# Fast and Memory Efficient Polygonal Simplification

Peter Lindstrom Greg Turk

Georgia Institute of Technology

## Abstract

Conventional wisdom says that in order to produce high-quality simplified polygonal models, one must retain and use information about the original model during the simplification process. We demonstrate that excellent simplified models can be produced without the need to compare against information from the original geometry while performing local changes to the model. We use edge collapses to perform simplification, as do a number of other methods. We select the position of the new vertex so that the original volume of the model is maintained and we minimize the per-triangle change in volume of the tetrahedra swept out by those triangles that are moved. We also maintain surface area near boundaries and minimize the per-triangle area changes. Calculating the edge collapse priorities and the positions of the new vertices requires only the face connectivity and the the vertex locations in the intermediate model. This approach is memory efficient, allowing the simplification of very large polygonal models, and it is also fast. Moreover, simplified models created using this technique compare favorably to a number of other published simplification methods in terms of mean geometric error.

## 1 INTRODUCTION

Automatic model simplification begins with a geometric description of an object and produces a new description that is similar in appearance to the original and that has many fewer geometric primitives. Many approaches to simplification have appeared in the recent literature, at least in part because this technique has the potential to dramatically speed up many interactive graphics applications. Such techniques are becoming increasingly important due to the increasingly large models that are created from medical data acquisition (CT, MRI, etc.), range scanners, computer vision algorithms, satellite radar, and other sources. Models with more than one million triangles are becoming commonplace, thus we have paid particular attention to the performance of our method on large models.

Some computational problems seem to attract a never-ending array of proposed solutions. Such problems include sorting, Delaunay triangulation, convex hull, collision detection, hidden surface determination, ray tracing acceleration, and polygonal model simplification. Inventors of new algorithms seek new techniques that strike a balance between a number of algorithm characteristics, some that are domain-specific, and others that are more universal. A few of the important characteristics of any algorithm include speed, memory efficiency, robustness, and ease of coding. These basic considerations were a guide to our own work in polygonal model simplification.

We use edge collapse as the method for simplifying geometry, as have a number of previous researchers. This approach selects an edge and replaces it with a single vertex (Figure 2). This removes one vertex, three edges, and two faces. The edge collapse operation is attractive because it allows the new vertex to be placed in a manner that helps preserve the location and shape of the original surface. It is also a more atomic operation than vertex or face removal, and does not require the invocation of a triangulation algorithm. Two decisions are central to a simplification method that uses edge collapse: 1) the position of the new vertex created by the edge collapse, and 2) the ordering of the edges to be collapsed (the edge priority). We use volume and surface area information to make both of these decisions.

We constrain the placement of new vertices so that the volume of a closed model is not altered. If the new vertex is near a boundary of the model, we also preserve the surface area in the region surrounding the edge that is being collapsed. Often these two constraints do not fully determine the position of the new vertex, so we optimize additional geometric properties. We minimize the volume swept out by triangles that are moved by the operation, minimize the area swept out by boundary edges, and finally attempt to produce well-shaped triangles if the vertex is still underconstrained. Our method unifies these different constraints by describing each of them as one or more planes in which the new vertex must lie. When three non-parallel planes are determined, the vertex position is fully defined. We use per-triangle volume and area differences to dictate the priority of edge collapses.

The remainder of the paper is organized as follows. First, we review some of the previous work on polygonal simplification. Next, we describe how we select the new vertex resulting from an edge collapse and also how related information is used to prioritize the edge collapses. We then present numerical comparisons between our approach and other published methods. Finally, we conclude with directions for future research.

## 2 PREVIOUS WORK

There have been a large number of recent publications on automatic model simplification, hence it is not possible to cover all of this work here. In this section we will concentrate on the particular forms of geometric information that guides various simplification methods. Our attention below is focused on those simplification algorithms that make many small local changes to the geometry of a model. Not all simplification methods are based on incremental changes, and exceptions include [6, 11, 19, 22].

Some of the earliest simplification algorithms used successive vertex removal as the method for gradually simplifying a model. Schroeder and co-workers use the distance from a vertex to the plane most nearly passing through adjacent vertices as their priority measure [20]. No history of the original geometry is kept during this process. A more recent variant of this technique includes a scalar value at each vertex that encodes the maximum error created so far in the neighborhood of the vertex [21]. Renze and Oliver concentrate on triangulation algorithms in their work, and they use the same distance to plane method as Schroeder [17]. Hamann uses a measure of local curvature to decide which vertices to remove, and here again no history of the original surface is used to guide these decisions [10]. More recently, Gieng, Hamann and coworkers successively collapse triangles into vertices, and the new vertex is placed using a local quadratic approximation to the nearby surface [8].

Some more recent vertex removal techniques use some form of history about the original geometry as a way of tracking the error incurred during simplification. Bajaj and Schikore keep track

of "positive" and "negative" error bounds at each vertex throughout the iterative removal process [1]. Planar projections of old and new triangles in a region allow them to compute these error bounds. Ciampalini and co-workers associate with each triangle a list of vertices from the original model [2]. These vertices are used to maintain an error estimate for each triangle during simplification. Cohen et al. use geometric envelopes built around the original model to constrain the selection of vertices that may be removed [4].

A number of methods use iterative edge collapse to simplify models. Hoppe and co-workers use edge collapse together with edge swap and edge split to create simplified surfaces [12]. They use a distance measure from a sampled set of points on the original model in order to determine new vertex positions. In more recent work, Hoppe uses only edge collapses to simplify models, and still uses distance to sampled points as a guide to simplification [13]. Ronfard and Rossignac essentially keep at each vertex a list of planes that are a history of the original surface in the region surrounding the vertex [18]. This information is used to prioritize edge collapses and to place the new vertices. Their work was a jumping off point for Garland and Heckbert, who maintain a 4 by 4 symmetric matrix at each vertex that allows them to track squared distances to planes of the original model [7]. The position of a new vertex is found simply by minimizing the quadric error. Elements of our simplification method are related to such quadric error measures, and we will return to this point later. Guéziec associates error radii at each vertex during successive edge collapses in order to bound the simplification error [9]. To our knowledge, his method is the first approach that explicitly adds a constraint to vertex placement in order to maintain the volume of the original model. Our method is also volume preserving, but our additional position constraints and our edge cost are quite different. (Guéziec uses edge length for edge cost.) Cohen and co-workers use edge collapse operations and a planar projection analysis similar to those of Bajaj and Schikore to bound the error in a region [5]. They keep track of this error by associating an axially-aligned box to each triangle. Conceptually, the original surface is guaranteed to lie within the union of all such boxes convolved with their associated triangles.

Two trends in simplification have attracted our attention. One is that more researchers appear to be using edge collapse (or generalizations that allow topology modification) to create local changes to the geometry. Our work follows the edge collapse paradigm. A second trend is that more algorithms are maintaining some form of history about the original surface. Our approach to vertex placement and edge cost determination yields a counter-intuitive result: we can perform high-quality simplification without retaining any history about the original model.

# 3 Notation

Before describing our simplification method, we will briefly introduce some terminology and notation. The topological entity called a 0-simplex, or a *vertex*, is denoted by $v$, with its geometric counterpart written as the 3-vector $\mathbf{v}$. A non-oriented edge $\bar{e}$ is a set $\{v_0^e, v_1^e\} = \{\lfloor e \rfloor_0, \lfloor e \rfloor_1\}$, where $\lfloor s \rfloor$ denotes the $n-1$-*faces* of an $n$-simplex $s$; in this case the vertices of the 1-simplex $e$. Oriented edges are written as ordered pairs $\vec{e}_{0,1} = (\lfloor e \rfloor_0, \lfloor e \rfloor_1)$. All higher order simplices are assumed to be oriented unless otherwise specified, and we use the distinction $\bar{s}$ and $\vec{s}$ only to resolve ambiguities. A *triangle*, or 2-simplex, is a set of oriented edges, e.g. $t = \{\vec{e}_0^t, \vec{e}_1^t, \vec{e}_2^t\} = \{(v_0^t, v_1^t), (v_1^t, v_2^t), (v_2^t, v_0^t)\}$. For convenience, we sometimes write $t = (v_0^t, v_1^t, v_2^t)$ to mean $\{(v_0^t, v_1^t), (v_1^t, v_2^t), (v_2^t, v_0^t)\}$.

The operator $\lceil s \rceil$ gives the $n+1$-simplices of which an $n$-simplex $s$ is a subset of, e.g. $\lceil v \rceil$ denotes the edges that are incident upon the vertex $v$. This notation trivially extends to sets, for example $\lfloor S \rfloor = \{\lfloor s \rfloor : s \in S\}$. Thus, the operator $\lfloor S \rfloor$ reduces the dimension of $S$ by one, while $\lceil S \rceil$ adds a dimension. See Figure 1 for examples of these operators.

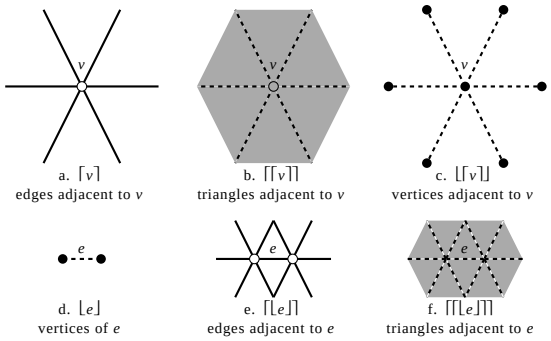



Figure 1: The simplex operators $\lfloor s \rfloor$ and $\lceil s \rceil$.

Simplices are classified as *boundary*, *manifold*, and *non-manifold*. A boundary edge has exactly one incident triangle, a manifold edge has two, while non-manifold edges have three or more incident triangles. We write the boundary of a set $S$ as the set of oriented edges $\partial S = \{\vec{e} : e \in S, |\lceil e \rceil| = 1\}$.

All vectors are assumed to be column vectors, and are written as small, bold-face letters; matrices are written as capital letters. Transposition is used to denote row vectors, e.g. $\mathbf{v}^{\mathrm{T}}$. The expression $\mathbf{u} \times \mathbf{v}$ denotes the cross product of two 3-vectors. We use the shorthand notation $(\mathbf{v}_0 \quad \mathbf{v}_1 \quad \mathbf{v}_2)$ to denote the $3 \times 3$ matrix made up of three column vectors. The functions $V$, $A$, and $L$ denote signed volume, signed area, and length, respectively.

# 4 SIMPLIFICATION ALGORITHM

## 4.1 Algorithm Overview

Our simplification method consists of repeatedly selecting the edge with a minimum cost, collapsing this edge, and then re-evaluating the cost of edges affected by this edge collapse. Specifically, an edge collapse operation takes an edge $e = \{v_0, v_1\}$ and substitutes its two vertices with a new vertex $v$. In this process, the triangles $\lceil e \rceil$ are collapsed to edges, and are discarded. The remaining edges and triangles incident upon $v_0$ and $v_1$, i.e. $\lceil \lfloor e \rfloor \rceil - \{e\}$ and $\lceil \lceil \lfloor e \rfloor \rceil \rceil - \lceil e \rceil$, respectively, are modified such that all occurrences of $v_0$ and $v_1$ are substituted with $v$. Figure 2 illustrates the edge collapse operation.

The first step in the simplification process is to assign costs to all edges in the mesh, which are maintained in a priority queue. For each iteration, the edge with the lowest cost is selected and tested for candidacy. As we will describe later, an edge is rejected as a candidate if no solution exists for its replacement vertex. We additionally impose some topological constraints to preserve the genus and to avoid introducing non-manifold simplices. If the edge is not a valid candidate, its cost is set to infinity, and the edge is moved to the back of the queue. Given a valid edge, the edge collapse is performed, followed by a re-evaluation of edge costs for all nearby edges affected by the collapse. As the triangles $\lceil \lceil \lfloor e \rfloor \rceil \rceil$ are modified, all other edges $\{e_j\}$ for which $\lceil \lceil \lfloor e \rfloor \rceil \rceil \cap \lceil \lceil \lfloor e_j \rfloor \rceil \rceil \neq \emptyset$ must be updated, i.e. $\{e_j\} = \lceil \lfloor \lceil v \rceil \rfloor \rceil$, with $|\{e_j\}| = 38$ on average. Once the costs for $\{e_j\}$ have been updated, the next iteration begins, and the process is repeated until a desired number of simplices remain.

The general edge collapse method involves two major steps: choosing a measure that specifies the cost of collapsing an edge $e$, and choosing the position $\mathbf{v}$ of the vertex $v$ that replaces the edge. Many approaches to vertex placement have been proposed, such as

picking one of the vertices of $e$, using the midpoint of $e$, or choosing a position that minimizes the distance between the mesh before and after the edge collapse. This problem can be viewed as an optimization problem, that is, given an objective function $c(e,\mathbf{v})$ that specifies the cost of replacing $e$ with $v$, $\mathbf{v}$ is chosen such that $c$ is minimized. We have chosen a cost function $c$ that encapsulates volume and area information about a model. These geometric issues are described in the following subsections.

## 4.2 Vertex Placement

In choosing the vertex position $\mathbf{v}$ from an edge collapse, we attempt to minimize the change of several geometric properties such as volume and area. We require a unified mathematical framework for such constraints. Our basic approach to finding $\mathbf{v}$ is to combine a number of linear equality constraints $\mathbf{a}_i{}^{\mathrm{T}}\mathbf{v} = b_i$, i.e. $\mathbf{v}$ is the intersection of three non-parallel planes in $\Re^3$. We have decided to incorporate more than three such constraints in the event that two or more of them are linearly dependent, and the constraints are computed and added in a pre-determined order of importance. If two or more of these planes are nearly parallel, minor perturbations to the plane coefficients lead to large variations in the solution. Such perturbations frequently occur due to roundoff errors and limited numerical precision. To compensate for such problems, we add a constraint $(\mathbf{a}_n, b_n)$ to a set of existing constraints $(\mathbf{A},\mathbf{b})$ only if the plane normal $\mathbf{a}_n$ does not fall within an angle $\alpha$ of all linear combinations of the plane normals $\{\mathbf{a}_i : 0 < i < n\}$ of the previous constraints. Thus, given $n-1$ previous constraints ($n \leq 3$), we accept $(\mathbf{a}_n, b_n)$ if any of the following conditions hold:

(i) $n = 1$ and $\mathbf{a}_1 \neq \mathbf{0}$

(ii) $n = 2$ and $(\mathbf{a}_1{}^{\mathrm{T}}\mathbf{a}_2)^2 < (\|\mathbf{a}_1\|\|\mathbf{a}_2\|\cos(\alpha))^2$

(iii) $n = 3$ and $((\mathbf{a}_1 \times \mathbf{a}_2)^{\mathrm{T}}\mathbf{a}_3)^2 > (\|\mathbf{a}_1 \times \mathbf{a}_2\|\|\mathbf{a}_3\|\sin(\alpha))^2$

If the constraint meets these conditions, we say that it is $\alpha$-compatible with the list of prior constraints. For all results presented in this paper, $\alpha$ has been set to $1^\circ$. After three compatible constrains have been found, $\mathbf{v}$ is computed as

$$\mathbf{v} = \mathbf{A}^{-1}\mathbf{b} \tag{1}$$

That is, $\mathbf{a}_n{}^{\mathrm{T}}$ is the $n^{th}$ row of $\mathbf{A}$.

Throughout the remainder of this paper, we will assume that $e$ is the edge to be collapsed, $v$ is the replacement vertex, $\{t_i\} = \lceil\lceil\lfloor e\rfloor\rceil\rceil$ are the triangles surrounding an edge, $\{\vec{e}_i\} = \partial\lceil\lfloor e\rfloor\rceil$ are the boundary edges of the changing region, and $\{v_i\} = \lfloor\lceil v\rceil\rfloor - \{v\}$ are the neighboring vertices to the edge.

### 4.2.1 Volume Preservation

When an edge is collapsed, the local shape of the model is generally modified. If the replacement vertex is not chosen carefully, each edge collapse will alter the volume of the model. For example, if each edge is substituted by its midpoint, repeated edge collapses of a tessellated sphere will reduce the sphere to an inscribed tetrahedron whose volume is much smaller than the original sphere. Similarly, edge collapses in concave regions result in a volume increase. Even if the model is not a closed, manifold, orientable surface, we can think of local surface patches as bounding some volume, and preserving the volume both locally and globally is desirable as it tends to preserve both the 3D shape of the model and its 2D projection.

In this discussion, we will assume that $\lceil\lceil\lfloor e\rfloor\rceil\rceil$ with its edges and vertices are manifold. In fact, it matters little whether the mesh is locally manifold or not, or if it has a boundary; the preservation and

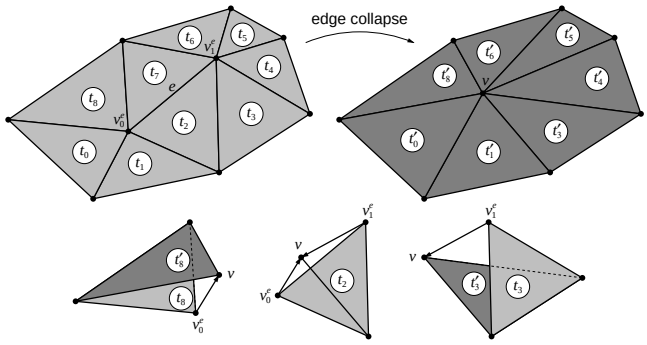



Figure 2: The edge collapse operation. The manifold edge $e$ is collapsed and replaced with a vertex $v$. Triangles $t_2$ and $t_7$ are removed in the process. Example tetrahedral volumes associated with triangles $t_2$, $t_3$, and $t_8$ are shown.

optimization steps generally handle such cases in a consistent and intuitive manner. First, observe what collapsing $e$ does to volume of the model. When a triangle $t = (v_i^e, v_1^t, v_2^t)$ is replaced by $t' = (v, v_1^t, v_2^t)$, a volume is swept out by $t$ as $\mathbf{v}_i^e$ moves in a linear path to $\mathbf{v}$ (Figure 2). This volume is described by a tetrahedron $p = (v, v_i^e, v_1^t, v_2^t)$. If $\mathbf{v}$ is "above" the plane of $t$ (i.e. outside the model), we say that the volume of $p$ is positive, and the model expands locally at $t$. Conversely, if $\mathbf{v}$ is "below" the plane, $p$ yields a negative volume contribution, and the model shrinks. Thus, to preserve the volume of the model, we set

$$\sum_i V\left((v, v_0^{t_i}, v_1^{t_i}, v_2^{t_i})\right) = \sum_i \frac{1}{6}\begin{vmatrix} v_x & v_{0x}^{t_i} & v_{1x}^{t_i} & v_{2x}^{t_i} \\ v_y & v_{0y}^{t_i} & v_{1y}^{t_i} & v_{2y}^{t_i} \\ v_z & v_{0z}^{t_i} & v_{1z}^{t_i} & v_{2z}^{t_i} \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0 \tag{2}$$

and solve for $\mathbf{v}$. Equation 2 can be rewritten as

$$\sum_i (\mathbf{v}_0^{t_i} \times \mathbf{v}_1^{t_i} + \mathbf{v}_1^{t_i} \times \mathbf{v}_2^{t_i} + \mathbf{v}_2^{t_i} \times \mathbf{v}_0^{t_i})^{\mathrm{T}}\mathbf{v} =$$

$$\left(\sum_i \mathbf{n}_i{}^{\mathrm{T}}\right)\mathbf{v} = \sum_i \begin{vmatrix} \mathbf{v}_0^{t_i} & \mathbf{v}_1^{t_i} & \mathbf{v}_2^{t_i} \end{vmatrix} \tag{3}$$

where $\mathbf{n}_i$ is the outward normal vector of triangle $t_i$, with magnitude twice the area of $t_i$. It is clear that this local preservation of volume also implies global preservation as we have accounted for all triangles changed by the edge collapse. Equation 3 is a linear equality that constrains the solution $\mathbf{v}$ to a plane. Note that if the surface is locally non-orientable, folds over itself, or is otherwise geometrically or topologically degenerate, $\sum_i \mathbf{n}_i$ may be zero, in which case we discard the constraint. Since volume preservation only restricts $\mathbf{v}$ to a plane, we can place additional constraints on its final position.

### 4.2.2 Boundary Preservation

Analogous to volume preservation, our algorithm preserves the shape of surface boundaries in models that are not closed. For a planar boundary, we attempt to preserve the area enclosed by the boundary, which is the 2D equivalent of preserving the volume in 3D. Rather than using signed changes in volume, boundary preservation involves operations with signed changes in area. Thus, for a planar boundary, we set

$$\sum_i A\left((v, v_0^{e_i}, v_1^{e_i})\right) = \|\sum_i \frac{1}{2}(\mathbf{v} \times \mathbf{v}_0^{e_i} + \mathbf{v}_0^{e_i} \times \mathbf{v}_1^{e_i} + \mathbf{v}_1^{e_i} \times \mathbf{v})\| = 0 \tag{4}$$

where each term in the sum is a vector orthogonal to the boundary plane, with magnitude equal to the change in area associated with the corresponding edge. The vector direction determines the sign of the change. Figure 3 illustrates the changes in area for the boundary $\partial\lceil\lfloor e\rfloor\rceil = \{\vec{e}_0, \vec{e}_1, \vec{e}_2\}$ when the edge $\vec{e} = \vec{e}_1$ is collapsed to $v$.

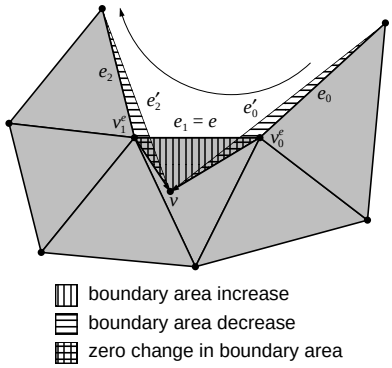



Figure 3: Collapsing a boundary edge $e$. The sum of signed areas of the hatched triangles is zero. The arc indicates the orientation of the boundary edges $\{\vec{e}_i\}$.

In general, surface boundaries are not planar, however, and we are forced to revise the notion of boundary area. A reasonable way of doing this is to relax the requirement that the terms in Equation 4 be parallel, and simply express each change in area as having a direction—instead of a binary sign—in addition to magnitude. Equation 4 generally has no solution for non-planar boundaries, however, so we have chosen to minimize the magnitude of the sum of directed area vectors, noting that the magnitude of the residual vector is a measure of how faithfully the boundary "area" has been preserved. Thus, we seek to minimize

$$\begin{aligned} & \|\sum_i \frac{1}{2}(\mathbf{v}\times\mathbf{v}_0^{e_i} + \mathbf{v}_0^{e_i}\times\mathbf{v}_1^{e_i} + \mathbf{v}_1^{e_i}\times\mathbf{v})\|^2 \\ = \;& \frac{1}{4}\|\mathbf{v}\times\sum_i(\mathbf{v}_1^{e_i}-\mathbf{v}_0^{e_i}) + \sum_i(\mathbf{v}_1^{e_i}\times\mathbf{v}_0^{e_i})\|^2 \\ = \;& \frac{1}{4}\|\mathbf{v}\times\sum_i \mathbf{e}_{1i} + \sum \mathbf{e}_{2i}\|^2 \\ = \;& \frac{1}{4}\|\mathbf{v}\times\mathbf{e}_1 + \mathbf{e}_2\|^2 \end{aligned}$$

where $\mathbf{e}_{1i} = \mathbf{v}_1^{e_i} - \mathbf{v}_0^{e_i}$, $\mathbf{e}_{2i} = \mathbf{v}_1^{e_i}\times\mathbf{v}_0^{e_i}$, and $\mathbf{e}_3 = \sum_i \mathbf{e}_{1i}\times\sum_i \mathbf{e}_{2i} = \mathbf{e}_1\times\mathbf{e}_2$. The solution space of this optimization problem is the intersection of two planes, defined by

$$\mathbf{e}_1{}^{\mathrm{T}}\mathbf{e}_1\mathbf{e}_3{}^{\mathrm{T}}\mathbf{v} + \mathbf{e}_3{}^{\mathrm{T}}\mathbf{e}_3 = 0 \quad (5)$$

$$(\mathbf{e}_1\times\mathbf{e}_3)^{\mathrm{T}}\mathbf{v} = 0 \quad (6)$$

Each of these constraints is added to $\mathbf{A}$ (in no particular order) subject to the compatibility rules mentioned above. Recall that these two constraints are only used when $\partial\lceil\lfloor e\rfloor\rceil$ is non-empty.

### 4.2.3 Volume Optimization

The remaining constraint methods are all optimization problems that alone yield single solutions in the unconstrained, non-degenerate case. Because prior constraints may exist, these methods involve minimization of some objective function $f(e,\mathbf{v})$ subject to zero, one, or two linear equality constraints $(\mathbf{A},\mathbf{b})$. In addition, these objective functions are all quadratic, and we are faced with a *quadratic programming problem*. Fortunately, all objective functions discussed here can be reduced to a certain form for which a simple method for finding $\mathbf{v}$ exists. These functions can all be written in the form

$$f(e,\mathbf{v}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{H}\mathbf{v} + \mathbf{c}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}k \quad (7)$$

where $\mathbf{H}$ is the symmetric Hessian, $\mathbf{H}\mathbf{v}+\mathbf{c}$ is the gradient of $f$, and $k$ is a constant. Note that Garland and Heckbert use a function of this form, but their constraints are derived in quite a different manner than ours [7]. Given $n$ constraints $(\mathbf{A},\mathbf{b})$, let the columns of $\mathbf{Z}$ be a basis in $\Re^3$, with the first $n$ equal to $\mathbf{A}^{\mathrm{T}}$. The remaining $3-n$ columns of $\mathbf{Z}$ are made orthogonal to the vectors $\mathbf{a}_i$. Then the additional $3-n$ constraints are

$$\mathbf{I}_{(3-n,3)}\mathbf{Z}^{-1}(\mathbf{H}\mathbf{v}+\mathbf{c}) = \mathbf{0} \quad (8)$$

where $\mathbf{I}_{(3-n,3)}$ is the $3-n$ by 3 submatrix formed by removing the top $n$ rows from the identity matrix $\mathbf{I}$. As above, these additional constraints are added provided they satisfy the compatibility rules.

As described above, the volume is preserved by setting the sum of *signed* tetrahedral volumes to zero, which leaves an entire plane of candidate vertices. To further constrain the vertex position, we also attempt to minimize the *unsigned* volume of each individual tetrahedron, which is a measure of the local surface error for each corresponding triangle in $\lceil\lceil\lfloor e\rfloor\rceil\rceil$. To minimize these errors, we find the minimum of

$$f_V(e,\mathbf{v}) = \sum_i V\left((v, v_0^{t_i}, v_1^{t_i}, v_2^{t_i})\right)^2$$

After expanding this and doing some algebra, we have

$$\begin{aligned} f_V(e,\mathbf{v}) &= \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{H}_V\mathbf{v} + \mathbf{c}_V{}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}k_V \\ &= \frac{1}{18}\left[\frac{1}{2}\mathbf{v}^{\mathrm{T}}\left(\sum_i \mathbf{n}_i\mathbf{n}_i{}^{\mathrm{T}}\right)\mathbf{v} + \right. \\ &\quad \left(-\sum_i \begin{vmatrix} \mathbf{v}_0^{t_i} & \mathbf{v}_1^{t_i} & \mathbf{v}_2^{t_i} \\ \end{vmatrix}\mathbf{n}_i{}^{\mathrm{T}}\right)\mathbf{v} + \\ &\quad \left.\frac{1}{2}\left(\sum_i \begin{vmatrix} \mathbf{v}_0^{t_i} & \mathbf{v}_1^{t_i} & \mathbf{v}_2^{t_i} \end{vmatrix}^2\right)\right] \end{aligned} \quad (9)$$

which is of the same form as Equation 7, and Equation 8 can be used to find the remaining constraints. Figure 4 illustrates how $v$ is found as the intersection of three planes, two of which are obtained from volume optimization.

If the vertices $\lfloor\lceil\lfloor e\rfloor\rceil\rfloor$ are all coplanar, the volume optimization yields infinitely many solutions as each tetrahedral volume is zero. In the case when the vertices are nearly coplanar, Equation 9 results in constraints that are not $\alpha$-compatible with prior constraints, and we assume that the surface errors are small enough that optimization of other aspects of the mesh are more important, for example maximizing the aspect ratio of the affected triangles. Additional optimization is discussed in the next two subsections.

### 4.2.4 Boundary Optimization

For boundary optimization, we use a 2D analogy to the above mentioned volume optimization. That is, we minimize the sum of squared areas described in Section 4.2.2:

$$f_B(e,\mathbf{v}) = L(e)^2\sum_i A\left((v, v_0^{e_i}, v_1^{e_i})\right)^2$$

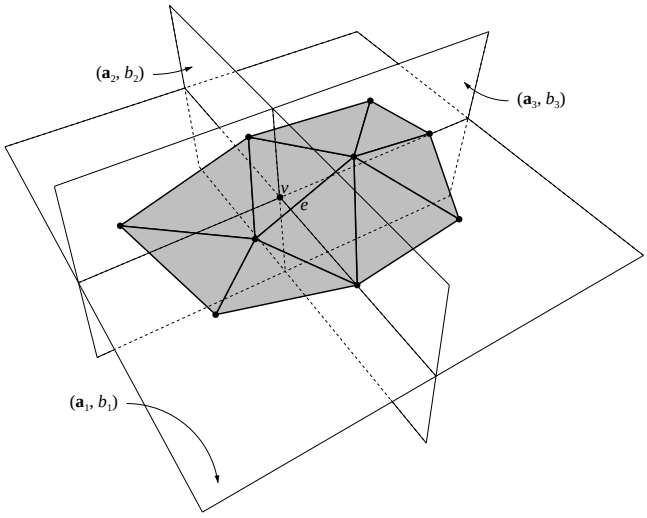



Figure 4: The optimal vertex $v$ expressed as the intersection of three planes; $(\mathbf{a}_1, b_1)$ ensures that the volume is preserved, while $(\mathbf{a}_2, b_2)$ and $(\mathbf{a}_3, b_3)$ correspond to the constraints due to volume optimization.

where $L(e)^2$ is the squared length of the edge $e$. The reason for including this non-negative constant will be explained later. It should be clear that it has no effect on where the minimum of $f_B$ occurs. The above equation reduces to

$$\begin{aligned} f_B(e,\mathbf{v}) &= \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{H}_B\mathbf{v} + {\mathbf{c}_B}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}k_B \\ &= \frac{L(e)^2}{2}\left[\frac{1}{2}\mathbf{v}^{\mathrm{T}}\left(\sum_i (\times\mathbf{e}_{1i})(\mathbf{e}_{1i}\times)\right)\mathbf{v} + \right. \\ &\quad \left(\sum_i (\mathbf{e}_{1i}\times\mathbf{e}_{2i})^{\mathrm{T}}\right)\mathbf{v} + \\ &\quad \left.\frac{1}{2}\left(\sum_i {\mathbf{e}_{2i}}^{\mathrm{T}}\mathbf{e}_{2i}\right)\right] \end{aligned} \tag{10}$$

where the matrix $(\mathbf{v}\times)$ is defined as

$$(\mathbf{v}\times) = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix}$$

and $(\times\mathbf{v}) = (\mathbf{v}\times)^{\mathrm{T}}$. As before, Equation 8 is used to further constrain $\mathbf{v}$. Figure 5 shows constraints associated with boundary preservation and optimization.

#### 4.2.5 Triangle Shape Optimization

Under certain circumstances, the constraints discussed previously are not all compatible and do not yield a single solution, and further optimization can be employed. Typically, these cases occur when the objective functions are constant, e.g. when the vertices $\lfloor\lceil\lfloor e\rfloor\rceil\rfloor$ are all coplanar, or when the vertices $\lfloor\partial\lceil\lfloor e\rfloor\rceil\rfloor$ are collinear. In these circumstances, we have decided to optimize the shape of the triangles $\lceil\lceil v\rceil\rceil$ such that equilateral triangles are preferred over long and skinny ones. Elongated triangles can introduce unwanted shading discontinuities and may slow down some rendering methods. As a measure of triangle shape quality, we have chosen the following expression:

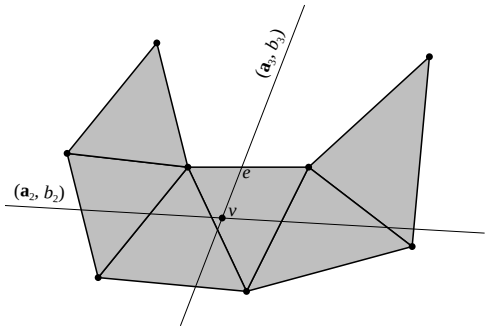



Figure 5: The optimal vertex $v$ for a collapsed boundary edge $e$. $(\mathbf{a}_2, b_2)$ is the set of vertices for which the boundary area is preserved; boundary optimization yields $(\mathbf{a}_3, b_3)$. The volume preservation constraint $(\mathbf{a}_1, b_1)$, not shown here, is parallel to the plane of the figure.

$$f_S(e,\mathbf{v}) = \sum_i L((v,v_i))^2$$

which is the sum of squared lengths of the edges incident upon $v$. By minimizing $f_S$, we maximize the quality of the resulting triangles. Because we know that the surface is locally planar (or nearly planar) the choice of $\mathbf{v}$ does not (significantly) alter the sum of areas of the incident triangles after the edge is collapsed. Thus, by minimizing the above edge lengths, we ensure that the area to perimeter ratios of the resulting triangles are maximized. The objective function $f_S$ can be written as

$$\begin{aligned} f_S(e,\mathbf{v}) &= \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{H}_S\mathbf{v} + {\mathbf{c}_S}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}k_S \\ &= 2\left[\frac{1}{2}\mathbf{v}^{\mathrm{T}}\left(\sum_i \mathbf{I}\right)\mathbf{v} + \left(-\sum_i {\mathbf{v}_i}^{\mathrm{T}}\right)\mathbf{v} + \frac{1}{2}\left(\sum_i {\mathbf{v}_i}^{\mathrm{T}}\mathbf{v}_i\right)\right] \end{aligned} \tag{11}$$

It is fairly easy to show that $f_S$ increases with the squared distance of $\mathbf{v}$ to the centroid of $\{\mathbf{v}_i\}$.

### 4.3 Edge Costs

Given an optimal vertex position for an edge collapse, we need to determine the cost of collapsing the edge. The term "optimal" needs to be put in context, however. By defining the edge cost in terms of the objective functions that were minimized above, the vertex position is optimal with respect to the incurred cost of collapsing the edge. We have chosen to write the edge cost as a combination of the following terms:

$$\begin{aligned} c(e,\mathbf{v}) &= c_V f_V(e,\mathbf{v}) + c_B f_B(e,\mathbf{v}) \\ &= c_V \sum_i V\left((v, v_0^{t_i}, v_1^{t_i}, v_2^{t_i})\right)^2 + \\ &\quad c_B L(e)^2 \sum_i A\left((v, v_0^{e_i}, v_1^{e_i})\right)^2 \end{aligned} \tag{12}$$

That is, the cost is a weighted sum of the terms minimized in the volume and boundary optimization. The squared length of the edge $e$, $L(e)^2$, used in the boundary objective function ensures scale invariance and makes $f_B$ compatible with $f_V$. We have omitted the term for triangle shape optimization as it tends to penalize edges that otherwise have low values for $f_V$ and $f_B$. Recall that $f_S$ is used

in the optimization only when $f_V$ and $f_B$ are both close to zero, and serves as a last resort to constraining $\mathbf{v}$ when all other methods have failed. For the results presented in this paper, $c_V = c_B = 1$, which has given good results for all the models that we have tried.

### 4.4 Summary of Vertex Placement

Given an underconstrained solution for the vertex $\mathbf{v}$ that preserves both volume and boundary area, we choose $\mathbf{v}$ such that $c$ in Equation 12 is minimized. Notice that $c$ can be written in the form of Equation 7. This allows us to minimize the volume and boundary objective functions simultaneously, i.e. following volume and boundary preservation, we minimize

$$\begin{aligned} c(e,\mathbf{v}) \quad = \quad & \frac{1}{2}\mathbf{v}^{\mathrm{T}}(c_V\mathbf{H}_V + c_B\mathbf{H}_B)\mathbf{v} + \\ & (c_V\mathbf{c}_V + c_B\mathbf{c}_B)^{\mathrm{T}}\mathbf{v} + \\ & \frac{1}{2}(c_V k_V + c_B k_B) \end{aligned} \tag{13}$$

If the solution is still underconstrained, we employ triangle shape optimization. Thus, to find $\mathbf{v}$, constraints are added as follows:

| | *method* | *constraints* | *equation* |
|---|---|---|---|
| 1. | volume preservation | $\leq 1$ | 3 |
| 2. | boundary preservation | $\leq 2$ | 5, 6 |
| 3. | volume/boundary optimization | $\leq 3$ | 8, 13 |
| 4. | triangle shape optimization | $\leq 3$ | 8, 11 |

The constraints presented in this section rely on the use of quadratic objective functions. These squared terms were introduced to eliminate square roots and to allow an efficient optimization procedure. However, one drawback of this approach is that it is sensitive to modifications of the mesh connectivity that don't affect the geometry, e.g. splitting a triangle in two affects the sums of squared areas and volumes associated with the triangle. Such cases can be detected and eliminated as a pre-processing step.

In rare cases, the solution to $\mathbf{v}$ remains underconstrained, at which point some additional criterion could be used to determine $\mathbf{v}$. We have simply chosen to reject such edges as edge collapse candidates.

## 5 RESULTS

### 5.1 Geometric Comparison Tool

In order to assess the quality of our simplification method, we created a number of simplified models and we have compared them to simplified models created using other published simplification techniques. We could have attempted to implement these algorithms from their published descriptions, but in doing so we might have introduced bias by not tuning the parameters of the other methods with enough care. Instead, we chose to compare our method to results taken from implementations by the authors of the methods. Doing so is not free from bias either because some simplification methods are not publically available, often due to commercial or intellectual property right restrictions, thus some techniques inevitably will not be represented in comparisons such as ours. We feel, however, that simplification techniques have become mature enough as a sub-area within computer graphics that researchers should at least make an attempt to assess the quality of their results.

We have chosen to use the *Metro* geometric comparison tool in order to measure differences between an original model and a simplified version of that polygonal model [3]. Metro accepts two polygonal models—an original and a simplified model—and computes the maximum and mean geometric errors of the simplified model with respect to the original. This is done by point sampling the simplified model uniformly, using Phong interpolation to estimate the surface normal at each sample, and intersecting the line defined by the point sample and its normal with the original model. Both the maximum and mean distance between the point samples and their corresponding intersections with the original are recorded.

We selected Metro for several reasons. First, we did not author this tool, and it is our hope that this eliminates one potential source of bias on our part. Second, Metro uses a point sampled distance measure that is quite different from the volume-based calculations that drive our simplification technique. Thus it is less likely that we have tuned our algorithm to Metro's distance measure. Third, it is publically available so that others may perform evaluations that can be matched with those presented here. Finally, we have some degree of confidence in Metro because the values that it returns for various models are a good match to our own visual impressions of the qualities of the different models.

### 5.2 Comparisons

We used two test models in our comparisons. One of these models is the Stanford Bunny, which is often used as a test object for simplification. It is also a good candidate object because it contains several regions with boundaries on its underside. The second object is a model of the bones in a human hand that was constructed for stereolithography (Plate 4a). The bones are joined by cylinders so that the model will be a single connected object after the physical model is created using stereolithography. At 650,000 triangles, this model allows us to see the performance of the simplification methods over a wide range of detail levels. For both of these models we produced simplified versions at eight different levels of detail, roughly halving the number of edges at each level. For each simplification method, we attempted to produce simplified models with the same number of edges, which takes into account both the number of vertices and triangles. Two versions of a model with surface boundaries that have the same number of triangles may have rather large differences in the number of vertices, and vice versa. Keeping $E \approx V + T$ the same for all methods ensures a more fair comparison.

We simplified both models using six different algorithms, including our own:

(1) Mesh Optimization [12]

(2) Progressive Meshes [13]

(3) Simplification Envelopes [4]

(4) JADE (vertex decimation) [2]

(5) QSlim (quadric error) [7]

(6) Memoryless (our method)

Methods 3 and 4 use vertex removal and the remaining four use edge collapse. All the methods besides our own use some form of geometric history to guide simplification, hence we use the name "Memoryless" for our method.

Figures 6 and 7 show the performances of the different algorithms on the bunny and hand models. As can be seen, QSlim is the fastest of the methods, followed by our method. All models were simplified on a four-processor, 195 MHz R10000 Silicon Graphics Onyx$^2$ machine with 1 GB of main memory.[1]

[1]The Progressive Mesh models were generated on a one-processor, 195 MHz R10000 Silicon Graphics Octane. For our purposes, these machines are comparable in performance.

Figures 8 and 10 show the mean geometric deviations between the original and the simplified models. The scale of the logarithmic *y*-axis is 1,000 times the ratio of the error and the bounding box diagonal. It can be seen from these graphs that the models with the best mean geometric errors were produced by our method and by Mesh Optimization. These two algorithms give nearly identical mean geometric errors. (Recall, however, that our method is orders of magnitude faster than Mesh Optimization.) There is remarkable consistency in the data points of these graphs. Each algorithm gives a nearly straight line in log(edge count) versus log(mean error). Also, the relative behaviors of the different methods are the same for both models across all levels of detail. The consistency of these results gives us confidence in the reliability of these Metro measurements.

Figures 9 and 11 show the maximum geometric deviations between the original and simplified models. There appears to be little consistency in the maximum errors shown in these graphs. We can think of two possible causes for this. One is that the relationship between number of edges in a model and the maximum error a particular algorithm gives is not consistent across models and levels of detail. Another possibility is that Metro's measurement of maximum error is inaccurate.

Plates 1a through 1f show the bunny models created by the six different simplification techniques. Each model has approximately 2,000 edges. All of these models appear to be reasonable low-resolution versions of the original model (Plate 3a). We have found that people tend to focus their attention on facial features of the bunny such as the eyes and nose, and a chance polygon at such places can dramatically alter the casual observer's impression of these models. We note that for these and other models, the two vertex removal methods (Plates 1c and 1d) seem to produce more sliver triangles than the other methods. This is probably due to the restriction of never being able to move vertex positions.

Plates 2a through 2f show the underside of the same bunny models shown in Plate 1. Plate 3b shows the original model's underside. Mesh optimization (Plate 2a) appears to have distorted the boundary positions. Both JADE and QSlim (Plates 2d and 2e) used a large number of rather thin triangles in order to maintain the boundaries, although it is possible that parameter tuning could change this. Progressive meshes, envelopes, and our memoryless technique yield similar results on the boundaries.

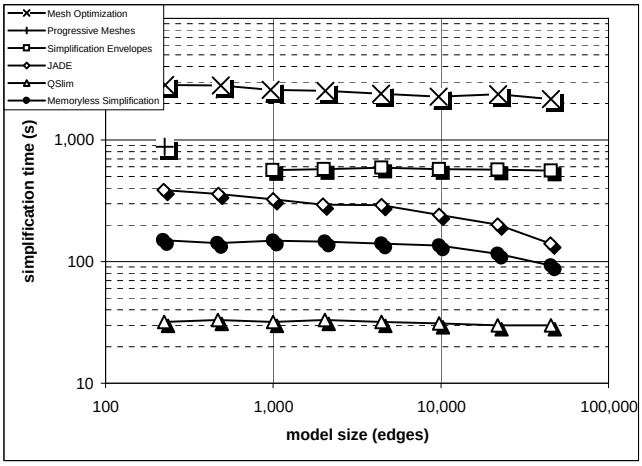



Figure 6: Simplification time for bunny model.

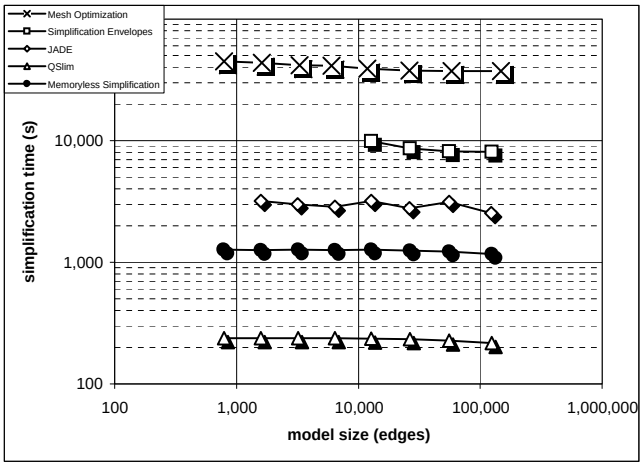



Figure 7: Simplification time for hand model.

### 5.3 Additional Results

Plates 4 through 6 show additional results from our algorithm. Plate 4b is a 4,266 triangle version of the hand model created using our method. Plates 5a and 5b are of a turbine blade. This turbine blade model consists of 1.8 million triangles, and contains very fine interior detail. Simplifying this model is challenging due to its sheer size and its topological complexity, with a large number of tiny holes and a very noisy surface. This model also has many sharp edges, and so provides a different challenge than the rounded features of the bunny and hand models. The model in Plate 5b is a 13,332 triangle version of this model that was created using our algorithm in an hour and fifteen minutes. The only other algorithm that we were able to use to simplify this model was QSlim, which required nearly ten hours due to disk thrashing from lack of memory on the 1 GB machine.

Plate 6a is a model of a range scanned dragon with 870,000 triangles. The result of simplifying this model using our algorithm is shown in Plate 6b. Notice that when simplified to 10,922 triangles, the model still retains the scaly texture on the body as well as features such as the teeth and the fins on its back.

There is no one “best” algorithm for simplifying models. Each of the methods used above has a niche for which it is well suited, depending on speed and memory requirements, the types of models being used, and the geometric and visual requirements of the application. We believe that the memoryless method that we have presented in this paper should be particularly useful for applications that require the simplification of large models or those that require a low mean geometric error. Mean geometric error is a good indicator of visual fidelity to the original model. Thus we believe that models created using our method are suitable for applications where the overall visual impression is important, such as vehicle simulators, building walkthroughs, and educational software. Applications requiring absolute guarantees on error bounds, however, should instead use a method that provides absolute distance bounds. We plan to make our code available on the Web so that others can use it for their applications or may incorporate aspects of our method in other simplification systems.

## 6 FUTURE WORK

There are several possibilities future directions for this work. A straightforward extension to our method would be to allow any two vertices to be merged, whether or not they share a common edge. This would allow topology changes as demonstrated in [7, 16].

Our decision procedures for edge collapse could also be used in combination with other frameworks for simplification. If, for example, maximum error is a high priority, then our edge collapse operation could be merged with any of a number of approaches that track maximum error. This could produce a method that would retain the mean error behavior of our method but also would bound the maximum error. As another example, our method would be a memory efficient way to create the edge collapse history used in a view-dependent simplification framework [14, 15]. Finally, we have demonstrated that volume preservation and per-triangle volume optimization is an effective measure of deviation from a surface. It is likely that a still more accurate measure of volume deviation could be performed if each triangle or vertex carried with it additional information.

## Acknowledgement

We would like to thank the Stanford Computer Graphics Laboratory for providing the bunny model, the authors of the Visualization Toolkit for the turbine blade model, Hugues Hoppe for providing progressive mesh models of the bunny and an implementation of "Mesh Optimization", and the people at CMU, CNUCE, and UNC for making their simplification tools available.

## References

[1] BAJAJ, C. and SCHIKORE, D. R. Error-Bounded Reduction of Triangle Meshes with Multivariate Data. In *Visual Data Exploration and Analysis III* Proceedings, SPIE 2656, 1996, pp. 34–45.

[2] CIAMPALINI, A., CIGNONI, P., MONTANI, C., and SCOPIGNO, R. Multiresolution Decimation Based on Global Error. *The Visual Computer*, Springer International, 13(5), 1997, pp. 228–246.

[3] CIGNONI, P., ROCCHINI, C., and SCOPIGNO, R. Metro: Measuring Error on Simplified Surfaces. Technical Report B4–01–01–96, Istituto I. E. I.–C. N. R., Pisa, Italy, January 1996.

[4] COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., and WRIGHT, W. Simplification Envelopes. Proceedings of SIGGRAPH 96. In *Computer Graphics* Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 119–128.

[5] COHEN, J., MANOCHA, D., and OLANO, M. Simplifying Polygonal Models Using Successive Mappings. In *IEEE Visualization '97* Proceedings, October 1997, pp. 395–402.

[6] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., and STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. Proceedings of SIGGRAPH 95. In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 173–182.

[7] GARLAND, M. and HECKBERT, P. S. Surface Simplification using Quadric Error Metrics. Proceedings of SIGGRAPH 97. In *Computer Graphics* Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 209–216.

[8] GIENG, T. S., HAMANN, B., JOY, K. I., SCHUSSMAN, G. L., and TROTTS, I. J. Smooth Hierarchical Surface Triangulations. In *IEEE Visualization '97* Proceedings, October 1997, pp. 379–386.

[9] GUÉZIEC, A. Surface Simplification Inside a Tolerance Volume. *Second Annual International Symposium on Medical Robotics and Computer Aided Surgery*, November 1995, pp. 132–139.

[10] HAMANN, B. A Data Reduction Scheme for Triangulated Surfaces. *Computer Aided Geometric Design*, 11(2), April 1994, pp. 197–214.

[11] HE, T., HONG, L., KAUFMAN, A., VARSHNEY, A., and WANG, S. Voxel Based Object Simplification. In *IEEE Visualization '95* Proceedings, October 1995, pp. 296–303.

[12] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., and STUETZLE, W. Mesh Optimization. Proceedings of SIGGRAPH 93. In *Computer Graphics* Proceedings, Annual Conference Series, 1993, ACM SIGGRAPH, pp. 19–26.

[13] HOPPE, H.. Progressive Meshes. Proceedings of SIGGRAPH 96. In *Computer Graphics* Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 99–108.

[14] HOPPE, H.. View-Dependent Refinement of Progressive Meshes Proceedings of SIGGRAPH 97. In *Computer Graphics* Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 189–198.

[15] LUEBKE, D. and ERIKSON, C. View-Dependent Simplification of Arbitrary Polygonal Environments Proceedings of SIGGRAPH 97. In *Computer Graphics* Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 199–208.

[16] POPOVIC, J. and HOPPE, H.. Progressive Simplicial Complexes. Proceedings of SIGGRAPH 97. In *Computer Graphics* Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 217–224.

[17] RENZE, K. J. and OLIVER, J. H. Generalized Surface and Volume Decimation for Unstructured Tessellated Domains. In *VRAIS '96* Proceedings, 1996, pp. 111–121.

[18] RONFARD, R. and ROSSIGNAC, J. Full-Range Approximation of Triangulated Polyhedra. Proceedings of Eurographics 96. In *Computer Graphics Forum*, 15(3), August 1996, pp. 67–76.

[19] ROSSIGNAC, J. and BORREL, P. Multi-Resolution 3D Approximations for Rendering Complex Scenes. In *Modeling in Computer Graphics*, edited by B. Falcidieno and T. L. Kuuni, Springer-Verlag, 1993, pp. 455–465.

[20] SCHROEDER, W. J., ZARGE, J. A., and LORENSEN, W. E. Decimation of Triangle Meshes. Proceedings of SIGGRAPH 92. In *Computer Graphics* 26(2) (July 1992), pp. 65–70.

[21] SCHROEDER, W. J. A Topology Modifying Progressive Decimation Algorithm. In *IEEE Visualization '97* Proceedings, October 1997, pp. 205–212.

[22] TURK, G. Re-Tiling Polygonal Surfaces. Proceedings of SIGGRAPH 92. In *Computer Graphics* 26(2) (July 1992), pp. 55–64.

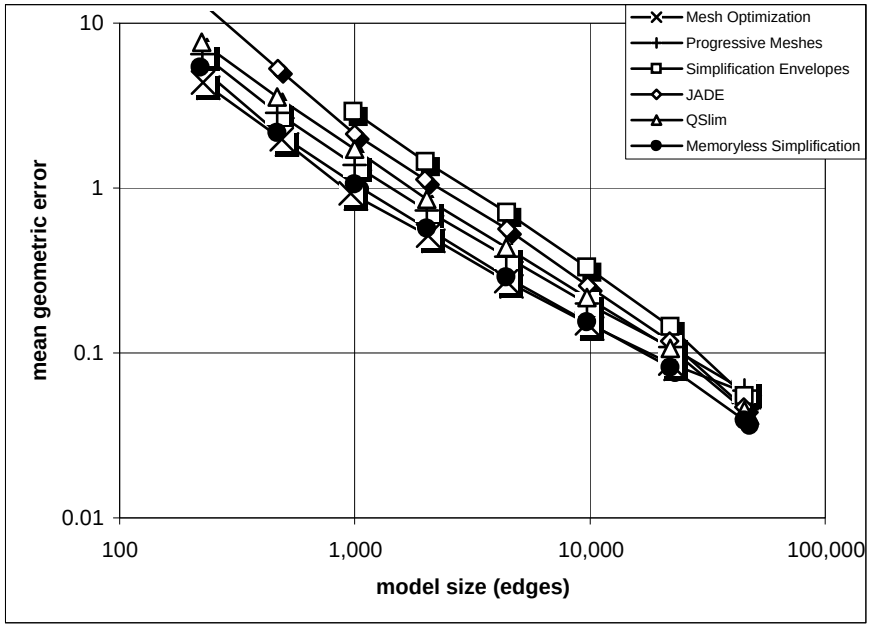



Figure 8: Mean geometric error for bunny model

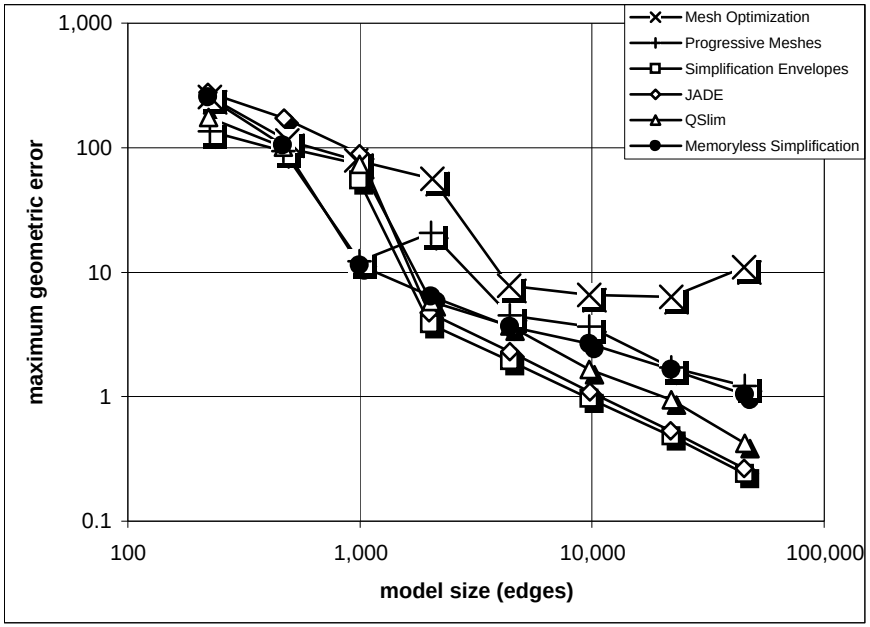



Figure 9: Maximum geometric error for bunny model

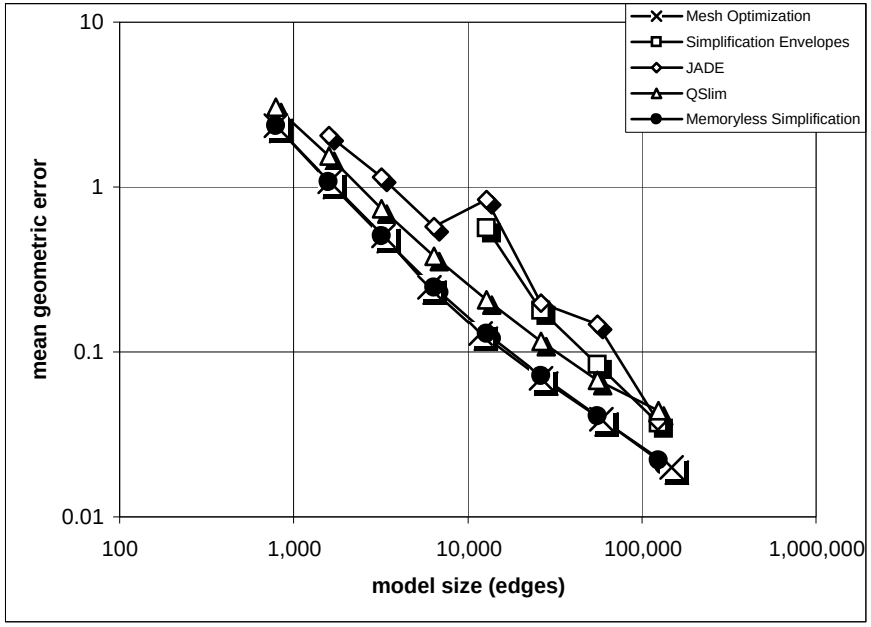



Figure 10: Mean geometric error for hand model

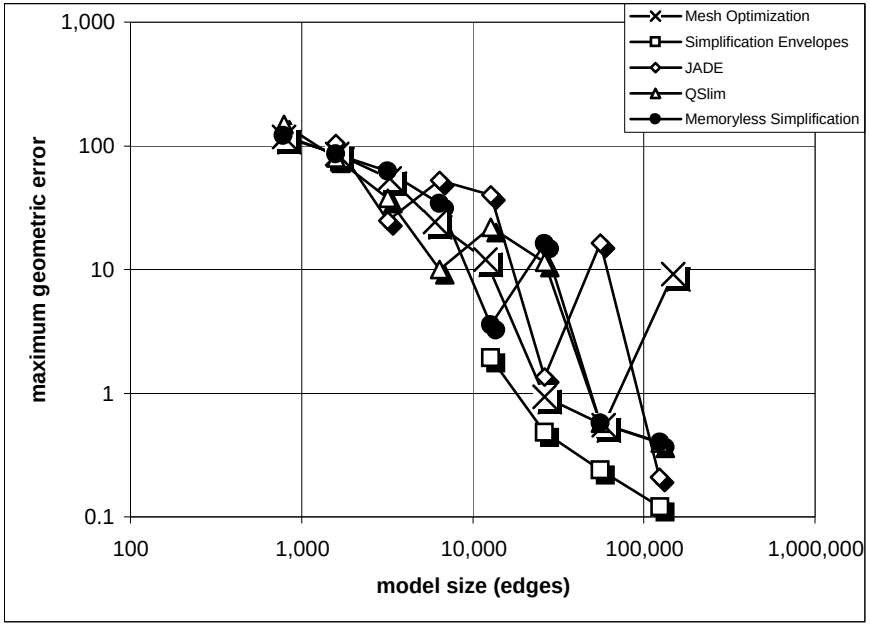
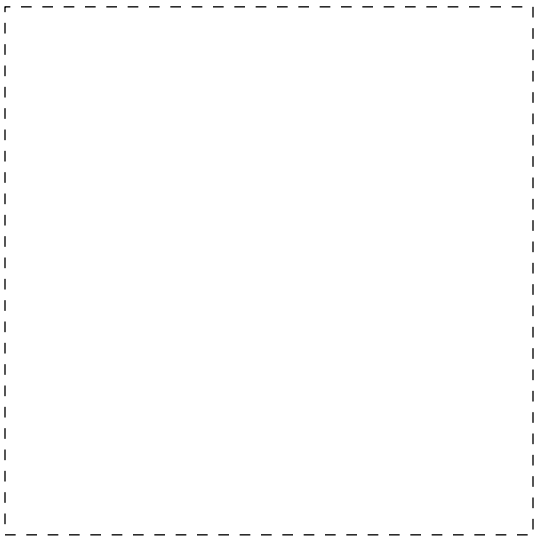
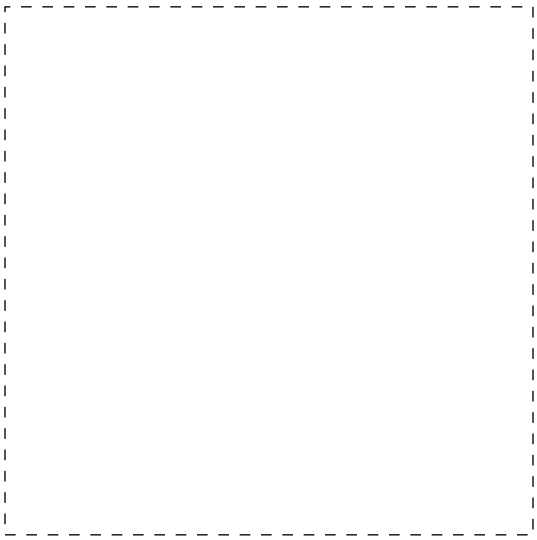


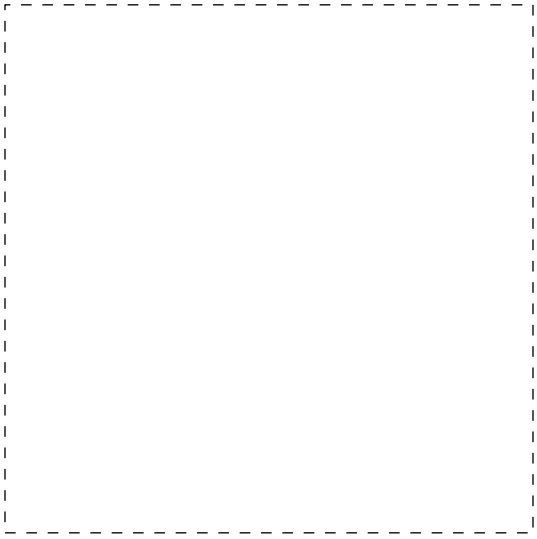

Figure 11: Maximum geometric error for hand model



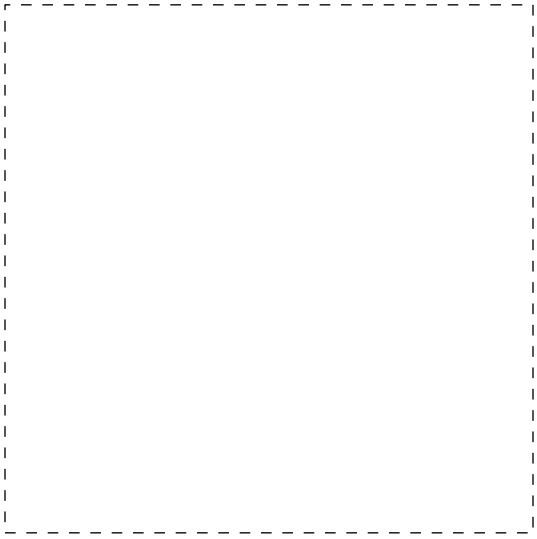
1a. Mesh Optimization



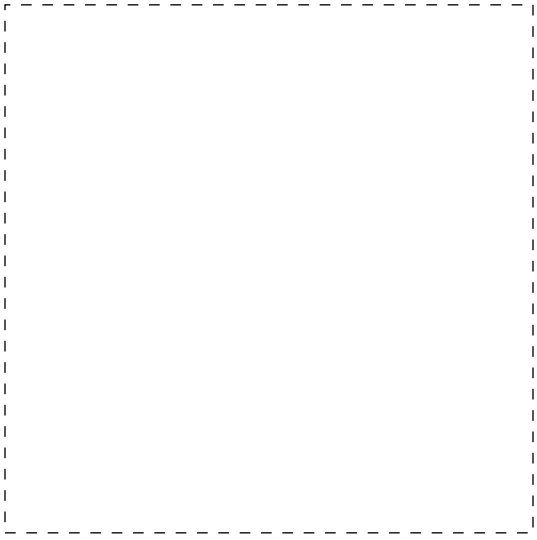
1b. Progressive Meshes



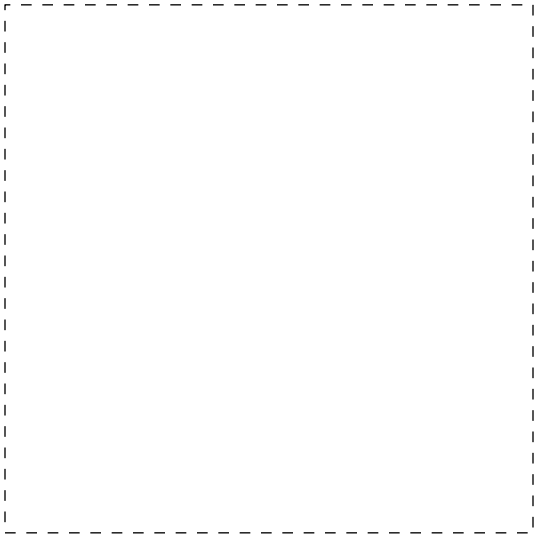
1c. Simplification Envelopes



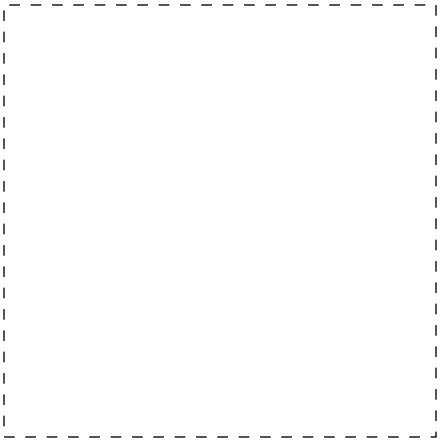
1d. JADE



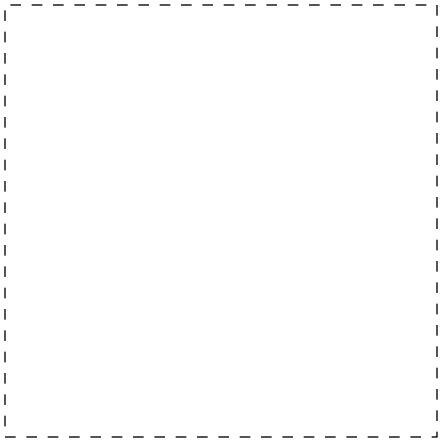
1e. QSlim



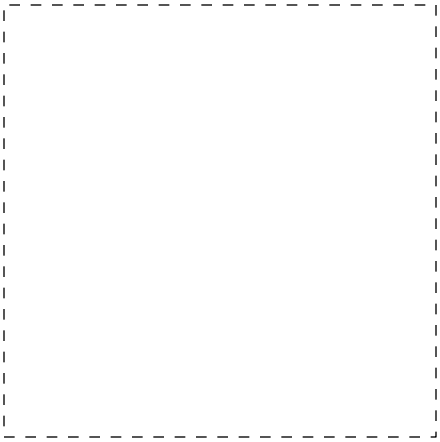
1f. Memoryless Simplification



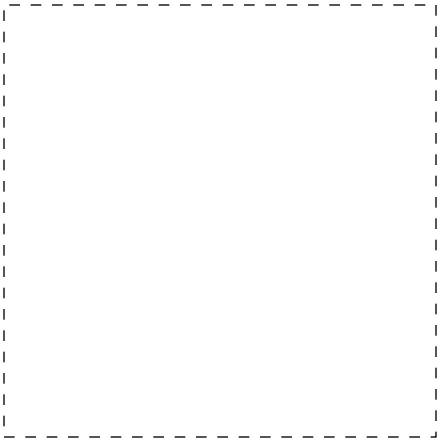
2a. Mesh Optimization
($V$ = 701 $E$ = 2,046 $T$ = 1,342)



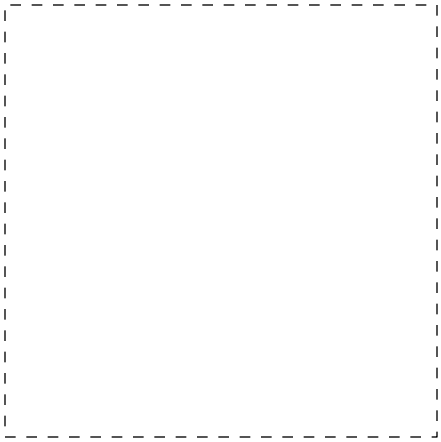
2b. Progressive Meshes
($V$ = 686 $E$ = 2,027 $T$ = 1,338)



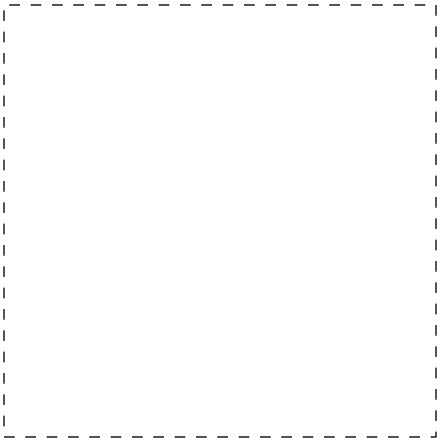
2c. Simplification Envelopes
($V$ = 686 $E$ = 2,003 $T$ = 1,314)



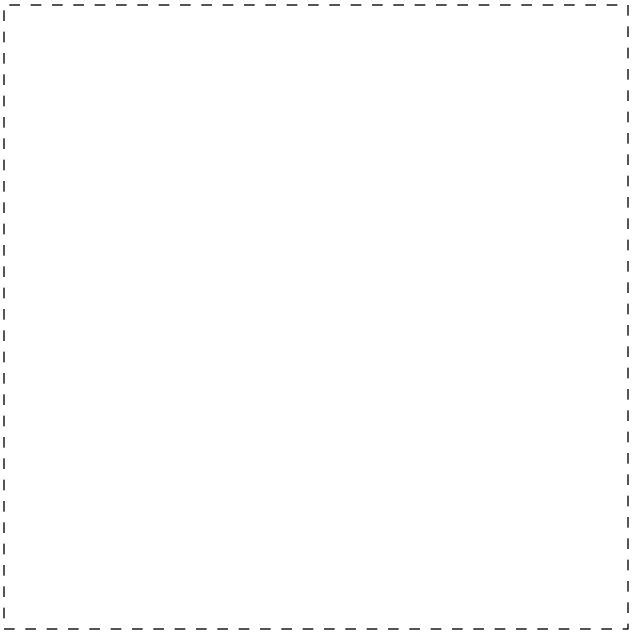
2d. JADE
($V$ = 691 $E$ = 1,983 $T$ = 1,289)



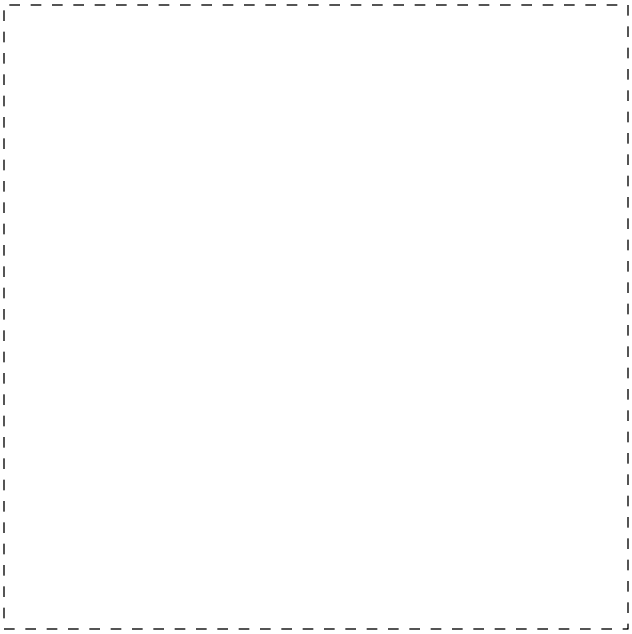
2e. QSlim
($V$ = 711 $E$ = 2,027 $T$ = 1,313)



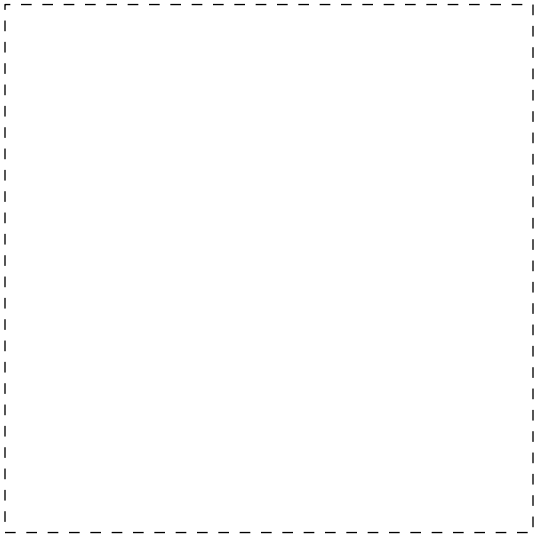
2f. Memoryless Simplification
($V$ = 687 $E$ = 2,027 $T$ = 1,337)



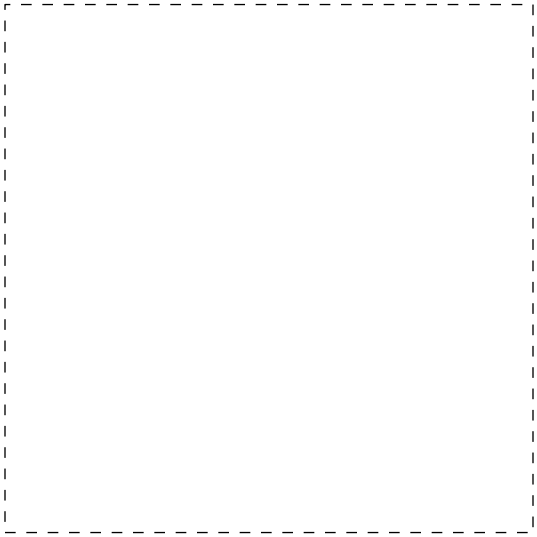
3a. Original bunny model
($V$ = 34,834 $E$ = 104,288 $T$ = 69,451)



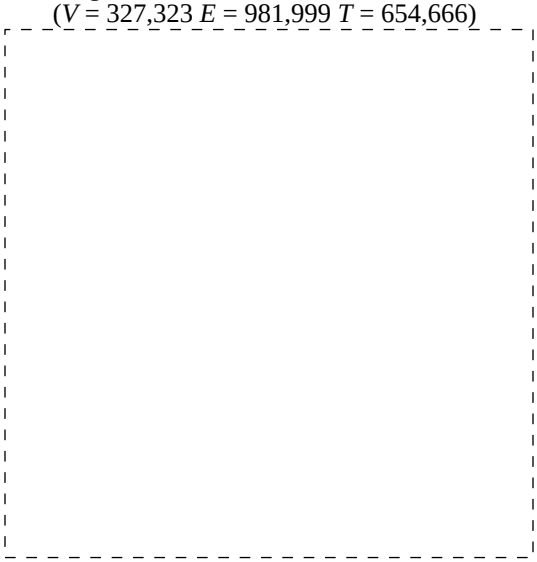
3b. Base of bunny model with surface boundaries



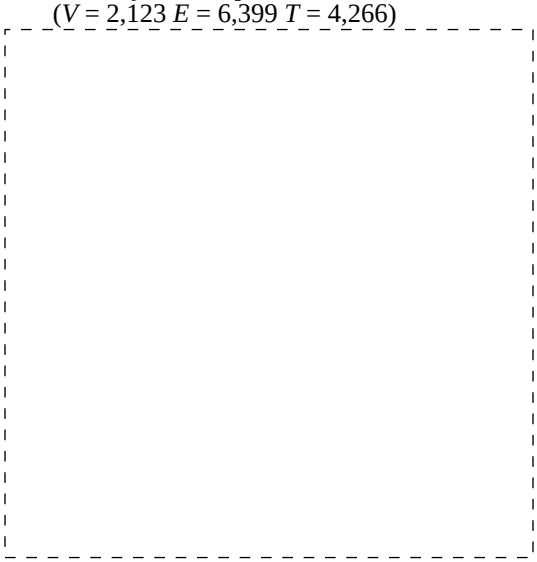
4a. Original hand model
($V$ = 327,323 $E$ = 981,999 $T$ = 654,666)



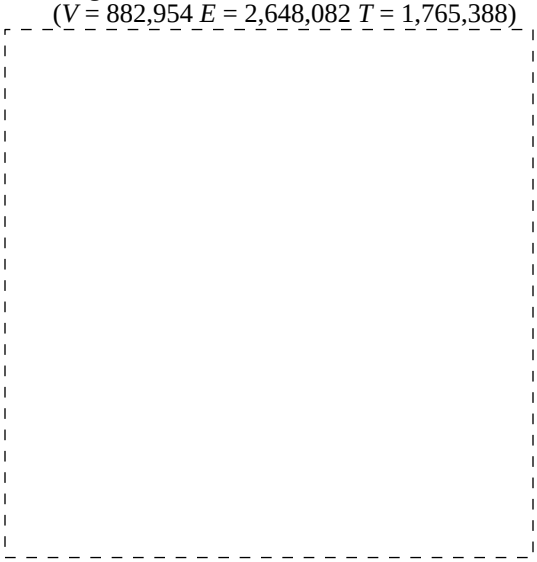
4b. Memoryless Simplification
($V$ = 2,123 $E$ = 6,399 $T$ = 4,266)



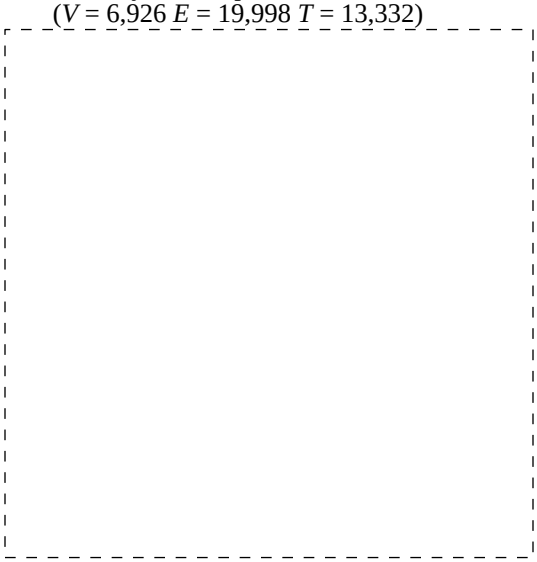
5a. Original turbine blade model
($V$ = 882,954 $E$ = 2,648,082 $T$ = 1,765,388)



5b. Memoryless Simplification
($V$ = 6,926 $E$ = 19,998 $T$ = 13,332)



6a. Original dragon model
($V$ = 435,545 $E$ = 1,306,959 $T$ = 871,306)



6b. Memoryless Simplification
($V$ = 5,353 $E$ = 16,383 $T$ = 10,922)