Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

Simplification and Improvement of Tetrahedral Models for Simulation

B. Cutler,[1] J. Dorsey,[2] and L. McMillan[3]

[1] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
[2] Department of Computer Science, Yale University
[3] Department of Computer Science, University of North Carolina at Chapel Hill

Abstract

Most 3D mesh generation techniques require simplification and mesh improvement stages to prepare a tetrahedral model for efficient simulation. We have developed an algorithm that both reduces the number of tetrahedra in the model to permit interactive manipulation and removes the most poorly shaped tetrahedra to allow for stable physical simulations such as the finite element method. The initial tetrahedral model may be composed of several different materials representing internal structures. Our approach targets the elimination of poorly-shaped elements while simplifying the model using edge collapses and other mesh operations, such as vertex smoothing, tetrahedral swaps, and vertex addition. We present the results of our algorithm on a variety of inputs, including models with more than a million tetrahedra. In practice, our algorithm reliably reduces meshes to contain only tetrahedra that meet specified shape requirements, such as the minimum solid angle.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - volumetric simplification, tetrahedral models

1. Introduction

With the increased speed and availability of computing cycles, physically-based simulations are becoming more practical and popular in computer graphics. We are interested in modeling and rendering detailed solid objects with complex internal structures and interactively sculpting or manipulating those models with physically-plausible simulations. Recent developments in 3D scanning technology have made it easy to acquire highly-detailed surface models of real-world objects. These meshes are a natural starting point for the creation of complex solid models; however, preparing interesting volumetric models for simulation is a difficult task. The initial models usually have too many tetrahedra, and the shape of many of the tetrahedra is poor, making physical simulations impossible. In this paper, we present an algorithm that takes a complex volumetric model and makes it viable for simulation.

1.1. Related Work

Below we describe previous research in mesh generation, simplification, improvement and refinement. A more detailed survey of these techniques has been compiled by Owen [37], and the companion website includes a huge database of the different meshing implementations currently available. Unfortunately, we could not use any of these off-the-shelf packages for our application because they are not robust enough to handle complex scanned surface geometry and cannot be used to design objects with layers of material or internal structures.

Mesh Generation There are three basic techniques for meshing the interior of a 3D surface. The Advancing Front and Advancing Layers techniques directly tetrahedralize a volume by adding well-proportioned (near equilateral) tetrahedra, one at a time, to the interior of a triangle mesh [30, 38]. This method works well when the initial triangle mesh is manifold and consists of well-proportioned triangles. However, it is non-trivial to implement, and in many applications it is not necessary to match the input surface triangulation. For example, scanned meshes and polygonized implicit surfaces contain many triangles, which are evenly distributed rather than concentrated in areas of high detail.

Another approach is to compute a Delaunay triangulation of the vertices and then override the Delaunay property by performing various element swaps to match the original surface edges and faces [4, 8, 15, 41]. Unfortunately, in 3D and higher dimensions, the Delaunay property alone is insufficient to guarantee well-shaped tetrahedra and often

these models require further modifications to improve element shape.

The most brute-force technique, structured grid or octree tetrahedralization, is simple and straightforward to implement and will always produce a consistent mesh [36, 44]. However, the method produces a large number of tetrahedra, and elements near a boundary often have near-zero volume and very poor shape. This technique is best paired with simplification and mesh improvement.

Mesh Simplification Motivated by the need for interactive rendering and transmission of complex meshes, much work has been done on simplifying triangular surface meshes while maintaining surface fidelity [22, 40]. Some of these techniques have been translated to volumetric meshes [9, 10, 11, 42, 43]. Unfortunately, these techniques focus primarily on simplification, and do not necessarily improve the *quality* of the mesh elements. Work on nested hierarchical coarsenings can guarantee bounds on element shape in 2D triangulations [6, 32], but has not been demonstrated on complex volumetric models.

Mesh Improvement A variety of local transformations can be used to improve the quality of individual elements [17, 24]. In practice, a combination of remeshing techniques is more effective at improving shape than any single type of operation [16]. We have expanded on this work and show that the addition of aggressive simplification to traditional mesh improvement techniques can achieve better overall tetrahedral shape.

Mesh Refinement To increase the resolution of a mesh for improved simulation accuracy and to implement geometry modification operations such as fracture and the removal of material, meshes can be locally refined. One approach is to use *regular subdivision* (in 2D, simply bisect each edge to create four self-similar triangles), which can be made adaptive with careful bookkeeping [5, 7]. Another approach is to perform successive, local longest-edge bisections [1, 39]. This technique is effective for 2D planar triangulations; however, in higher dimensions it may not terminate, and in general, it does not maintain element quality. Instead, each original tetrahedron should be projected to a special *reference tetrahedron*, upon which longest-edge bisection results in finitely many congruency classes of tetrahedra [29]. Similarly, vertex ordering [31] or edge marking [2] schemes may be used. However, if the mesh is to be deformed or fractured, local simplification and/or mesh improvement may also be necessary to ensure well-proportioned elements [19].

1.2. Overview

Our work makes the following contributions:

- Simplification of complex tetrahedral models generated from scanned surface meshes, while preserving the topology and surface detail of exterior and interior material boundaries.
- Improvement of tetrahedral shape/proportion to meet the requirements of physical simulations such as the Finite Element Method (FEM).
- New element quality metric that relies on just a few user-specified parameters, which are intuitive and easy to set based on the application.
- Robust and efficient implementation to simplify and improve large tetrahedral models with more than a million tetrahedra. We plan to make the implementation publicly available.

We detail the tetrahedral mesh requirements of our target application in Section 2. In Section 3, we present our algorithm, which eliminates the most poorly-shaped elements by using edge collapses and other local remeshing operations, which are described in Section 4. In Section 6, we present the results of the implementation on a variety of interesting models. Finally, in Section 7, we compare the algorithm to similar simplification and mesh improvement techniques.

2. Goals and Requirements

Often, the initial tetrahedralization of a volumetric model has too many tetrahedra and many of the elements have poor shape, making physical simulations impossible. There are a number of objectives, described below, that must be addressed for these meshes to be useful in an interactive modeling system and to behave correctly during simulation.

Reduce the Overall Number of Elements Generally, the running time of a physical simulation is polynomial in the number of elements. Our interactive FEM simulation engine operates at interactive rates with models of 5,000 or fewer tetrahedra [33, 34], so this is often the target tetrahedral count for simplification.

Maintain the Shape and Topology of Boundaries One of the main goals of any simplification algorithm is to preserve the original data, though some small-scale features must be sacrificed to achieve the target element count. For our application, it is important to maintain the shape and topology of both exterior (material/air) and interior (material/material) boundaries. For example, if the object is covered by a thin layer of material, the simplified object should also have this layer. A naive implementation might allow a thin layer to *collapse*, revealing the underlying material [25].

An alternate goal used in other applications is to preserve a scalar/vector field defined throughout space. These goals are related when we consider preserving multiple isosurfaces embedded in a volume. However, in order to maintain the visual appearance of a rendered isosurface it is necessary to consider the normals of the boundary faces [20].

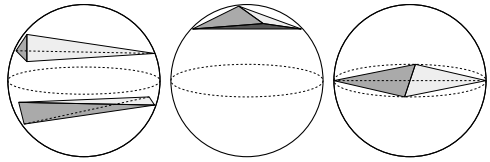

Figure 1: *Illustration of poorly-shaped tetrahedral elements. Unlike the 2D case, it is not sufficient to judge element shape quality by the ratio of shortest and longest element edges. While this test detects needle and wedge tetrahedra, it fails for sliver tetrahedra. A better metric in 3D is the minimum solid angle. Illustration redrawn from [41].*

Improve or Maintain the Shape of Each Element Physical simulations and interactive manipulation such as sculpting or deformation move vertices of the mesh relative to each other, and thus place restrictions on the shape of volumetric elements. Errors related to the FEM approximation increase when the elements have extremely large dihedral angles, and the stiffness matrix is severely constrained when these angles are very small [3, 26]. In general, such tetrahedral elements are said to be degenerate or have poor *shape* (Figure 1). To ensure mathematical stability of the simulations and guarantee that the volume of each element remains positive, the initial model should consist of only well-shaped tetrahedral elements. If a mesh contains a *sliver* tetrahedron, the slightest translation of one of its vertices can cause the sign of its volume to flip, leading to errors when determining the exterior triangular skin of the model, and resulting in incorrect renderings and missed collisions. Adding constraints to the simulation to prevent inverted volumes introduces non-linearities to the system, making it expensive and difficult to solve. It is much more efficient to begin the simulation with well-proportioned elements that can withstand considerable deformation.

In 2D, the quality of a triangle's shape can be measured as a ratio of longest to shortest edges. However, in 3D, a tetrahedron with similar edge lengths is not guaranteed to have good shape; e.g., a sliver tetrahedron. Instead, we chose to measure the quality of a tetrahedron's shape by its minimum solid angle. Other 3D metrics that are equally effective in judging tetrahedral shape include the surface area to volume ratio and the minimum dihedral angle [28].

Maintain a Reasonable Distribution of Elements Simulations on uniform density meshes, in which all elements have similar volume and are nearly equilateral, are accurate and stable, but slow, due to the high number of elements. For many applications, the best compromise is to use a mesh with comparable surface detail but a gradation in element sizes; that is, larger tetrahedra on the interior of the model and in areas of low detail and smaller tetrahedra near the surface details. However, if the elements are too large, the model may be over-constrained and will not behave realistically. Soft material deformations will look polygonal rather than continuous and the choice of fracture planes for a brittle material will be limited. These problems can be addressed by adaptive refinement of highly strained elements; however, refinement is expensive and often too slow for an interactive simulation.

Scalability The simplification and mesh improvement strategy should be scalable, and handle large models with more than a million tetrahedra. It will be primarily used for offline computation; though to be useful in an iterative design process, the results should be available in about an hour, even for the largest models. Additionally, the strategy could be applied locally, for online remeshing.

3. Algorithm

Block Iteration The basic approach of our iterative simplification and mesh improvement algorithm is to identify and correct poorly-shaped tetrahedra by using well-known atomic remeshing operations (Section 4). Poorer quality elements have higher priority for these operations to ensure their removal or improvement (if possible). However, we do not insist on a total ordering of the operations, as this would be expensive (see Section 7). Instead, as an acceleration, we process a block of the poorest quality elements in arbitrary order. The algorithm estimates q, a quality cutoff value, gathers the tetrahedra with quality less than q, randomly reorders the list, and then tries to perform one of the local mesh operations on each element. If an element has been modified by a previous action on a neighbor, it is skipped.

Tetrahedral Shape Quality Metric To find the tetrahedra targeted for elimination or improvement, a quality metric $\mathrm{Quality}(t)$, ranging from 0.0 (very poor shape/volume) to 1.0 (near perfect shape/volume), is computed for each tetrahedron t. The metric is the geometric mean of three factors that each range from 0.0 to 1.0. A geometric mean rather than an arithmetic mean is used to ensure that a poor rating for any one factor dominates the score for the element.

$$\begin{aligned}
\mathrm{Quality}(t) &= \sqrt[3]{\mathrm{Q}_{angle}(t) \cdot \mathrm{Q}_{volume}(t) \cdot \mathrm{Q}_{edge}(t)} \\
\mathrm{Q}_{angle}(t) &= \frac{\text{minimum solid angle}(t)}{0.55} \\
\mathrm{Q}_{volume}(t) &= \text{clamp}\left(\frac{\text{volume}(t)}{\textit{ideal volume}}\right) \\
\mathrm{Q}_{edge}(t) &= \text{clamp}\left(\frac{\beta \cdot \textit{ideal edge length}}{\text{longest edge}(t)}\right)
\end{aligned}$$

The first two terms, $\mathrm{Q}_{angle}(t)$ and $\mathrm{Q}_{volume}(t)$, reward elements that are near equilateral (minimum solid angle $\approx$ 0.55 steradians) and have volume greater than or equal to the *ideal volume* (total model volume / *target tetrahedral*

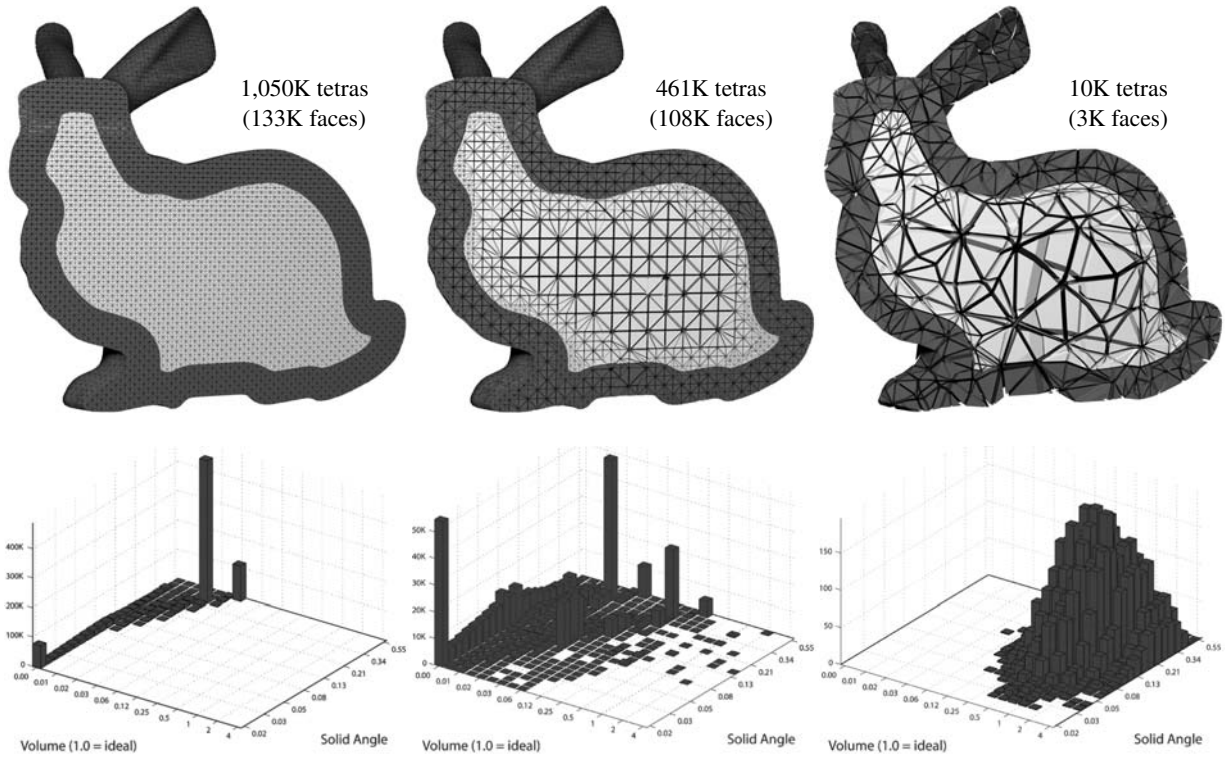

Figure 2: *The meshes generated from a uniform distance field (top left) and an octree version of the same distance field (top middle) have similar boundary quality. The simplified version (top right) maintains the exterior and interior interfaces. Below each mesh is a histogram of the element quality. The leftmost corner of each plot represents the worst quality elements (near zero volume, near zero minimum solid angle), and the rightmost corner represents the best elements.*

count). The final term, $Q_{edge}(t)$, penalizes a tetrahedron whose longest edge is much greater than the *ideal edge length* ($\sqrt[3]{\textit{ideal volume}}$). This last term may appear redundant, but we found it necessary to discourage the creation of tetrahedra with poor angles but reasonable volume due to their large overall size (long edges). These elements can be difficult to remove in later iterations. In our examples we use $\beta = 5$. Larger values for β allow a wider range in element size; that is, larger elements on the interior of the model. Each factor is clamped to $[0, 1]$. A visualization of initial and final element quality is shown in Figure 2.

Surface Preservation Metric On each iteration, the algorithm considers many possible actions but does not perform any that introduce negative volume elements, worsen the local minimum element quality, or significantly modify the exterior or interior boundary surfaces relative to the allowable boundary error, E. Our initial value for E is the expected average edge length in the simplified model, the *ideal edge length*. We define $\text{Error}(op)$, the boundary error introduced by operation op, with a volume conservation metric [27], which preserves these boundaries by penalizing change in volume of any material (computed separately).

$$\text{Error}(op) \;=\; \gamma \cdot \sum_{materials} \sqrt[3]{\text{abs}(\Delta \text{ volume})}$$

If $\text{Error}(op) > E$ then op is not performed. Larger values for γ result in increased retention of surface detail but poorer element quality. We found $\gamma = 10$ to be an effective parameter setting and use this value for all examples shown in this paper. Other surface preservation metrics can be substituted [20].

Minimum Solid Angle Requirements Often it is impossible to both maintain the boundary surface of a particular model and enforce a minimum shape requirement on all elements. Thus we do not place a hard requirement on the minimum solid angle; rather, the algorithm works to improve element shape as much as possible. In general, the algorithm only performs operations that locally improve the model; that is, the poorest-quality tetrahedron after the operation should be no worse than the poorest-quality tetrahedron before the operation. However, several consecutive operations

E = *ideal edge length*
percent = 10%
while (tetrahedral count > *target tetrahedral count*)
  q = estimate cutoff for poorest *percent*
  $T = \{\ t \mid \text{Quality}(t)\ \leq\ q\ \}$
  randomly reorder T
  foreach $t \in T$, try these actions:
    • tetrahedral swaps
    • edge collapse
    • move a vertex
    • add a vertex
  percent += 10%
  $E *= \sqrt[3]{\frac{\text{tetrahedral count}}{\textit{target tetrahedral count}}}$

Figure 3: *Pseudo-code for our algorithm.*

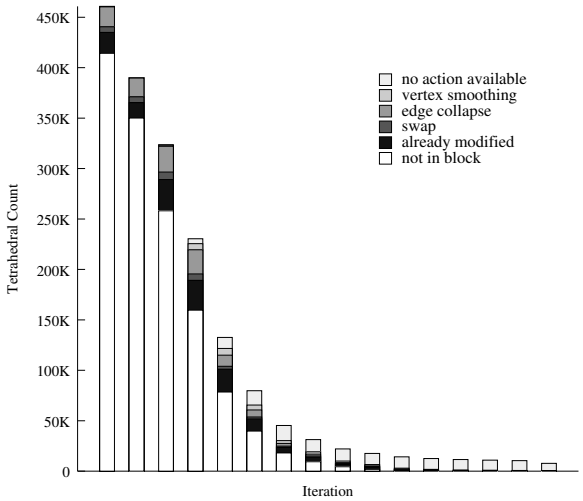

Figure 4: *Each bar in this visualization represents an iteration of the algorithm. The height of the bar is the number of tetrahedra at the beginning of the iteration, with the poorest tetrahedra at the top, where early computation is focused. The number of elements is quickly reduced and then levels off to allow the model to settle into the best compromise between element quality and surface detail for the desired number of tetrahedra.*

may be necessary to remove/improve a particular element. These operations may temporarily create worse quality elements, trapping a greedy approach in a local minima. So in the spirit of simulated annealing, early in computation we allow some operations that worsen the local quality of the mesh and decrease the chance of taking these steps to zero as we approach our target tetrahedral count.

Algorithm Summary Pseudo-code for the algorithm is shown in Figure 3, and a visualization of the number of operations performed during each iteration is shown in Figure 4. Two variables are updated after each block iteration: the quality cutoff, q, is updated to select an additional 10% of the poorest quality elements in the next iteration, and the boundary error, E, is scaled by the (dimension-corrected) tetrahedral-count reduction remaining. The only user-specified parameters to the algorithm are the *target tetrahedral count* and the *target solid angle*. Application-specific controls may be added; for example, the internal parameters β and γ could be exposed to the user.

4. Mesh Operations

Below we briefly describe the well-known local mesh operations that are used to simplify and improve tetrahedral meshes. Our algorithm attempts to perform these operations in the order listed to improve connectivity, remove vertices, adjust vertex positions, and add vertices. Further studies are needed to determine if this is the optimal order, or if perhaps a random choice would be more effective.

Swapping The first actions attempted are tetrahedral swaps, which can dramatically improve tetrahedral shape by switching inter-element connectivity. We have implemented the two most common types of tetrahedral swapping, shown in Figure 5, but other tetrahedral swaps are possible [16]. The $2 \rightarrow 3$ swap is performed by removing two adjacent tetrahedral elements and replacing them with three tetrahedra that share the edge connecting the two opposite vertices. The $3 \rightarrow 2$ swap is performed in reverse. Although the configuration with two tetrahedra contains fewer elements, the configuration with three tetrahedra is often selected because it has a better local minimum element quality. The $2 \rightarrow 2$ swap allows re-triangulation of the exterior boundary.

Point Deletion Next, the algorithm attempts to eliminate the target tetrahedron by performing a half-edge collapse (Figure 7), which removes one vertex and all tetrahedral elements surrounding the collapsed edge. Elements that pointed to the removed vertex are stretched to instead point to the vertex at the other end of the edge. There are two directions in which each of a tetrahedron's six edges may be collapsed. Some of these collapses may be disallowed because they introduce negative-volume tetrahedra or change the topology of the exterior or interior boundary surfaces (Figure 6). Other methods for identifying topology-preserving collapses may be used [14, 43]. Certain collapses are more desirable because they more accurately preserve the details of boundaries. Like Progressive Meshes [22], we use an edge weighting function, Weight(e), to prioritize edge collapses that best maintain surface details.

$$\text{Weight}(e)\ =\ \text{length}(e)\ +\ \text{Error}(\text{collapse}(e))$$

If no e exists such that Weight(e) < E, then no collapse will be performed on this tetrahedron, during this iteration.

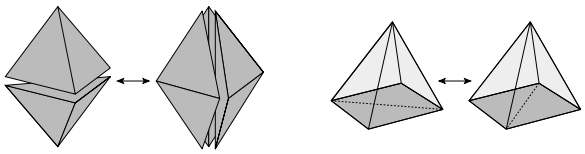

Figure 5: *Two tetrahedra that share a face may be replaced with three tetrahedra that share an edge (or vice versa). Similarly, two tetrahedra that share a face and have neighboring boundary faces that are nearly parallel can be replaced with two tetrahedra that essentially swap the edge between these two boundary faces. Illustration redrawn from [16].*

Vertex Smoothing If the procedures outlined above are unable to improve the shape of the targeted tetrahedron, the algorithm will move each vertex of the element to the centroid of its connected vertex neighbors [17, 16]. A vertex is moved only if the new position does not introduce any tetrahedra of negative volume and $\text{Error}(\text{smooth}(v)) < E$.

Point Addition A final remeshing option is the addition of new vertices at the center of a tetrahedron, the center of a boundary face, or at the midpoint of a boundary edge [41]. Although these actions will increase the total number of tetrahedra, they can provide significant help when the mesh contains roughly the target number of elements, but the shape requirements have not been met.

5. Implementation

To achieve a constant running time for the mesh operations described above, inter-element neighbor pointers must be maintained. We have implemented the algorithm using a 3D analog of the half-edge data structure. A hash table is used to pair the faces of neighboring elements from "polygon soup" in constant time. For efficiency, we cache element quality and edge collapse weights. After each operation, we update the element neighbor pointers and invalidate cached values, as appropriate. To ensure the robustness of our implementation, we found it necessary to perform the following degeneracy checks on the resulting mesh when considering a particular operation:

- Exactly two tetrahedra or one tetrahedron and one exterior triangle share each triangular face.
- Exactly two exterior triangles share each exterior edge.
- At least three exterior triangles meet at each exterior vertex.
- At least three elements meet at each edge.
- All elements sharing a vertex can be reached by following a chain of tetrahedral neighbor pointers.
- Two tetrahedra that share a face must share exactly three vertices (a naive edge collapse implementation may create an illegal situation where a pair of tetrahedra also share the fourth vertex).

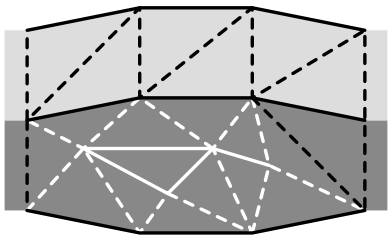

Figure 6: *2D illustration showing different edge collapses.* Interior edges *(solid white) are completely surrounded by one type of material and can be collapsed since they do not affect the material boundaries.* Boundary touching edges *(dashed white) are surrounded by one material at their interior endpoint and two or more at their boundary endpoint. They should be collapsed to the boundary endpoint.* Spanning edges *(dashed black) are surrounded around the edge by a single material type, but have at least one other material at each endpoint. They should never be collapsed, as this would cause a point of zero thickness in a material layer.* Boundary edges *(solid black) can be collapsed as long as the boundary is not unacceptably modified.*

Some of these checks are guaranteed if all elements have positive volume; however, they should be consistently performed due to limited floating point precision and rounding errors. Our implementation is tolerant of negative- and zero-volume elements that are often present in the input meshes or at intermediate stages of a deformation simulation.

6. Performance and Results

Our example meshes are generated using structured, grid-based mesh generation [12], although the algorithm can also be applied to output from other techniques. A sufficiently high resolution grid is chosen to adequately capture the details of the input surface. Unfortunately this also results in a large number of tetrahedra evenly distributed throughout the volume of the model without regard to the complexity of surface detail. The models may be composed of different materials, arranged in layers or other patterns [13]. A material layer that is thick relative to the grid leads to extraneous tetrahedral sampling within the layer (Figure 2). Prior to tetrahedralization, grid cells that are adequately represented by interpolation of the coarser grid or do not contain an interface crossing are collapsed. Tetrahedralization of this octree or *Adaptive Distance Field* results in fewer tetrahedra [18]. Cells on the boundary between differing octree resolutions are triangulated to avoid T-junctions and cracks.

Our simplification and mesh improvement implementation has been stable and effective, even on very large meshes (Figures 8 and 9). Simplifications with smaller target tetrahedral counts often have faster running times (Table 1) because the initial boundary error value, E, is higher. This results in a

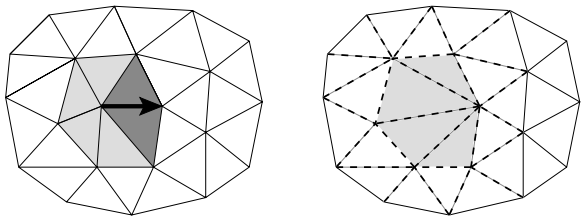

Figure 7: *When an edge is collapsed (black arrow), the triangles or tetrahedra that share the edge are removed (dark gray) and the elements that touch the deleted vertex are deformed (light gray). Since the collapse weight of an edge includes data about element shape, the weight of all edges that share a vertex with a deformed element (dashed lines) must be recomputed.*

model	tetra count	# of iters	mm:ss	min angle < 0.1	min after
bunny	461K → 2K	18	07:10	41.7% → 18.5%	0.02
	→ 10K	16	10:44	→ 10.1%	0.02
	→ 50K	15	23:12	→ 5.6%	0.01
dragon	825K → 5K	18	14:31	39.7% → 25.9%	0.02
	→ 100K	13	35:32	→ 4.4%	0.01
hand	1,596K → 10K	18	29:01	40.5% → 24.8%	0.02
	→ 100K	16	58:32	→ 4.7%	0.02
gargoyle	792K → 10K	17	15:29	39.4% → 11.4%	0.02
	→ 30K	15	22:15	→ 6.7%	0.01
	→ 50K	14	25:19	→ 5.5%	0.01
	→ 200K	12	29:30	→ 5.4%	0.01

Table 1: *Statistics for the meshes shown in Figures 8 and 9 run on a Pentium 4 Xeon 2.4 GHz machine with 4 GB of RAM. From left to right: the number of tetrahedral elements in the initial and final meshes; the number of block iterations performed; the running time; the percent of elements in the initial and final meshes with solid angle < 0.1 steradians; and the minimum solid angle of all elements in the final mesh. In each example, the* target solid angle *was 0.1 steradians.*

faster reduction of the tetrahedral count and fewer elements to process in later iterations. The actual times vary due to the general difficulty of estimating the boundary error, E, needed to represent a complex object with a given number of elements. If the model can be represented with large tetrahedra on the interior (saving more elements for the surface) the final boundary error will be smaller.

In practice, all tetrahedra exceed the minimum quality requirements for simulations. The simplified models have been tested in an interactive volumetric editing, visualization and simulation environment and perform quite well. Traditionally, creating models for use in such systems has been challenging due to the fixed time steps and minimum required refresh rates.

7. Comparison to Previous Work

We considered using a Progressive Mesh (PM) approach [9, 22, 42] for the simplification phase of our system. In a PM, each edge is given a weight that indicates how its collapse will affect various mesh properties such as connectivity, conformity to the original mesh, etc. The edges are stored in a priority queue, and after each edge is collapsed, neighboring edge weights are recomputed and reheaped as necessary. However, maintaining a total ordering of the edges is expensive, as well as unnecessary for many applications, such as extreme simplification of very large models. After a single edge collapse operation is performed on a tetrahedral mesh, an average of over 100 edge weights must be recomputed and re-heaped (see Figure 7). Furthermore, the cost of actually performing an edge collapse is small compared to the cost of recomputing the weight of an edge collapse to determine whether it is legal and desirable.

In contrast to a PM, with our algorithm the edge weights are computed only for the current block of low-quality tetrahedra, and it is very likely that if a legal collapse exists, it will be performed. In the initial iterations of our algorithm, many operations are performed in an arbitrary order, saving considerable computation. The change in the boundary error, ΔE, is decreased as the target tetrahedral count is approached. In the limit, as $\Delta E \to 0$, our method (restricted to edge collapses) is equivalent to the continuous simplification of a Progressive Mesh. We found that our algorithm required only 5-15% as many edge collapses as an equivalent PM. This is considerable since about 80-90% of the running time is spent recomputing edge weights.

Overall, our algorithm has much in common with the Mesh Optimization [23] approach, in that we also use element swaps, vertex smoothing, and vertex addition. These operations are necessary to improve meshes that are already near their target tetrahedral count or contain very poor inter-element connectivity. We select local remeshing operations with a greedy strategy (to ensure well-proportioned elements), and use the boundary error, E, to control the global optimization of the mesh.

By combining mesh improvement operations with aggressive simplification, our technique achieves higher element quality in practice than mesh improvement alone. As expected, our algorithm is somewhat slower than the simplification-only algorithms, mostly due to the consistency and topology checks required for robustness. Our running times are comparable to the simplification and mesh improvement strategy of [35]; although they focus on preserving a scalar field defined throughout space rather than the boundary surfaces and therefore the extracted isosurfaces contain significant visual artifacts.

8. Conclusions and Future Work

Simplification is a necessary component in modeling to produce meshes at a resolution appropriate for interactive exploration, manipulation, and simulation. Our strategy focuses aggressive simplification and mesh improvement operations on the poorest-quality elements to ensure that these models are also viable for physical simulations. Our technique is efficient enough to be used during iterative modeling. The algorithm could easily be adapted for local mesh improvement, targeting areas where simulation or sculpting have modified the mesh and created poorly-shaped tetrahedra.

In our implementation, the topology of the object and the materials within the object are strictly preserved. However, noise and alignment errors in the acquisition of 3D scanned meshes often lead to digital models with higher genus than the original object. Topological simplification of surface meshes [21] can be used to pre-process the input surface prior to mesh generation. Alternatively, topological simplification of the tetrahedral model could be performed when erroneous or extraneous topology is identified.

Acknowledgments

We would like to thank Hugues Hoppe for helpful discussions, Justin Legakis for the use of his rendering software, Derek Bruening for writing batch scripts and proofreading, and Frédo Durand for additional feedback. This work was supported by NSF grants CCR-9988535, CCR-0072690, and EIA-9802220 and by a gift from Pixar Animation Studios.

References

[1] A. Adler. On the bisection method for triangles. *Mathematics of Computation*, 40(162):571–574, April 1983.

[2] D. N. Arnold, A. Mukherjee, and L. Pouly. Locally adapted tetrahedral meshes using bisection. *SIAM Journal of Scientific Computing*, 22(3):431–448, 2000.

[3] I. Babuska and A. Aziz. On the angle condition in the finite element method. *SIAM Journal of Numerical Analysis*, 13(2):214–226, 1976.

[4] T. J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained delaunay triangulation. *Engineering with Computers*, (5):161–175, 1989.

[5] R. E. Bank, A. H. Sherman, and A. Weiser. Some refinement algorithms and data structures for regular local mesh refinement. pages 3–17, 1983.

[6] R. E. Bank and J. Xu. An algorithm for coarsening unstructured meshes. *Numerische Mathematik*, 73(1):1–36, 1996.

[7] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

[8] P. R. Cavalcanti and U. T. Mello. Three-dimensional constrained delaunay triangulation: a minimalist approach. In *Proceedings of the 8th International Meshing Roundtable*, pages 119–129, 1999.

[9] Y.-J. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum*, 22(1):493–504, 2003.

[10] P. Chopra and J. Meyer. Tetfusion: An algorithm for rapid tetrahedral mesh simplification. In *Proceedings of IEEE Visualization 2002*, pages 133–140, 2002.

[11] P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with accurate error evaluation. In *IEEE Visualization 2000*, pages 85–92, 2000.

[12] B. Cutler. *Procedural Authoring of Solid Models*. PhD dissertation, Massachusetts Institute of Technology, August 2003.

[13] B. Cutler, J. Dorsey, L. McMillan, M. Müller, and R. Jagnow. A procedural approach to authoring solid models. *ACM Transactions on Graphics*, 21(3):302–311, July 2002.

[14] T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S.)*, 66:23–45, 1999.

[15] H. Edelsbrunner and D. Guoy. An experimental study of sliver exudation. *Engineering With Computers, Special Issue on 'Mesh Generation' (10th IMR 2001)*, 18(3):229–240, 2002.

[16] L. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40:3979–4002, 1997.

[17] W. H. Frey and D. Field. Mesh relaxation: A new technique for improving triangulations. *International Journal for Numerical Methods in Engineering*, 31(6):1121–1133, 1991.

[18] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 249–254, 2000.

[19] F. Ganovelli and C.O'Sullivan. Animating cuts with on-the-fly re-meshing. In *Eurographics 2001 - short presentations*, pages 243–247, 2001.

[20] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 209–216, 1997.

[21] I. Guskov and Z. Wood. Topological noise removal. In *Graphics Interface*, pages 19–20, June 2001.

[22] H. Hoppe. Progressive meshes. In *Proceedings of ACM SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 99–108, 1996.

[23] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 19–26, Aug. 1993.

[24] B. Joe. Construction of three-dimensional improved-quality triangulations using local transformations. *Mathematics of Computation*, 16:1292–1307, 1995.

[25] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proceedings of ACM SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 302–311, 2002.

[26] M. Krizek. On the maximum angle condition for linear tetrahedral elements. *SIAM Journal of Numerical Analysis*, 29:513–520, 1992.

[27] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization*, pages 279–286, 1998.

[28] A. Liu and B. Joe. Relationship between tetrahedron shape measures. *BIT*, 34:268–287, 1994.

[29] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM Journal of Scientific Computing*, 16(6):1269–1291, November 1995.

[30] R. Lohner. Generation of three-dimensional unstructured grids by the advancing front method. In *International Journal for Numerical Methods in Fluids*, volume 8, pages 1135–1149, 1988.

[31] J. M. Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *SIAM Journal of Scientific Computing*, 16:210–227, 1995.

[32] G. L. Miller, D. Talmor, and S.-H. Teng. Optimal good-aspect-ratio coarsening for unstructured meshes. In *Proceedings of the Eighth Annual Symposium on Discrete Algorithms*, pages 538–547, New Orleans, Louisiana, Jan. 1997. Association for Computing Machinery.

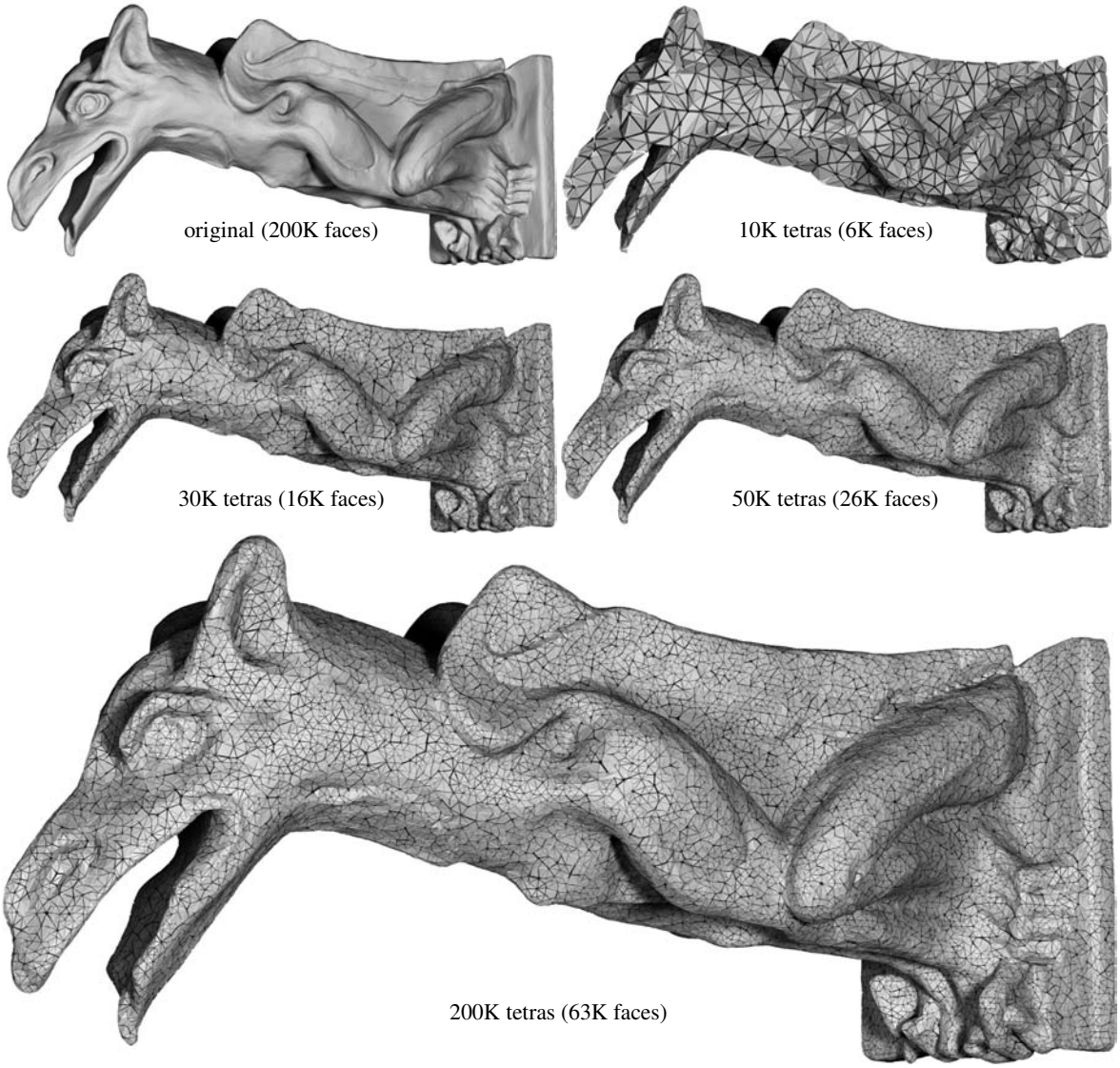

Figure 8: *Renderings of the gargoyle model at several resolutions.*

[33] M. Müller, J. Dorsey, L. McMillan, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of Eurographics Workshop on Animation and Simulation 2001*, pages 113–124, 2001.

[34] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002*, pages 49–54, 2002.

[35] V. Natarajan and H. Edelsbrunner. Simplification of three-dimensional density maps. *IEEE Transactions on Visualization and Computer Graphics*, to appear.

[36] G. M. Nielson and J. Sung. Interval volume tetrahedrization. In *IEEE Visualization '97*, pages 221–228, 1997.

[37] S. J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 239–267, 1998. `http://www.andrew.cmu.edu/user/sowen/softsurv.html`.

[38] S. Pirzadeh. Three-dimensional unstructured viscous grids by the advancing-layers method. 34(1):43–49, January 1996.

[39] M.-C. Rivara and C. Levin. A 3-D refinement algorithm suitable for adaptive and multi-grid techniques. *Communications in Applied Numerical Methods*, 8:281–290, 1992.

[40] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[41] J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD dissertation, Carnegie Mellon University, School of Computer Science, May 1997. Technical Report CMU-CS-97-137.

[42] O. G. Staadt and M. H. Gross. Progressive tetrahedralizations. In *IEEE Visualization '98*, pages 397–402, October 1998.

[43] I. J. Trotts, B. Hamann, K. I. Joy, and D. F. Wiley. Simplification of tetrahedral meshes. In *IEEE Visualization '98*, pages 287–296, October 1998.

[44] M. A. Yerry and M. S. Shephard. Automatic three-dimensional mesh generation by the modified octree technique. *International Journal For Numerical Methods in Engineering*, (20):1965–1990, 1984.

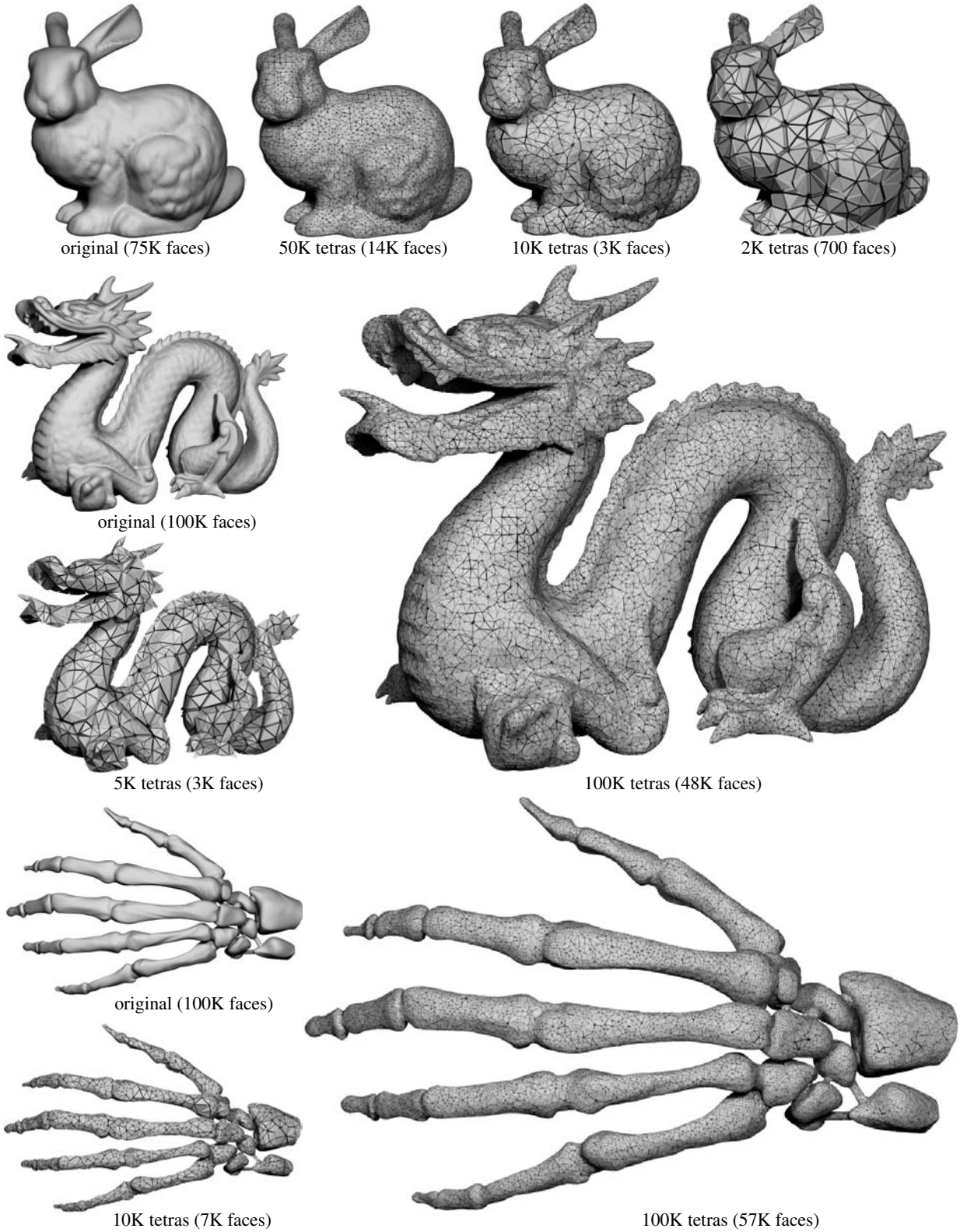

Figure 9: *Renderings of representative results. The original bunny and dragon meshes are from Stanford University and the original hand mesh is from Clemson University.*