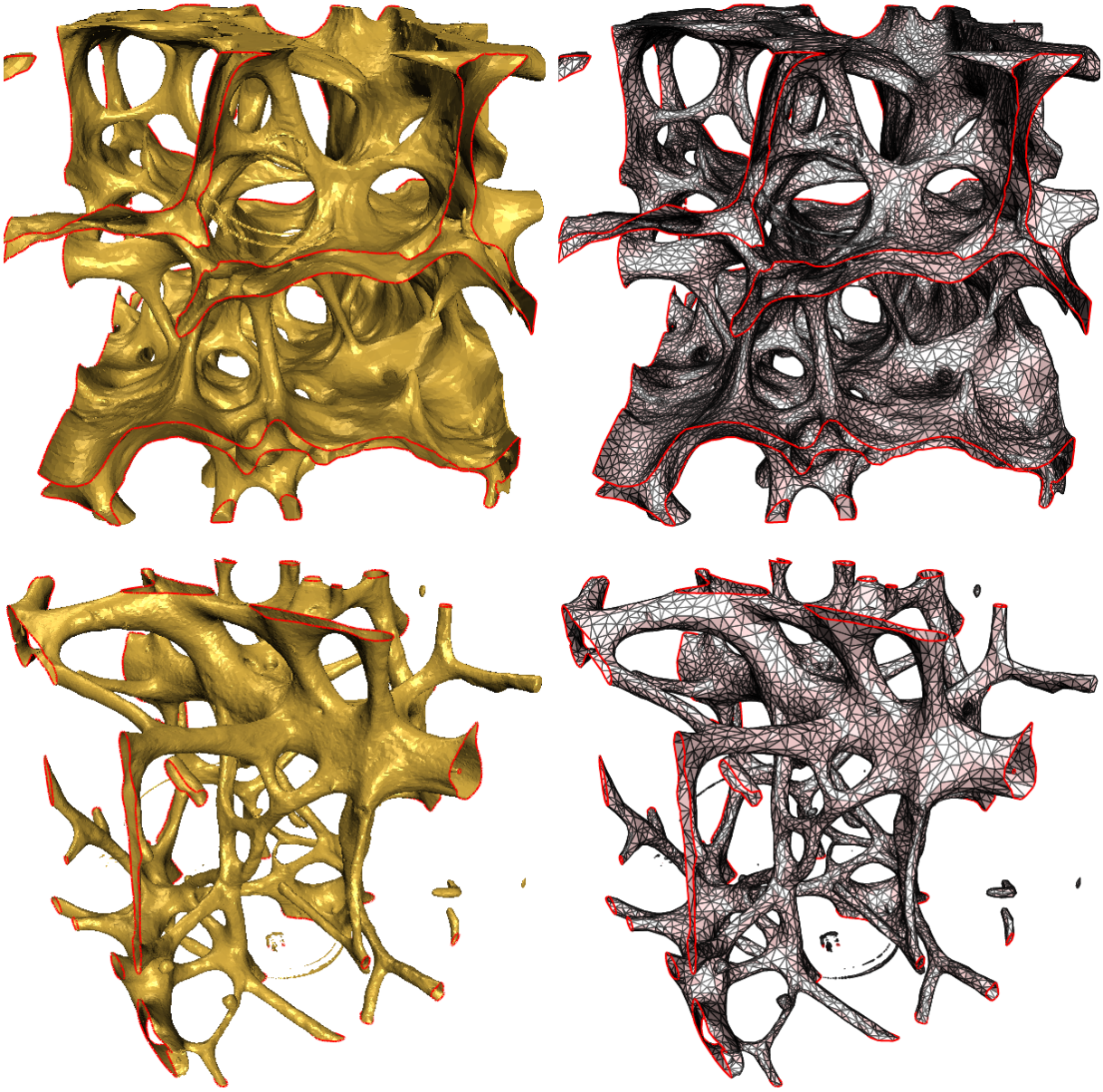

Figure 11: *Above: the surface extracted from* `Young Bone` *using the tandem algorithm with time-lag. We have 209,321 triangles on the left, which is further reduced to 36,645 triangles on the right. Below: the same for the* `Old Bone`*, with 205,482 triangles on the left and 21,103 triangles on the right.*