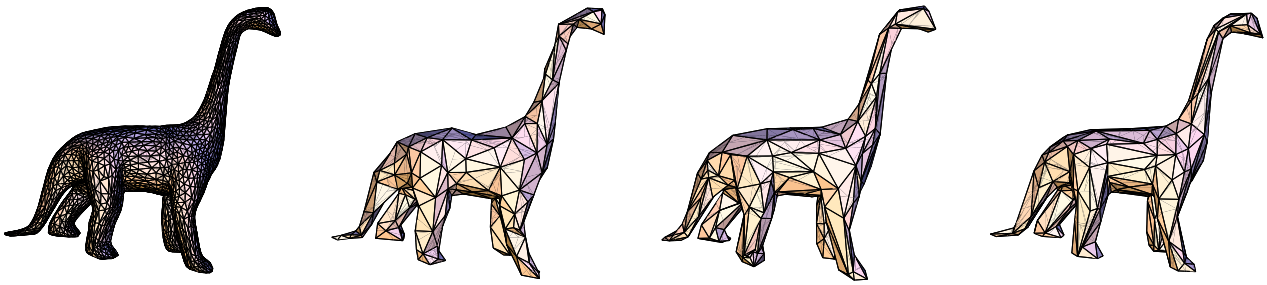
Stellar mesh simplification using probabilistic optimization

ANTÔNIO WILSON VIEIRA[1,2], THOMAS LEWINER[1,3], LUIZ VELHO[4], HÉLIO LOPES[1] AND GEOVAN TAVARES[1]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] CCET — Universidade de Montes Claros — Brazil
[3] Géométrica Project — INRIA – Sophia Antipolis — France
[4] Visgraf Project — IMPA — Rio de Janeiro — Brazil
{awilson, lopes, tavares, tomlew}@mat.puc--rio.br. lvelho@impa.br.

Abstract. This paper proposes the *Stellar Mesh Simplification* method, a fast implementation of the Four–Face Cluster algorithm. In this method, a probabilistic optimization heuristic substitutes the priority queue of the original method, which results in a 40% faster algorithm with the same order of distortion. It extends naturally to a progressive and/or multi–resolution scheme for combinatorial surfaces. This work also presents a simple way to encode the hierarchy of the resulting multi-resolution meshes. This work also focuses on important aspects for the development of a practical and robust implementation of this simplification technique, and on the analysis of the influence of the parameters.

Keywords: ***Computer Graphics. Computational Geometry and Object Modeling. Mesh simplification. Geometry processing. Randomized algorithms.***



(a) Original: 7400 faces. (b) Simpl.: 500 faces, $d = 4$. (c) Simpl.: 500 faces, $d = 8$. (d) Simpl.: 500 faces, $d = 12$.

Figure 1: *Simplification of the Dinosaur model with different random choice panel sizes.*

1 Introduction

The typical surface models handled by contemporary Computer Graphics applications have millions of triangles. Mesh Simplification has emerged as a critical step for handling such huge meshes. On one hand, there is an evident need for removing redundancies on meshes obtained by surface reconstruction algorithms, such as Marching Cubes [10]. On the other hand, meshes with a high level of detail even without redundancies could be extremely expensive to render, store or transmit [6]. In both cases, an efficient simplification process would generate simpler models with lower computational costs [2]. This paper proposes the *Stellar Mesh Simplification* simplification scheme, a fast implementation of the Four–Face Cluster algorithm [15]. It uses a probabilistic optimization heuristic based on the Multiple-Choice technique [16] that substitutes the priority queue of the original algorithm. The result is a 40% faster algorithm with the same order of distortion. It also induces naturally a hierarchical multi–resolution structure that can be used in a wide range of applications [13].

Prior work. Many efficient mesh simplification methods are based on local topological operators [4, 15, 16]. The most common is the *Edge–Collapse*, which consists in contracting the two vertices of an edge onto a unique vertex, eliminating its two incident faces. The inverse of the Edge–Collapse operation is called *Vertex–Split*. Simplification schemes construct a sequence of edges to be collapsed, resulting in a hierarchy of meshes of decreasing size. Traversing this sequence in a reverse order, with *Vertex–Split* operations, it is possible to recover details from the mesh of minimal size (base mesh). Although topological tests can be performed to preserve the topology of the surface during the simplification, its geometry will be distorted. To minimize this distortion, an energy

Preprint MAT. 10/04, communicated on March 1^{st}, 2004 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.

function measuring the quality of the mesh is used to guide the mesh simplification. An example of such function is the quadric error metric [4]. For those algorithms [4, 15], the priority queue is a natural data structure to store the order of the edges to be simplified, since it allows a variety of operations (inclusion, access and removal of the largest, etc.) to be efficiently performed.

Although the priority queue is efficient, it is necessary to build it before starting the simplification process. Moreover, at each step of the process a significant amount of time is consumed not only to reprocess the geometrical change for all edges involved in the local operation, but also to update their position in the priority queue. In order to accelerate this process, Wu and Kobbelt [16] presented a technique called Multiple–Choice based on probabilistic optimization where there is no use of a priority queue, but the edge to be contracted is chosen among a small number of randomly selected edges.

In [15], Velho proposed a mesh simplification method, called Four–Face cluster (FFC), which produces a sequence of *Edge–Weld* operations intercalated with *Edge–Flip* operations. Edge–Weld and Edge–Flip are two example of topological operators based on the Stellar Theory [1, 11]. The Edge–Flip swaps an internal edge of a 2 face cluster. The Edge–Weld removes a valence 4 vertex from a four–face cluster and replaces it by the internal edge of a two–face cluster. Edge–flips are required to change the valence of a vertex to 4. Stellar theory proved that those two operators forms a complete set for changing the connectivity of the mesh without modifying its topology. Differently from the other algorithms cited above, the FFC computes the energy function on the vertices, and not on the edges.

Contributions. This work enhances the Four–Face Clusters algorithm by substituting the priority queue by the Multiple-Choice technique. The result is an algorithm that is about 40% faster than the original one and that requires less memory. This algorithm is called *Stellar Mesh Simplification* (SMS), because it is based on the stellar operators Edge–Weld and Edge–Flip. To provide a formal definition for those operators, an introduction to Stellar Theory is given.

In this SMS implementation, the Corner–Table data structure [14] is adopted, which allows a concise and simple implementation of the algorithm. This paper also introduces a new scheme to represent the hierarchy of meshes obtained during the process of simplification. It is proposed a practical way to compress the hierarchy of the multi–resolution mesh using the Corner–Table.

A very significant objective of this work is to point out important aspects to be considered when developing a practical and robust implementation of this simplification technique, and also analyses the influence of the algorithm's parameters on the quality of the result. Finally, this work also compares the SMS method to important related works.

Paper outline. Section 2 *Surface Topology* and section 3 *Stellar Theory* will review some basic concepts of topology and Stellar Theory. Section 4 *Extended Corner–Table Data Structure* and section 5 *Local Operators and Topological Validation* describe the data structure and the topological operators that will be used in the algorithm. Section 6 *Simplification criterion* presents the geometric criterion used to guide the simplification. Section 7 *The Four–Face Cluster Method* presents the original Four–Face cluster algorithm. Section 8 *The Stellar Mesh Simplification Algorithm* introduces the *Stellar Mesh Simplification* algorithm based on the multiple choice technique. Section 9 *Simplification with Multi–Resolution* presents a scheme to encode and decode the hierarchy of the surfaces generated by the simplification algorithm. Finally, section 10 *Results* shows some experimental results for the algorithm's parameters and comparisons.

2 Surface Topology

A *simplex* σ^p of dimension p (p-simplex, for short) is the convex hull of $p+1$ points $\{v_0, ..., v_p\}, v_i \in \mathbb{R}^m$, in general position, i.e., when the vectors $v_1 - v_0, v_2 - v_0, ..., v_p - v_0$ are linearly independent. The points $v_0, ..., v_p$ are called the vertices of σ. A face of σ is the convex hull of some of the vertices of σ and therefore is also a simplex. The simplices of dimensions 2 and 1 will be called, respectively, triangles, edges. If σ is a face of a simplex τ then σ is said to be incident to τ. The boundary of a p-simplex σ, denoted by $\partial\sigma$, is the collection of all of its *proper faces*, i.e., those different from σ itself. Two k-simplices σ and $\rho \in K$ are *adjacent* when $\sigma \cap \rho \neq \emptyset$, and *independent* otherwise. The *valence* of a vertex $v \in K$ is the number of edges which have v as a vertex, and is denoted by $val(v)$.

A simplicial complex K is a finite set of simplices together with all its sub–simplices such that if σ and τ belong to K, then either σ and τ meet at a sub–simplex λ, or σ and τ are independent.

The *underlying polyhedron* $|K| \subset \mathbb{R}^m$ corresponds to the union of the simplices in K. A triangle mesh is the underlying polyhedron of a 2-dimensional simplicial complex.

The *join* $\sigma \operatorname{star} \tau$ of independent simplices σ and τ is the simplex whose vertices are those of σ and τ. The join of complexes K and L, written $K \operatorname{star} L$, is $\{\sigma \operatorname{star} \tau : \sigma \in K, \tau \in L\}$ if the following holds:

1. If $\sigma \in K$ and $\tau \in L$, σ and τ are independent.
2. If $\sigma_1, \sigma_2 \in K$ and $\tau_1, \tau_2 \in L$, then $\sigma_1 \operatorname{star} \tau_1 \cap \sigma_2 \operatorname{star} \tau_2$ is either empty or a face of $\sigma_1 \operatorname{star} \tau_1$ and $\sigma_2 \operatorname{star} \tau_2$.

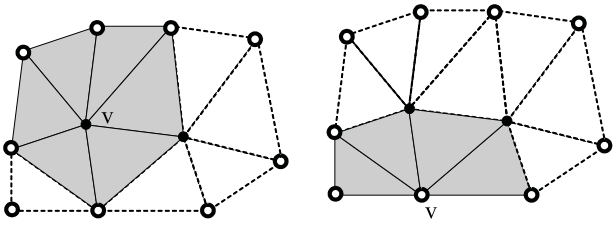
Consider a simplicial complex K and $\sigma \in K$. The local neighborhood of σ is described by the following elements:

- The *open star* of σ is $\operatorname{star}(\sigma, K) = \{\tau \in K : \sigma \text{ is a face of } \tau\}$.
- The *star* of σ is $\operatorname{star}(\sigma, K) = \{\tau \in K : \tau \text{ is a face of an element of } \operatorname{star}(\sigma, K)\}$.
- The *link* of σ is $\operatorname{link}(\sigma, K) = \{\tau \in K : \tau \text{ and } \sigma \text{ are independent and } \sigma \operatorname{star} \tau \in K\}$.

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.

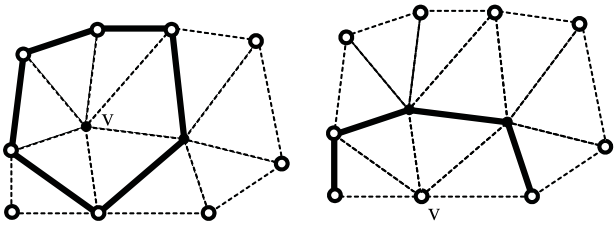
Definition 1 (combinatorial surface) *A triangular mesh M is a combinatorial surface if: Every edge in M is bounding either one or two triangles and the* link *of a vertex in M is homeomorphic either to an interval or to a circle.*

The edges in a combinatorial surface M incident to only one face are called *boundary edges*. A vertex incident to a boundary edge is called a *boundary vertex*. The set of the boundary simplices forms the *boundary* of M and is denoted by ∂M. The boundary of a combinatorial surface is a collection of closed curves. The edges and vertices that are not on the boundary are called, respectively, *interior edges*. Observe that the *link* of an interior edge is the pair of opposite vertices. and *interior vertices*. Figure 2 and Figure 3 illustrate the star and the link of interior and boundary vertices on a surface.



(a) Interior vertex (b) Boundary vertex

Figure 2: *Vertex star.*



(a) Interior vertex (b) Boundary vertex

Figure 3: *Vertex link.*

A combinatorial surface is *orientable* when it is possible to choose a coherent orientation on its edges, i.e., two adjacent triangles induce opposite orientations on their common edge.

From now on, a surface will always mean an oriented combinatorial surface. This is the general case of manifold triangle meshes embedded in $\mathbb{R}^3$. For more details on the following definitions, see [1].

3 Stellar Theory

The previous section defined several concepts of topology that will be used in this work. This section will show how to manipulate the structure of a combinatorial surface *without* modifying its topology, which is the main point of the Stellar theory.Stellar theory was developed in the 1920's, by [1] and [11]. It combines the abstract and piecewise linear approaches to combinatorial topology. More recently, [12] consolidated the theory. The main point of Stellar theory is the study of equivalences between simplicial complexes [8].

The link and the star of a simplex σ provide a combinatorial description of the neighborhood of σ. They can be used to define certain changes in a triangle mesh, without modifying essentially that neighborhood (i.e., perform a combinatorial modification taking care to do not "topologically" change the topology of the surface in $\mathbb{R}^3$). The *stellar operations* provide a such change. They comprise *bistellar moves* and *stellar subdivision*:

Definition 2 *Let K be an n-dimensional simplicial complex. Take an r-simplex $\sigma \in K$, and a $(n-r)$-simplex $\tau \notin K$, such that $\mathrm{link}(\sigma, K) = \partial\tau$. Then, the operation $\kappa(\sigma, \tau)$, called* bistellar move*, consists of changing K by removing $\sigma \,\mathrm{star}\, \partial\tau$ and inserting $\partial\sigma \,\mathrm{star}\, \tau$.*

The bistellar moves are atomic operations that make local changes to the neighborhood of a simplex, while maintaining the integrity of its combinatorial structure. In the case of combinatorial surfaces, there are three types of bistellar moves, for $\dim\sigma = 2, 1, 0$, called 2-move, 1-move, and 0-move. They are shown in Figure 4.

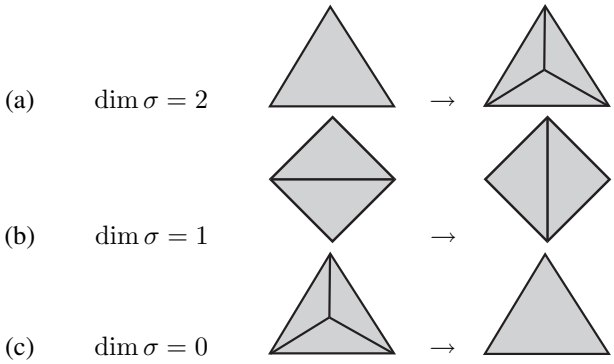


Figure 4: *Two dimensional bistellar moves.*

The fundamental result of the Stellar theory is given by the following theorem:

Theorem 3 *[11, 12] Two combinatorial surfaces are piecewise linearly homeomorphic if and only if they are bistellar equivalent.*

The above result guarantees that bistellar moves can change any triangulation of a closed piecewise linear manifold to any other. A version of this theorem for manifolds with boundary uses all stellar operations, including stellar subdivision [12].

Definition 4 *Let K be a 2-dimensional simplicial complex, take an r-simplex $\sigma \in K$ and a vertex ν in the interior of σ.*

The operation (σ, ν), called a stellar subdivision*, removes $\mathrm{star}(\sigma, K)$ and replaces it with $\nu \,\mathrm{star}\, \partial\sigma \,\mathrm{star}\, \mathrm{link}(\sigma, K)$.*

The inverse operation $(\sigma, \nu)^{-1}$ is called a stellar weld*.*

Notice that, some of the stellar subdivision and welds are also stellar moves, such as $\kappa(\sigma, \nu)$ on Figure 4(a) and $\kappa(\nu, \sigma)$ on Figure 4(b) in the two dimensional case.

The new operation in two dimensions, is the stellar subdivision on edges, called 1-split. Figure 5 shows the interior edge case and Figure 6 the boundary edge case.

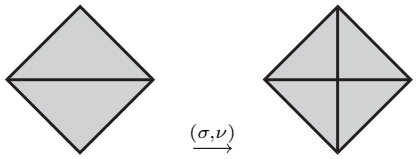


Figure 5: *Two dimensional stellar subdivision on interior edges.*

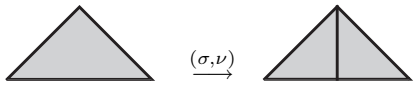


Figure 6: *Two dimensional stellar subdivision on boundary edges.*

Stellar subdivision is a very powerful concept and it is the cornerstone of Stellar theory. Here, only some results of the stellar subdivision theory [1] will be mentioned.

Proposition 5 *Any stellar move, $\kappa(\sigma,\tau)$, is the composition of a stellar subdivision and a weld, namely $(\tau,\nu)^{-1}(\sigma,\nu)$.*

This result can be easily seen through an example as shown in Figure 7.

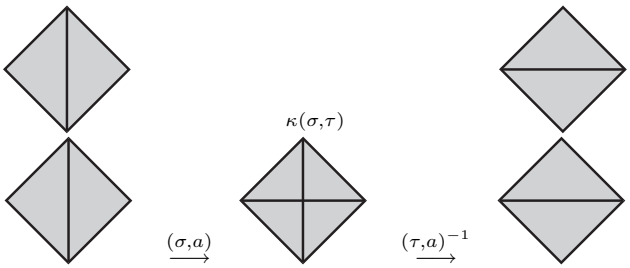


Figure 7: *A bistellar move on an edge can be decomposed into a subdivision and an weld.*

Proposition 6 *Any stellar operation can be decomposed into a finite sequence of elementary stellar operations on* edges.

This result is even stronger than the previous one. It basically allows one to restate the main theorem of Stellar theory only in terms of operations on edges.

4 Extended Corner–Table Data Structure

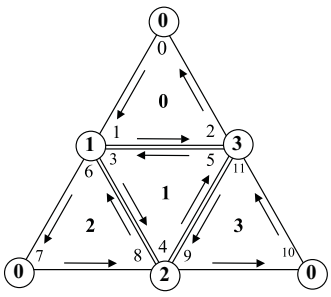
The Corner–Table (CT) is a very concise data structure for triangular meshes [14]. It uses the concept of *corner* to represent the association of a triangle with one of its bounding vertices, or equivalently the association of a triangle with its opposite bounding edge to that corner: it may be viewed as a compact version of the half–edge representation for triangular meshes.

In this data structure, the corners, the vertices and the triangles are indexed by non–negative integers. Each triangle is represented by 3 consecutive corners that define its orientation. For example, corners 0, 1 and 2 correspond to the first triangle, the corners 3, 4 and 5 correspond to the second triangle and so on... As a consequence, a corner with index c is associated with the triangle of index $trig(c) = floor\,(c/3)$.

The Corner–Table data structure represents the geometry of a surface by the association of each corner c with its geometrical vertex index V$[c]$.

Assuming a counter–clockwise orientation, for each corner c, the $next(c)$ and $prev(c)$ corners on its triangle boundary are obtained by the use of the following expressions: $next(c) = 3 \times trig(c) + [(c+1) \bmod 3]$, and $prev(c) = 3 \times trig(c) + [(c+2) \bmod 3]$.

The edge–adjacency between the neighboring triangles is represented by associating to each corner c, its opposite corner O$[c]$, which has the same opposite edge. This information is stored in two integer arrays, called the V and O tables. Figure 8 shows an example of a Corner–Table representation for a tetrahedron.



V[0]	= 0	O[0]	= 4
V[1]	= 1	O[1]	= 9
V[2]	= 3	O[2]	= 8
V[3]	= 1	O[3]	= 10
V[4]	= 2	O[4]	= 0
V[5]	= 3	O[5]	= 7
V[6]	= 1	O[6]	= 11
V[7]	= 0	O[7]	= 5
V[8]	= 2	O[8]	= 2
V[9]	= 2	O[9]	= 1
V[10]	= 0	O[10]	= 3
V[11]	= 3	O[11]	= 6

Figure 8: *Tetrahedron example.*

The CT concisely represents the connectivity of a triangular mesh using only the arrays O and V. To represent the mesh geometry, an array G is used to store the geometry of the vertices (coordinates, normals,...).

Notice that, the number of entries of the O and V arrays is the number of corners on the mesh, i.e., three times the number of triangles. And the number of entries of the G array is the number of vertices on the mesh.

Where are the edges? The CT does not need an explicit representation for edges because they are implicitly represented by its opposite corners. A corner c also represents an edge whose endpoints are $V[prev(c)]$ and $V[next(c)]$.

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.

To obtain the incidences and adjacencies of edges, two arrays CE and EC could be temporary allocated. The first array stores one corner $c = CE[e]$ that associates the edge e with one of its opposite corners c and the second array stores the edge $e = EC[c]$ that associates the corner c with its opposite edge e. These arrays will be used only to illustrate the incidences and adjacencies of an edge and can be allocated in linear time from the arrays O and V.

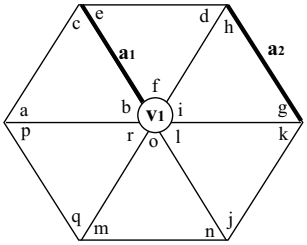


Figure 9: *Edge representation.*

In Figure 9 the edge a_1 is interior to the mesh and it could be represented by the corners a or d, while the edge a_2 is a boundary edge and can only be represented by the corner i.

It is important to notice that in the SMS algorithm the CE and EC arrays are not necessary. But they can be very useful to develop an efficient implementation of other algorithms that are based on the edge–collapse operation.

Toward efficiency on vertex queries. In the SMS algorithm, a very important step is to obtain the vertex star efficiently, since it is based on edge–weld. Therefore, it is possible to extend the original CT by allocating an extra integer array VC that for each vertex v stores an index of a corner whose vertex is v. For example, in the Figure 9 the vertex v_1 can be represented by the corner o. Only one corner is sufficient because all the incidence and adjacency relations can be obtained through the use of the O and V arrays according to the following algorithm.

Given a vertex v, the algorithm 1 gets the corner c associated to v and, in time O(*val(v)*), obtain the list of adjacent vertices and incident edges and faces, denoted respectively by $v < V >, v < E >$ and $v < F >$.

Algorithm 1: $\mathrm{star}(v)$

$c = j = VC[v]$;
$v < V >= \emptyset$; $v < E >= \emptyset$; $v < F >= \emptyset$;
Do
$v < V >= v < V > + V[next(j)]$;
$v < E >= v < E > + EC[next(j)]$
$v < F >= v < F > + \lfloor j/3 \rfloor$;
$j = next(right(j))$;
While ($j! = c$)

5 Local Operators and Topological Validation

The SMS algorithm is based on two local topological operators: the *Edge–Flip*, and the *Edge–Weld*. This section will describe not only those two, but also the more classical *Edge–Collapse* operator in order to compare them.

Edge–Collapse: This operator consists in removing an edge $e = (u, v)$ of a surface S, identifying its vertices to a unique vertex $\overline{v}$. From a combinatorial viewpoint, this operator will remove 1 vertex, 3 edges and 2 faces from original mesh, thus preserving its Euler characteristic. From a geometric viewpoint, the new position of the vertex $\overline{v}$ can be computed with the geometry around u and v.

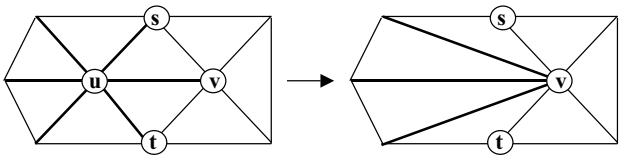


Figure 10: *Edge–Collapse.*

Let s and t be the vertices opposite to $e = (u, v)$, which is the edge to be collapsed (see Figure 10). Choosing the inner edges according to the following *collapse condition* will guarantee the topological consistency of this operation .

Lemma 7 (Collapse Condition) *[5] Let S be a combinatorial 2–manifold. The collapse of an edge $e = (u, v) \in S$ preserves the topology of S if the following conditions are satisfied:*

1. $\mathrm{link}(u) \cap \mathrm{link}(v) = \mathrm{link}(e)$*;*
2. *If u and v are both boundary vertices, e is a boundary edge;*
3. *S has more than 4 vertices if neither u nor v are boundary vertices, or S has more than 3 vertices if either u or v are boundary vertices.*

Figure 11(left) shows a valid operation and Figure 11(right) an invalid. Figure 12 illustrates two examples of invalid collapse operations on boundary edge.

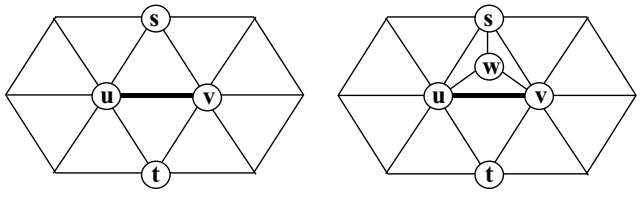


Figure 11: *Examples of a valid and an invalid collapse operation by the first condition.*

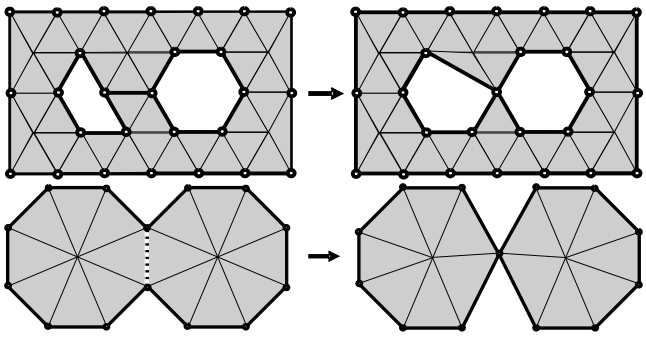


Figure 12: *Invalid edge-collapse operations by the second condition.*

Edge–Flip: This operation consists in transforming a two–face cluster into another two–face cluster by swapping its common edge.

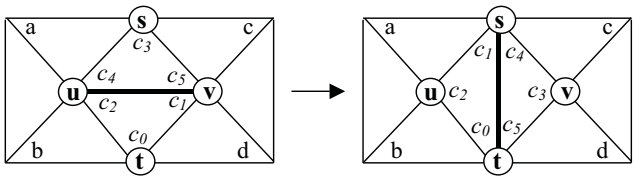


Figure 13: *Edge–Flip.*

Let consider the interior edge $e = (u, v)$ of a surface S and s and $t \in S$ the two vertices opposed to e (see Figure 13). The Edge–Flip operation will replace e by (s, t), and replace the 2 triangles incident to e by (u, s, t) and (v, t, s). The condition to apply this operator is the following.

Lemma 8 (Flip Condition) *[5] Let S be a combinatorial 2–manifold. The flip of an interior edge that replaces $e = (u, v) \in S$ by (s, t) preserves the topology of S if and only if $(s, t) \notin S$.*

In the Corner–Table data structure, each edge can be univocally represented by one of its opposite corners. This is used by the Algorithm 2 to perform the *Edge–Flip* operation on the edge opposite to the corner c_0.

Algorithm 2: Edge–Flip(c_0)

// Label incident corners
$c_2 = prev(c_0)$; $c_1 = next(c_0)$;
$c_3 = O[c_0]$; $c_4 = next(c_3)$; $c_5 = prev(c_3)$;
$a = O[c_5]$; $b = O[c_2]$; $c = O[c_4]$; $d = O[c_1]$;
// Label incident vertices
$t = V[c_0]$; $v = V[c_1]$; $s = V[c_3]$;
// Perform swap
$V[c_1] = s$; $V[c_3] = v$; $V[c_4] = s$; $V[c_5] = t$;
// Reset opposite corners
$O[c_2] = c_3$; $O[c_3] = c_2$;
$O[c_0] = a$; $O[a] = c_0$;
$O[c_4] = d$; $O[d] = c_4$;
$O[c_5] = c$; $O[c] = c_5$;

Edge–Weld. This operation consists in transforming a four–face cluster into a two–face cluster by removing its central vertex.

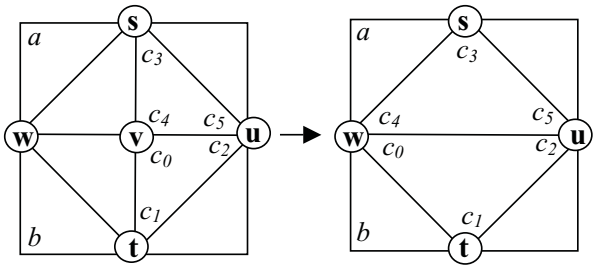


Figure 14: *Edge–Weld.*

Consider an interior valence–4 vertex v, adjacent to the vertices s, t, w and u (see Figure 14). The star of v forms a four–face cluster. The Edge–Weld operation removes the vertex v and re–triangulates the quadrangle by splitting it. The dividing edge e can be either (w, u) or (s, t). The result is a two–face cluster made of the two faces incident to e.

Corollary 9 *Consider a combinatorial 2–manifold M, and an interior vertex v of M with valence 4. With the notations of Figure 14, the removal of the vertex v by the Edge–Weld operation preserves the topology of M if and only if there is no edge in M connecting w to u.*

Proof: Consider s, w, t and u, in this order, the vertices adjacent to the interior vertex v. Removing v is equivalent to an Edge–Collapse operation on the edge $e = (v, u)$. Therefore, the topology of M is preserved if and only if the *collapse condition* is satisfied: $\text{link}(v) \cap \text{link}(u) = \{s, t\} = \text{link}(e)$. As $\{u, w\} = \text{link}(v) \backslash \text{link}(e)$, this condition is valid if and only if w does not belong to $\text{link}(u)$. This means that there is no edge connecting w to u. ■

Given a mesh represented by a Corner–Table, the Algorithm 3 performs the removal of the vertex incident to the corner c_0:

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.

Algorithm 3: Edge–Weld(c_0)

// Assign incidences
$c_1 = next(c_0)$; $c_2 = prev(c_0)$;
$c_4 = next(O[c_1])$; $c_5 = prev(O[c_1])$;
$a = O[next(O[c_5])]$; $b = O[prev(O[c_2])]$;
// Perform vertex removal
$V[c_0] = V[O[c_2]]$; $V[c_4] = V[c_0]$;
// Reset opposite corners
$O[c_5] = a$; $O[a] = c_5$;
$O[b] = c_2$; $O[c_2] = b$;

When an edge–weld is applied to remove a boundary vertex with valence 3, the edge-collapse condition should be observed.

As a consequence of proposition 5, one can prove that the *Edge–Collapse* operation can be decomposed into a sequence of *Edge–Flips* operations, followed by one *Edge–Weld* operation. Figure 15 shows a sequence example.

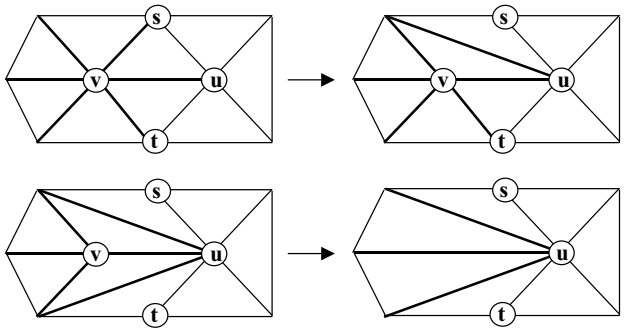


Figure 15: *Edge–Collapse decomposition*

The combination of Edge–Weld and Edge–Flip stellar operations provides more flexibility than the edge–collapse operation itself, as another sequence could remove the vertex v with a different configuration of the star of vertex u.

To illustrate that fact, consider the example of Figure 15. First, by some geometric criteria, some Edge–Flip operations are used to reduce the valence of vertex v to four. After that, an Egde–Weld operation is applied.

6 Simplification criterion

Simplification algorithms such as the Four–Face Cluster algorithm (FFC) try to remove the maximal number of faces while minimizing the geometrical distortion. This is a difficult optimization problem, essentially because there are different measures of the geometrical distortion. On one side, global error measures such as the volume or the Hausdorff distance imply a global optimization strategy, which involves more expensive techniques such as re-meshing. On the other side, local error measures such as the Quadric Error Metric (QEM) allow efficient greedy strategies which give generally low total distortion.

Quadric Error Metric. In the original FFC algorithm, each local simplification introduces a geometric error which is computed using the QEM [4], as follow:

Let v be a vertex of M^j, and $f_i \in \mathrm{star}(v)$ a face incident to v. Let a_i be the area of f_i and $p_i = (n_x, n_y, n_z, d)$ the plane supporting f_i. The *fundamental error quadric* Q_i is defined in [4] by:

$$Q_i = p_i p_i^T = \begin{pmatrix} n_x^2 & n_x n_y & n_x n_z & n_x d \\ n_x n_y & n_y^2 & n_y n_z & n_y d \\ n_x n_z & n_y n_z & n_z^2 & n_z d \\ n_x d & n_y d & n_z d & d^2 \end{pmatrix}$$

It can be used to compute the squared distance $d(w)$ of a point w to the plane p_i:

$$d_i(w) = w^T \left(p_i p_i^T\right) w = w^T Q_i w$$

Garland and Heckbert compute their quadric error metric by adding all of those distances $d_i(w)$ for all face f_i.

For boundary edge collapse, the QEM should be modified in order to preserve the geometrical aspect of the boundary. [4] suggests to construct, for each boundary edge, a quadric that corresponds to the plane perpendicular to the boundary (see 16) and adds each quadric related to the extreme vertices of such edge. Using this strategy, the edges that contribute to the geometric aspect of the boundary are penalized with a high cost of contraction.

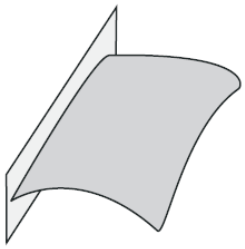


Figure 16: *Plane perpendicular to the boundary.*

Vertex cost. The FFC algorithm assigns a quadric Q_v to each vertex v of the mesh, which is computed as the weighted sum of the quadrics Q_i associated with each face f_i incident to v:

$$Q_v = \sum a_i Q_i$$

The error introduced by removing a vertex v from the mesh, according to [15], considers each Edge–Flip and Edge–Weld cost:

$$E(v) = \alpha C(v) + \beta S(v)$$

This cost balances the vertex removal distortion $C(v)$ and the swap distortion $S(v)$. Velho, in its original work [15], suggests to adopt $\alpha = 0.75$ and $\beta = 0.25$ as default values for the weights. Section 10 shows experimental results for different values of α and β and one can observe that the values $\alpha = 0.2$ and $\beta = 0.8$ also show nice rate/distortion

curves. In pratice, there are many different conbination for α and β in order to obtain nice results, which suggests that they are, separatlly, good strategyes for local error estimation.

The cost $C(v)$ of removing a vertex v of valence 4 is defined as:

$$C(v) = \min\left\{u^T (Q_v + Q_u)\, u, \;\; u \in \text{link}(v)\right\}$$

The cost $S(v)$ is the sum of the cost of each Edge–Flips in $Star(v)$ used to bring v to valence 4. The cost of each Edge–Flip is based on two measures: the aspect ratio [17] and the dihedral angle. The sequence of independent Edge–Flips to bring v to valence 4 are chosen in such a way to minimize the cost.

7 The Four–Face Cluster Method

The Four–Face Cluster algorithm (FFC) is based on the Edge–Weld and Edge–Flip operators. Since the Edge–Collapse operation is equivalent to a sequence of those stellar operators, Edge–Collapse based method is a special case of the FFC method. However, stellar operations are more flexible. In the case of the Edge–Collapse / Vertex–Split, there are many possible sequences of Edge–Flips leading to a final Edge–Weld. Therefore, the order of those Edges–Flips can be chosen to improve the quality of the mesh (for example improving aspect ratio or preserving dihedral angle), which is more tricky in the Edge–Collapse operations. As a conclusion, the Four–Face Cluster (FFC) algorithm is more flexible than Edge–Collapse based algorithms.

Algorithm's outline. The FFC algorithm constructs a hierarchy of meshes $\left(M^0, M^1, \ldots, M^n\right)$ with a decreasing number of elements (vertices, edges and faces). In this hierarchy, surface M^0 is the original surface M, and each surface M^j, $j = 1 \ldots n$, corresponds to the level of detail j of M. The following paragraph describes how to obtain such hierarchy.

In the initialization step for each level of detail j, all the vertices of M^{j-1} are marked as valid for removal. Then, the FFC algorithm selects a marked vertex v to be removed from the surface M^{j-1}. When the valence of the vertex v is not 4, it is necessary to apply a sequence of Edge–Flip operations to bring its valence to 4. The four vertices on the star of the modified vertex v are then unmarked. They will not be valid for further removal until the next level of detail $j+1$. Next, the algorithm removes the valence 4 vertex v using an Edge–Weld operation. Figure 15 illustrates this sequence of operations. When all the vertices are unmarked, the resulting surface M^j corresponds to level of detail j inside the hierarchy of surfaces. On entering the next step, the algorithm marks all the remaining vertices of M^j in order to create M^{j+1} and so on. The selection of the vertices to be removed is performed according to geometrical costs, such as the Quadric Error Metric [4].

Implementation. The original implementation uses a priority queue, which requires computing the costs of all vertices before beginning the simplification. Moreover, at each simplification step, the FFC algorithm needs to re–compute the cost of all vertices involved in the local operation and to update the corresponding positions in the priority queue. Algorithm 4 gives the pseudo–code for the Four–Face Cluster Simplification Algorithm introduced in [15]. It generates n levels of resolution of a given mesh M.

Algorithm 4: SimplifyFFC(M, n)

```
assign quadrics;
for all(v ∈ M) do
   compute E(v)
for (j = 1 to n) do
   mark all vertices as valid for removal
   insert all vertices in the priority queue
   while (queue is not empty) do
      get v from queue
      if (v marked) then
         perform edge swaps until Deg(v) = 4
         unmark the vertices w ∈ link(v)
         remove vertex (v)
         re–compute the errors Q_u and Q_w
         update queue for w ∈ link(u) ∪ link(w)
```

8 The Stellar Mesh Simplification Algorithm

An alternative to simplification methods based on priority queue was presented by Wu and Kobbelt in [16]. They propose, instead of using the priority queue, to choose the element to be simplified inside a reduced, randomly selected set of d elements. This probabilistic optimization strategy is motivated by the fact that, when simplifying high resolution meshes, most of the vertices will be removed anyway. As a consequence, it would not be necessary to choose a vertex with lower cost among all possible candidates at each simplification step.

Algorithm's outline. The basic idea of this Multiple–Choice technique is to obtain a small random subset of d edges to be collapsed, say $d = 8$, and perform the collapse for the edge with the lower cost among them. The implementation proposed in this work is an adaptation of this idea, since it computes costs on vertices and not on edges. It obtains a random subset of vertices to be simplified and performs the simplification for the vertex with the lower cost of this set of candidates. This strategy leads to a good choice of the vertex to be simplified, except for a rare probability that all d random vertices were among the vertices that should not be removed. Assuming that it simplifies a mesh down to 5% of its original complexity, it is expected that the resulting (not removed) vertices will have high costs. The probability of choosing one of the high–cost vertices on a random subset of $d = 8$ vertices is actually small enough:

$$\left(\frac{5}{100}\right)^8 \approx 4 \times 10^{-11}$$

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.

Implementation. Using this technique it is not necessary to sort the vertices by their cost, avoiding the expensive use of a priority queue. Since, at each randomly selected subset the costs of the d vertices will be recomputed, it is also not necessary to recompute the cost for the vertices on the neighborhood of the removed vertex. This allows a simpler implementation and improves the algorithm's performance. The Algorithm 5 shows the pseudo–code to generate n levels of resolution for a mesh M using this process:

Algorithm 5: SimplifySMS(M, n)
```
assign quadrics;
d = 8;
for (j = 1 to n) do
   mark vertices as valid for removal
   while (exist a valid vertex) do
      for (i = 1 to d) do
         v_i =valid random vertex
         if E(v_i) < E(v) then v = v_i
      perform edge swaps until Deg(v) = 4
      unmark the vertices w ∈ link(v)
      remove vertex (v)
      re–compute quadrics Q_u and Q_w
```

At the end of a level, when the number of valid vertices is less then d, one or more vertices can be selected more than once in order to remove all valid vertices.

9 Simplification with Multi–Resolution

One objective of this section is to introduce two strategies to encode the hierarchy of meshes $\left(M^0, M^1, .., M^n\right)$ generated by the SMS algorithm: the *parallel* and the *sequential* encoding. The main difference between those two strategies resides in the type of hierarchical structure they produce. The parallel encoding produces a multi–resolution mesh with separate levels of details, whose resolution changes globally on each level. The sequential encoding produces a progressive mesh, whose resolution changes locally inside each level.

Multi–resolution representation. The Corner–Table data structure uses only two arrays (O and V) of integers to represent the surface topology and a third array (G) to store the geometry of the mesh. In order to represent the hierarchy of surfaces, the following strategy is adopted. The connectivity of the surface M^j is represented by the arrays O^j and V^j, while its geometry uses the original array G for all levels of detail, since the simplification process does not modify the indexes of the vertices. Therefore, the procedure that changes from one level to another simply restores the O and V arrays corresponding to the desired level.

Parallel Encoding. This strategy produces a multi–resolution mesh whose resolution changes globally at each level. Parallel encoding is useful when the user wants to move from one level of resolution to another. In such application, it is sufficient to allocate the memory necessary to represent the surface of a certain level. Thus, parallel encoding simply needs to encode the surface M^j. This encoding is implemented in a compressed manner (with less than 2 bits per triangle) using the Edgebreaker compression scheme.

The Edgebreaker is adopted because there is a very concise and simple implementation of this compression scheme for surfaces with handles using the Corner–Table data structure [9]. A simple modification of this implementation has been done to add a fourth attribute to vertex geometry, its index on the G table of the original surface. Maintaining these indexes for the vertices at each level of detail enable us to interchange the use of parallel and sequential encoding. For example, in the application one can move directly to the level of resolution 5 using the parallel encoding and then, in a progressive way, return to the level 4 by the use of the sequential encoding.

Sequential Encoding. The main idea in the sequential case is to encode each stellar operation, in order to recover the history of the Edge–Flips and Edge–Welds operations at each vertex simplification. This strategy is similar to the Progressive Mesh encoding [6], which encodes the history of Edge–Collapse operations for their simplification algorithm. The main idea of this encoding is to obtain a refinement process by reading in the reverse order the history of stellar operations performed by the SMS algorithm.

In the sequential encoding, a string of integers is generated to represent the sequence of operations performed on each vertex simplification. Figure 17 illustrates a vertex simplification with the encoding of the corresponding operations.

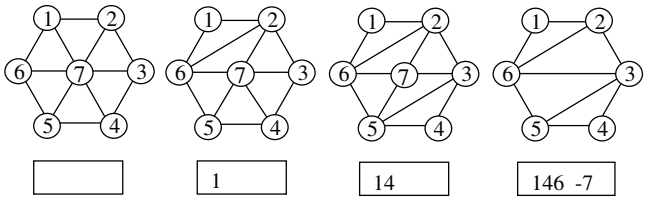


Figure 17: *Encoding of a vertex simplification.*

Let v be the vertex to be removed, and $e = (u, v) \in \mathrm{star}(v)$ an edge to be swapped. Since each e has v as extremity, the scheme encodes the numbers index of u in $\mathrm{star}(v)$ to represent an Edge–Flip operation. The Edge–Weld operation is encoded by the use of two more integers: one for the resulting edge of the vertex removal and the other integer to encode the original index of the removed vertex.

In order to interchange the parallel encoding with the sequential encoding, the scheme writes each level of detail in a separate file. As a consequence, one can start the progressive refinement process at any level of detail obtained by the simplification. In the Progressive Mesh [6], the starting surface for refinement is restricted to the base mesh.

	FFC		SMS	
Operation	$\mathbf{t}(s)$	**%**	$\mathbf{t}(s)$	**%**
Assign Quadrics	0.110	1.26	0.110	1.95
Compute $E(v)$	7.880	90.51	4.930	87.54
Update Queue	0.360	4.14	–	–
Choose d random v	–	–	0.340	6.04
Edge Swaps	0.165	1.90	0.088	1.56
Vertex Removal	0.170	1.95	0.141	2.50
Recompute Q_u and Q_w	0.021	0.24	0.023	0.41
Total	8.706	100	5.632	100

Table 1: *Algorithm's steps comparison.*

	# Triangles		Running time		Ratio
Model	**Input**	**Output**	**FFC**	**SMS**	$\frac{\textbf{SMS}}{\textbf{FFC}}$
Cow	5804	202	4.680	2.780	0,59
Terrain	5708	190	4.520	2.690	0,59
Torus	14144	344	9.100	5.110	0,56

Table 2: *Total time comparison.*

10 Results

The results presented in [16] shows that the performances of mesh simplification algorithms using the Multiple–Choice technique are 2.5 times faster than using a priority queue, with similar outputs. Furthermore, the Multiple–Choice technique leads to a simpler implementation, since there is no need for a priority queue construction or of re–computation at each simplification step. Similarly, the experiments below show that the SMS is 40% faster than the original Four–Face Cluster algorithm with the same order of distortion.

Table 1 shows the running time of each routine during the simplification, using the Four–Face Cluster (FFC) and Stellar Mesh Simplification (SMS) algorithms. Table 2 shows the total running time comparison (in sec) for simplifying some models.

The rate/distortion curve illustrated in Figure 18(a) is used to study the influence of the α and β parameters on both FFC and SMS algorithm compare the FFC and SMS algorithms. They correspond, respectively, to the weights for the Edge–Welds and Edge–Flips operations. In the example, the *Bunny* model is simplified to 10% of its size. The parameters α and β are studied in the following way: the α parameter varies from 0.0 to 1.0 by a step of 0.1, and β is set as $1-\alpha$. As a conclusion, there are many different good conbinations for α and β that gives better results. It is important to notice that all rate/distortion curves generated in this work are obtained by the use of METRO [3].

Figure 18 and Figure 1 show the influence of parameter d in the algorithm, where d is the number of vertices used on the multiple-choice step. For each d from 2 to 20, the algorithm is executed in order to simplify the *Dino* original surface with 7400 triangles to a surface with 500 triangles. Figure 18 shows the algorithm total time and the absolute maximum geometric error, both as a function of d. The absolute maximum geometric error was computed using METRO [3]. Figure 1 shows the original *Dino* model and its simplified versions using 3 different valued for d. As a conclusion, setting d as 8 is really a nice decision, since the total time varies linearly with d and there is no substantial gain on error reduction for d greater than 8.

Figure 20 shows two interesting implicit surfaces examples obtained by the use of a Marching Cube algorithm [7]. Notice that the surfaces on those figures have the same topological type. One can visually observe that the geometry of both objects are very well preserved after simplification, including the sharp edges, although the algorithm is performing a probabilistic optimization.

Figure 19 shows the rate/distortion curves to compare the Garland and Heckbert's [4], Wu and Kobbelt's [16], FFC and SMS algorithms applied to the *Bunny* model. Notice that on both figures the order of distortion are basically the same for all algorithms and that the original algorithm of Garland and Heckbert [4] has a better rate/distortion performance when compared to the multiple-choice technique proposed by Wu and Kobbelt [16]. In opposition to this fact, the SMS algorithm shows to have a better performance when compared to the original FFC. Actually, this observation holds for all tested examples illustrated in this work.

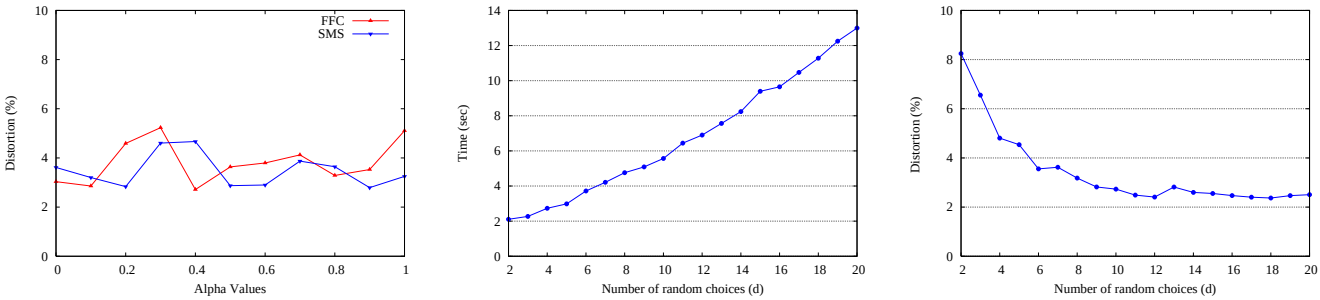
Finally, Figure 21 illustrates four surface models and its simplified versions to visually compare the FFC and the SMS algorithms. Notice that, as in [16], the results got in both methods are similar. The examples given are *Bunny* model, *Cow* model, linked tori model and other implicit surface model.

11 Conclusions and Future works

This paper presented the Stellar Mesh Simplification method, which is a fast implementation for the Four–Face cluster algorithm. Its implementation was described using the Corner–Table data structure to represent the surface topology, which is very suitable to obtain a simple and concise implementation of the method. The new strategy proposed to encode and decode hierarchy of surfaces generated by the mesh simplification is a very practical tool for a wide range of applications. This work also describes some elements of Stellar Theory, which is powerful mathematical tool for topological mesh operations that helped in stating conditions to safely apply the Edge–Weld and Edge–Flip operators. In conclusion, this work is a practical combination of several techniques to obtain a robust and efficient implementation of a nicely working scheme. Moreover, it provides an analysis of the parameters, that completes both works of Velho [15] and Wu and Kobbelt [16].

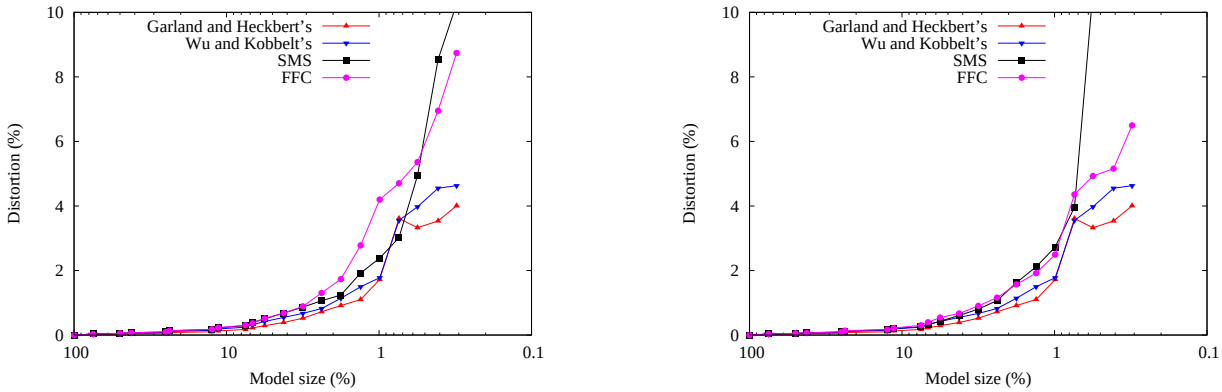
There are three directions to continue this work. One would consist in enhancing the efficiency of compression scheme for the sequential encoding. Another one would use this simplification algorithm to obtain a parameterization scheme for surfaces, which is a very active area of research in Computer Graphics. The last one would investigate an out–of–core implementation of this algorithm, taking advantage of the simplicity of the Corner–Table data structure.

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.



(a) Influence of α and β parameters. (b) Total algorithm time as a function of d. (c) Geometric error as a function of d.

Figure 18: *Influence of parameters*



(a) ($\alpha = 0.75$,$\beta = 0.25$) (b) ($\alpha = 0.2$,$\beta = 0.8$)

Figure 19: Bunny *rate/distortion curves for Garland and Heckbert's, Wu and Kobbelt's, FFC and SMS algorithms.*

References

[1] J. W. Alexander. The combinatorial theory of complexes. *Annals of Mathematics*, 31:219–320, 1930.

[2] P. Cignoni, C. Montani and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[3] P. Cignoni, C. Rocchini and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[4] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Computers & Graphics*, 31:209–216, 1997.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald and W. Stuetzle. Mesh optimization. In *Siggraph*, volume 27, pages 19–26. ACM, 1993.

[6] H. Hoppe. Progressive meshes. In *Siggraph*, pages 99–108, New Orleans, Aug. 1996. ACM.

[7] T. Lewiner, H. Lopes, A. W. Vieira and G. Tavares. Efficient implementation of Marching Cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[8] W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Kirbyfest*, volume 2 of *Geometry and Topology Monographs*, pages 299–320, 1999.

[9] H. Lopes, J. Rossignac, A. Safonova, A. Szymczak and G. Tavares. Edgebreaker: a simple compression for surfaces with handles. In C. Hoffman and W. Bronsvort, editors, *Solid Modeling and Applications*, pages 289–296, Saarbrücken, Germany, 2002. ACM.

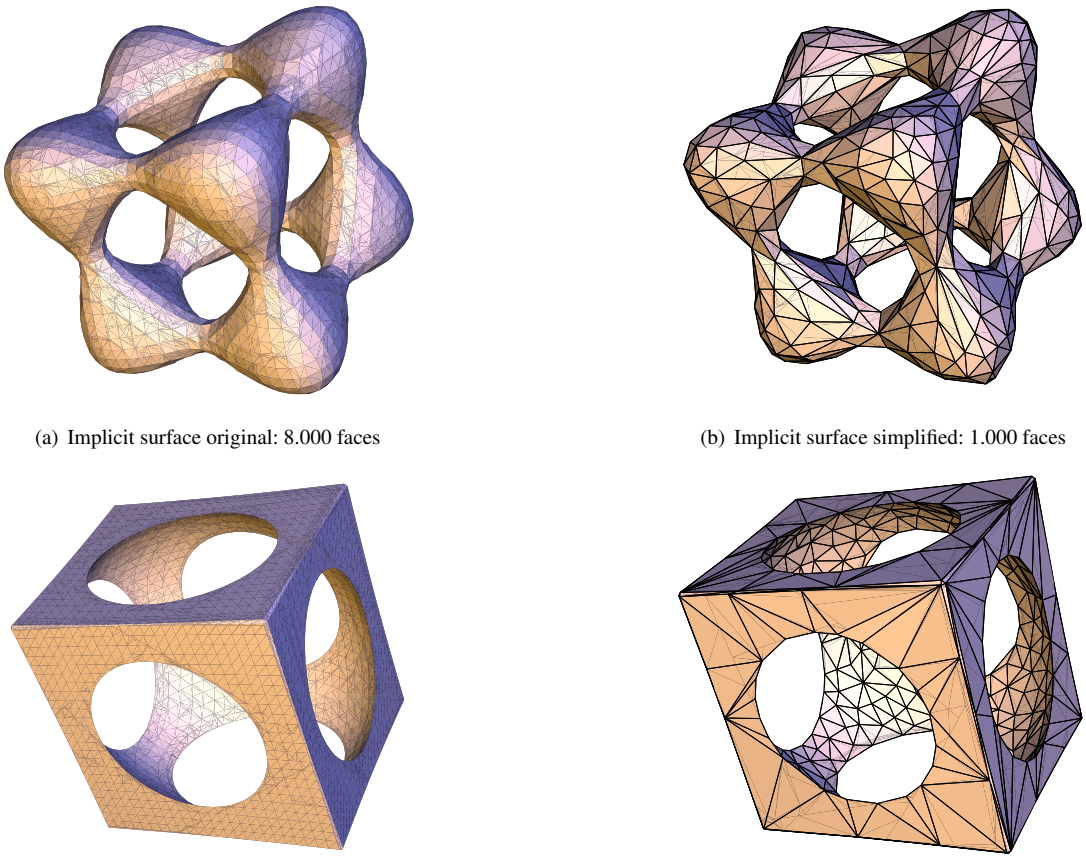
[10] W. E. Lorensen and H. E. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. In *Siggraph*, volume 21, pages 163–169. ACM, 1987.

[11] M. H. A. Newman. On the foundations of combinatorial analysis situs. *Royal Academy*, 29:610–641, 1926.

[12] U. Pachner. PL homeomorphic manifolds are equivalent by elementary shellings. *European Journal of Combinatorics*, 12:129–145, 1991.

[13] E. Puppo and R. Scopigno. Simplification, LOD and multireolution – principles and applications. In *Eurographics Tutorial*, page PS97 TN4, 1997.

[14] J. Rossignac, A. Safonova and A. Szymczak. 3S compression made simple: Edgebreaker on a corner–table.



(a) Implicit surface original: 8.000 faces

(b) Implicit surface simplified: 1.000 faces

(c) CSG model original: 8.000 faces

(d) CSG model simplified: 800 faces

Figure 20: *Simplification examples.*

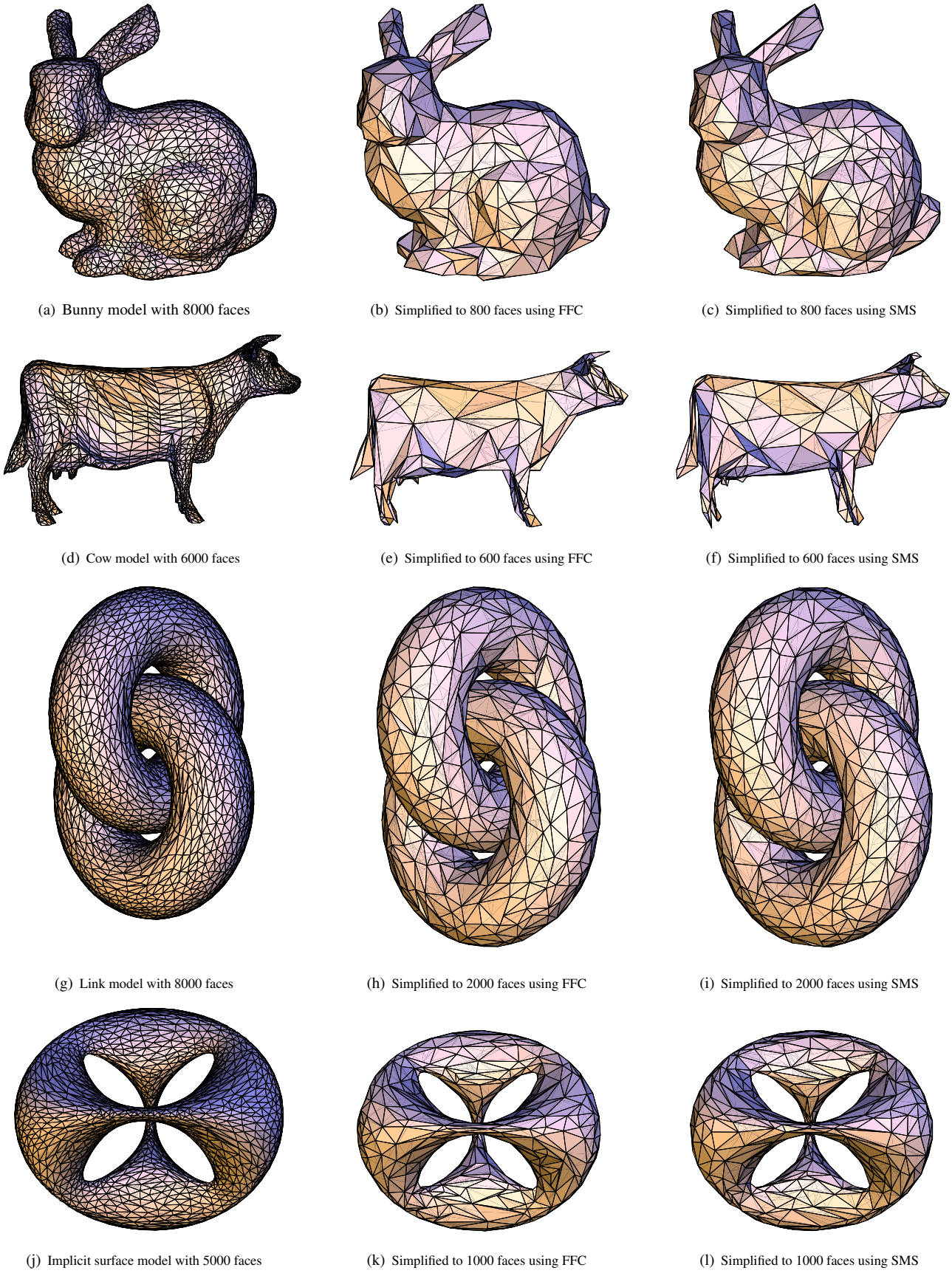
In *Shape Modeling International*, pages 278–283. IEEE, 2001.

[15] L. Velho. Mesh simplification using four–face clusters. In *Shape Modeling International*, pages 200–208. IEEE, 2001.

[16] J. Wu and L. Kobbelt. Fast mesh decimation by multiple–choice techniques. In *Vision, Modeling and Visualization*, pages 241–248. IOS Press, 2002.

[17] A. Guéziec, F. Bossen, G. Taubin and C. Silva. Efficient compression of non–manifold polygonal meshes. *Computational Geometry: Theory and Applications*, 14(1–3):137–166, 1999.

The corresponding work was published in Computer Graphics Forum, volume 23, number 4, pp. 825–838. Blackwell, october 2004.



(a) Bunny model with 8000 faces (b) Simplified to 800 faces using FFC (c) Simplified to 800 faces using SMS

(d) Cow model with 6000 faces (e) Simplified to 600 faces using FFC (f) Simplified to 600 faces using SMS

(g) Link model with 8000 faces (h) Simplified to 2000 faces using FFC (i) Simplified to 2000 faces using SMS

(j) Implicit surface model with 5000 faces (k) Simplified to 1000 faces using FFC (l) Simplified to 1000 faces using SMS

Figure 21: *Visual comparisons of FFC and SMS algorithms results.*